# Approximating the Exponential, the Lanczos Method and an $\tilde{O}(m)$-Time Spectral Algorithm for Balanced Separator

[Extended Abstract]

Lorenzo Orecchia
MIT
Cambridge, MA, USA.
orecchia@mit.edu

Sushant Sachdeva*
Princeton University
Princeton, NJ, USA.
sachdeva@princeton.edu

Nisheeth K. Vishnoi
Microsoft Research
Bangalore, India
nisheeth.vishnoi@gmail.com

## ABSTRACT

We give a novel spectral approximation algorithm for the balanced (edge-)separator problem that, given a graph $G$, a constant balance $b \in (0, 1/2]$, and a parameter $\gamma$, either finds an $\Omega(b)$-balanced cut of conductance $O(\sqrt{\gamma})$ in $G$, or outputs a certificate that all $b$-balanced cuts in $G$ have conductance at least $\gamma$, and runs in time $\tilde{O}(m)$. This settles the question of designing asymptotically optimal spectral algorithms for balanced separator. Our algorithm relies on a variant of the heat kernel random walk and requires, as a subroutine, an algorithm to compute $\exp(-L)v$ where $L$ is the Laplacian of a graph related to $G$ and $v$ is a vector. Algorithms for computing the matrix-exponential-vector product efficiently comprise our next set of results. Our main result here is a new algorithm which computes a good approximation to $\exp(-A)v$ for a class of symmetric positive semidefinite (PSD) matrices $A$ and a given vector $v$, in time roughly $\tilde{O}(m_A)$, independent of the norm of $A$, where $m_A$ is the number of non-zero entries of $A$. This uses, in a non-trivial way, the result of Spielman and Teng on inverting symmetric and diagonally-dominant matrices in $\tilde{O}(m_A)$ time. Finally, using old and new uniform approximations to $e^{-x}$ we show how to obtain, via the Lanczos method, a simple algorithm to compute $\exp(-A)v$ for symmetric PSD matrices that runs in time roughly $O(t_A \cdot \sqrt{\|A\|})$, where $t_A$ is the time required for the computation of the vector $Aw$ for given vector $w$. As an application, we obtain a simple and practical algorithm, with output conductance $O(\sqrt{\gamma})$, for balanced separator that runs in time $\tilde{O}(m/\sqrt{\gamma})$. This latter algorithm matches the running time, but improves on the approximation guarantee of the Evolving-Sets-based algorithm by Andersen and Peres for balanced separator.

*Part of this work was done while this author was interning at Microsoft Research India, Bangalore.


*STOC'12,* May 19–22, 2012, New York, New York, USA.


## Categories and Subject Descriptors

G.2.2 [**Mathematics of Computing**]: Discrete Mathematics—*Graph Theory, Graph Algorithms*; F.2.2 [**Analysis Of Algorithms And Problem Complexity**]: Nonnumerical Algorithms and Problems

## General Terms

Algorithms, Theory

## Keywords

Spectral Algorithms, Balanced Graph Partitioning, Matrix Exponential, Lanczos Method, Uniform Approximation.

## 1. INTRODUCTION AND OUR RESULTS

### 1.1 Balanced Edge Separator

The Balanced Separator problem (BS) asks the following decision question: given an unweighted graph $G = (V, E)$, $V = [n], |E| = m$, a constant balance parameter $b \in (0, 1/2]$, and a target conductance value $\gamma \in (0, 1)$, does $G$ have a $b$-balanced cut $S$ such that $\phi(S) \leq \gamma$? Here, the conductance of a cut $(S, \bar{S})$ is defined to be $\phi(S) \stackrel{\text{def}}{=} |E(S,\bar{S})| / \min\{\mathsf{vol}(S), \mathsf{vol}(\bar{S})\}$, where $\mathsf{vol}(S)$ is the sum of the degrees of the vertices in the set $S$. Moreover, a cut $(S, \bar{S})$ is $b$-balanced if $\min\{\mathsf{vol}(S), \mathsf{vol}(\bar{S})\} \geq b \cdot \mathsf{vol}(V)$. This is an NP-hard problem and a central object of study for the development of approximation algorithms. On the theoretical side, BS has far reaching connections to spectral graph theory, the study of random walks and metric embeddings. In practice, algorithms for BS play a crucial role in the design of recursive algorithms [32], clustering [16] and scientific computation [29].

Spectral methods are an important set of techniques in the design of graph-partitioning algorithms and are based on the study of the behavior of random walks over the instance graph. Spectral algorithms tend to be conceptually appealing, because of the intuition based on the underlying diffusion process, and easy to implement, as many of the primitives required, such as eigenvector computation, already appear in highly-optimized software packages. The most important spectral algorithm for graph partitioning is the Laplacian Eigenvector (LE) algorithm of Alon and Milman [1], which, given a graph of conductance at most $\gamma$, outputs a cut of conductance at most $O(\sqrt{\gamma})$, an approximation guarantee that is asymptotically optimal for spectral

algorithms. A consequence of the seminal work of Spielman and Teng [34] is that the LE algorithm can run in time $\tilde{O}(m)$ using the Spielman-Teng solver. Hence, LE is an asymptotically optimal spectral algorithm for the minimum-conductance problem, both for running time (up to poly-log factors) and approximation quality. In this paper, we present a simple random-walk-based algorithm that is the first such asymptotically optimal spectral algorithm for BS. Our algorithm can be seen as an analogue to the LE algorithm for the balanced version of the minimum-conductance problem and settles the question of designing spectral algorithms for BS. The following is our main theorem on graph partitioning.

THEOREM 1. (Spectral Algorithm for Balanced Separator). *We give a randomized algorithm* BALSEP *such that, given an unweighted graph $G = (V, E)$, a balance parameter $b \in (0, 1/2]$, $b = \Omega(1)$ and a conductance value $\gamma \in (0, 1)$,* $\text{BALSEP}(G, b, \gamma)$ *either outputs an $\Omega(b)$-balanced cut $S \subset V$ such that $\phi(S) \leq O(\sqrt{\gamma})$, or outputs a certificate that no $b$-balanced cut of conductance $\gamma$ exists.* BALSEP *runs in time $O(m \operatorname{poly}(\log n))$.*

The algorithm for Theorem 1 relies on our ability to compute the product of the matrix-exponential of a matrix and an arbitrary vector in time essentially proportional to the sparsity of the matrix. Our contribution to the problem of computing the matrix-exponential-vector product appear in Section 1.2. The algorithm required for Theorem 1 runs in time $\tilde{O}(m)$ and, notably, makes use of the Spielman-Teng solver in a non-trivial way. We also prove an alternative result on how to perform this matrix-exponential computation, which relies just on matrix-vector products. This result, when combined with our BS algorithm based on random walks, yields the following theorem identical to Theorem 1 except that the running time now increases to $\tilde{O}(m/\sqrt{\gamma})$.

THEOREM 2. (Simple Spectral Algorithm for Balanced Separator). *We give a randomized algorithm that, given an unweighted graph $G = (V, E)$, a balance parameter $b \in (0, 1/2]$, $b = \Omega(1)$ and a conductance value $\gamma \in (0, 1)$, runs in time $\tilde{O}(m/\sqrt{\gamma})$, that either outputs an $\Omega(b)$-balanced cut $S \subset V$ such that $\phi(S) \leq O(\sqrt{\gamma})$ or outputs a certificate that no $b$-balanced cut of conductance $\gamma$ exists.*

However, this latter algorithm not only turns out to be almost as simple and practical as the LE algorithm, but it also improves in the approximation factor upon the result of Andersen and Peres [4] who obtain the same running time using Evolving-Sets-based random walk.

### 1.1.1 Comparison to Previous Work on Balanced Separator

The best known approximation for BS is $O(\sqrt{\log n})$ achieved by the seminal work of Arora, Rao and Vazirani [6] that combines semidefinite programming (SDP) and flow ideas. A rich line of research has centered on reducing the running time of this algorithm using SDP and flow ideas [17, 5, 21]. This effort culminated in Sherman's work [30], which brings down the required running time to $O(n^{\varepsilon})$ $s$-$t$ maximum-flow computations.[1] However, these algorithms are based on advanced theoretical ideas that are not easy to implement or even capture in a principled heuristic. Moreover, they fail to achieve a nearly-linear[2] running time, which is crucial in many of today's applications that involve very large graphs. To address these issues, researchers have focused on the design of simple, nearly-linear-time algorithms for BS based on spectral techniques. The simplest spectral algorithm for BS is the Recursive Laplacian Eigenvector (RLE) algorithm (see, for example, [16]). This algorithm iteratively uses LE to remove low-conductance unbalanced cuts from $G$, until a balanced cut or an induced $\gamma$-expander is found. The running time of the RLE algorithm is quadratic in the worst case, as $\Omega(n)$ unbalanced cuts may be found, each requiring a global computation of the eigenvector. Spielman and Teng [33] were the first to design nearly-linear-time algorithms outputting an $\Omega(b)$-balanced cut of conductance $O(\sqrt{\gamma \operatorname{polylog} n})$, if a $b$-balanced cut of conductance less than $\gamma$ exists. Their algorithmic approach is based on local random walks, which are used to remove unbalanced cuts in time proportional to the size of the cut removed, hence avoiding the quadratic dependence of RLE. Using similar ideas, Andersen, Chung and Lang [2], and Andersen and Peres [4] improved the approximation guarantee to $O(\sqrt{\gamma \log n})$ and the running time to $\tilde{O}(m/\sqrt{\gamma})$. More recently, Orecchia and Vishnoi (OV) [22] employed an SDP formulation of the problem, together with the Matrix Multiplicative Weight Update (MMWU) of [5] and a new SDP rounding, to obtain an output conductance of $O(\sqrt{\gamma})$ with running time $\tilde{O}(m/\gamma^2)$, effectively removing unbalanced cuts in $O(\log n/\gamma)$ iterations. In Section 4.5, we give a more detailed comparison with OV and discussion of our novel width-reduction techniques from an optimization point of view. Finally, our algorithm should also be compared to the remarkable results of Madry [19] for BS, which build up on Räcke's work [25] to achieve a trade-off between running time and approximation. For every integer $k \geq 1$, he achieves roughly $O((\log n)^k)$ approximation in time $\tilde{O}(m + 2^k \cdot n^{1+2^{-k}})$. Calculations show that for $\gamma \geq 2^{-(\log \log n)^2}$, our algorithm achieves strictly better running time and approximation than Madry's for sparse graphs. More importantly, we believe that our algorithm is significantly simpler, especially in its second form mentioned above, and likely to find applications in practical settings.

## 1.2 The Matrix Exponential, the Lanczos Method, and Approximations to $e^{-x}$

We first state a few definitions used in this section. We will work with $n \times n$, symmetric and positive semi-definite (PSD) matrices over $\mathbb{R}$. For a matrix $M$, abusing notation, we denote its exponential by $\exp(-M)$, or by $e^{-M}$, and define it as $\sum_{i \geq 0} \frac{(-1)^i}{i!} M^i$. $M$ is said to be *Symmetric and Diagonally Dominant* (SDD) if, $M_{ij} = M_{ji}$, for all $i, j$ and $M_{ii} \geq \sum_{j \neq i} |M_{ij}|$, for all $i$. Let $m_M$ denote the number of non-zero entries in $M$ and let $t_M$ denote the time required to multiply the matrix $M$ with a given vector $v$. In general, $t_M$ depends on how $M$ is given as an input and can be $\Theta(n^2)$. However, it is possible to exploit the special structure of $M$ if given as an input appropriately: It is possible to just multiply the non-zero entries of $M$, giving $t_M = O(m_M)$. Also,

[1] Even though the results of [6] and [30] are stated for the Sparsest Cut problem, the same techniques apply to the conductance problem, e.g. by modifying the underlying flow problems. See for example [3].

[2] Following the convention of [35], we denote by nearly-linear a running time of $\tilde{O}(m/\operatorname{poly}(\gamma))$.

if $M$ is a rank one matrix $ww^\top$, where $w$ is known, we can multiply with $M$ in $O(n)$ time. We move on to our results.

At the core of our algorithm for BS, and more generally of most MMWU based algorithms, lies an algorithm to quickly compute $\exp(-A)v$ for a PSD matrix $A$ and a unit vector $v$. It is sufficient to compute an approximation $u$, to $\exp(-A)v$, in time which is as close as possible to $t_A$. It can be shown that using about $\|A\|$ terms in the Taylor series expansion of $\exp(-A)$, one can find a vector $u$ that approximates $\exp(-A)v$. Hence, this method runs in time roughly $O(t_A \cdot \|A\|)$. In our application, and certain others [5, 15, 14, 13], this dependence on the norm is prohibitively large. The following remarkable result was cited in Kale [15].

**Hypothesis.**

*Let $A \succeq 0$ and $\varepsilon > 0$. There is an algorithm that requires $O\left(\log^2 1/\varepsilon\right)$ iterations to find a vector $u$ such that $\|\exp(-A)v - u\| \leq \|\exp(-A)\|\,\varepsilon$, for any unit vector $v$. The time for every iteration is $O(t_A)$.*

This hypothesis would suffice to prove Theorem 1. But, to the best of our knowledge, there is no known proof of this result. In fact, the source of this unproved hypothesis can be traced to a paper of Eshof and Hochbruck (EH) [11]. EH suggest that one may use the Lanczos method (described later), and combine it with a rational approximation for $e^{-x}$ due to Saff, Schonhage and Varga [28], to reduce the computation of $\exp(-A)v$ to a number of $(I+\alpha A)^{-1}v$ computations for some $\alpha > 0$. Note that this is insufficient to prove the hypothesis above as there is no known way to compute $(I+\alpha A)^{-1}v$ in time $O(t_A)$. They note this and propose the use of iterative methods to do this computation. They also point out that this will only result in an approximate solution to $(I + \alpha A)^{-1}v$ and make no attempt to analyze the running time or the error of their method when the inverse computation is approximate. We believe that we are quite distant from proving the hypothesis for all PSD matrices and a significant part of this paper is devoted to a proof of the above hypothesis for a class of PSD matrices that turns out to be sufficient for the BS application. For the norm-independent, fast-approximate inverse computation, we appeal to the result of Spielman and Teng [34] (also see improvements by Koutis, Miller and Peng [18]). The theorem we prove is the following.

THEOREM 3. (SDD Matrix Exponential Computation). *There is an algorithm that, given an $n \times n$ SDD matrix $A$, a vector $v$ and a parameter $0 < \delta \leq 1$, can compute a vector $u$ such that $\|\exp(-A)v - u\| \leq \delta \|v\|$ in time $\tilde{O}((m_A + n)\log(2+\|A\|))$. The tilde hides* $\mathrm{poly}(\log n)$ *and* $\mathrm{poly}(\log 1/\delta)$ *factors.* [3]

First, we note that for our application, the dependence of the running time on the $\log(2+\|A\|)$ turns out to just contribute an extra $\log n$ factor. Also, for our application $\delta = 1/\mathrm{poly}(n)$. Secondly, for our BS application, the matrix we need to invert is not SDD or sparse. Fortunately, we can combine Spielman-Teng solver with the Sherman-Morrison formula to invert our matrices; see Theorem 7. A significant effort goes into analyzing the effect of the error introduced due to approximate matrix inversion. This error can cascade due to the iterative nature of our algorithm that proves this theorem.

Towards proving the hypothesis above, when the only guarantee we know on the matrix is that it is symmetric and PSD, we prove the following theorem, which is the best known algorithm to compute $\exp(-A)v$ for an arbitrary symmetric PSD matrix $A$, when $\|A\| = \omega(\mathrm{poly}(\log n))$.

THEOREM 4. (PSD Matrix Exponential Computation). *There is an algorithm that, given an $n \times n$ symmetric PSD matrix $A$, a vector $v$ and a parameter $0 < \delta \leq 1$, computes a vector $u$ in time $\tilde{O}\left((t_A + n)\sqrt{1+\|A\|}\log(2+\|A\|)\right)$, such that $\|\exp(-A)v - u\| \leq \delta \|v\|$. Here the tilde hides* $\mathrm{poly}(\log n)$ *and* $\mathrm{poly}(\log 1/\delta)$ *factors.*

In the symmetric PSD setting we also prove the following theorem which, for our application, gives a result comparable to Theorem 4.

THEOREM 5. (Simple PSD Matrix Exponential Computation). *There is an algorithm that, given an $n \times n$ symmetric PSD matrix $A$, a vector $v$ and a parameter $\delta \leq 1$, computes a vector $u$ such that $\|\exp(-A)v - u\| \leq \delta \|v\|$, in time $O((t_A + n)\cdot k + k^2)$, where $k \stackrel{def}{=} \tilde{O}(\sqrt{1+\|A\|})$. Here the tilde hides* $\mathrm{poly}(\log 1/\delta)$ *factors.*

As noted before, $t_A$ can be significantly smaller than $m_A$. Moreover, it only uses multiplication of a vector with the matrix $A$ as a primitive and does not require matrix inversion. Consequently, it does not need tools like the SDD solver or conjugate gradient, thus obviating the error analysis required for the previous algorithms. Furthermore, this algorithm is very simple and when combined with our random walk-based BALSEP algorithm, results in a very simple and practical $O(\sqrt{\gamma})$ approximation algorithm for BS that runs in time $\tilde{O}(m/\sqrt{\gamma})$. Finally, we note that Theorem 5 is also implied by an independent earlier work of Hochbruck and Lubich (Theorem 2 in [12]) of which we were unaware when we first wrote this paper.

Theorem 5 relies on the Lanczos method which can be used to convert guarantees about polynomial approximation from scalars to matrices. In particular, it uses the following structural result (the upper bound) on the best degree $k$ polynomial $\delta$-uniformly approximating $e^{-x}$ in an interval $[a, b]$. We also prove a lower bound which establishes that the degree cannot be improved beyond lower order terms. This suggests that improving on the $\tilde{O}(m/\sqrt{\gamma})$ running time in Theorem 5 requires more advanced techniques.

THEOREM 6. (Uniform Approximation to $e^{-x}$).

- **Upper Bound.** *For every $0 \leq a < b$, and $0 < \delta \leq 1$, there exists a polynomial $p$ that satisfies,*

$$\sup_{x\in[a,b]} |e^{-x} - p(x)| \leq \delta \cdot e^{-a},$$

*and has degree*

$$O\left(\sqrt{\max\{\log 1/\delta, (b-a)\}} \cdot (\log 1/\delta)^{3/2} \cdot \log\log 1/\delta\right).$$

- **Lower Bound.** *For every $0 \leq a < b$ such that $a + \log_e 4 \leq b$, and $\delta \in (0, 1/8]$, any polynomial $p(x)$ that approximates $e^{-x}$ uniformly over the interval $[a, b]$ up to an error of $\delta \cdot e^{-a}$, must have degree at least $\frac{1}{2} \cdot \sqrt{b-a}$.*

[3] Note that $\|\exp(-A)\| = e^{-\lambda_n(A)}$, where $\lambda_n(A)$ is the smallest eigenvalue of $A$. If $\|\exp(-A)\|$ is a very small factor, the error guarantee of Theorem 3 could be significantly worse than the guarantee from the hypothesis above. However, all matrices $A$ that we wish to exponentiate, have $\lambda_n(A) = 0$, in which case, the two error guarantees are equivalent.

## 2. ORGANIZATION OF THE MAIN BODY OF THE PAPER

In Section 3 we present an overview of our results. In the following three sections, we give formal descriptions of our algorithms and provide more details. Due to space restrictions, most of the proofs are deferred to the full version [20]. Section 4 describes our algorithm for the Balanced Separator problem and related theorems (Theorems 1 and 2). Section 5 contains our results on computing the matrix exponential; in particular proof details for Theorems 3, 5 and 7. Section 6 contains our structural results on approximating $e^{-x}$ and the proof details for Theorem 6. Finally, in Section 7 we discuss the open problems arising from our work.

## 3. OVERVIEW OF OUR RESULTS

### 3.1 Our Spectral Algorithm for Balanced Separator

In this section, we provide a high level idea of the the algorithm BalSep mentioned in Theorem 1. As pointed out in the introduction, our algorithm, BalSep, when combined with the matrix-exponential-vector algorithm in Theorem 5 results in a very simple and practical algorithm for BS. A more detailed and technical description of this part appears in Section 4.

#### 3.1.1 *The* RLE *Algorithm*

Before we introduce our algorithm, it is useful to review the RLE algorithm. Recall that given $G, \gamma$ and $b$, the goal of the BS problem is to either certify that every $b$-balanced cut in $G$ has conductance at least $\gamma$, or produce a $\Omega(b)$ balance cut in $G$ of conductance $O(\sqrt{\gamma})$. RLE achieves this task by applying LE iteratively to remove unbalanced cuts of conductance $O(\sqrt{\gamma})$ from $G$. The iterations stop and the algorithm outputs a cut, when it either finds a $(b/2)$-balanced cut of conductance $O(\sqrt{\gamma})$ or the union of all unbalanced cuts found so far is $(b/2)$-balanced. Otherwise, the algorithm terminates when the spectral gap (of the normalized Laplacian) of the residual graph is at least $2\gamma$. In the latter case, any $b$-balanced cut must have at least half of its volume lie within the final residual graph, and hence, have conductance at least $\gamma$ in the original graph. Unfortunately, this algorithm may require $\Omega(n)$ iterations in the worst case. For instance, this is true if the graph $G$ consists of $\Omega(n)$ components loosely connected to an expander-like core through cuts of low conductance. This example highlights the weakness of the RLE approach: the second eigenvector of the Laplacian may only be correlated with one low-conductance cut and fail to capture at all even cuts of slightly larger conductance. This limitation makes it impossible for RLE to make significant progress at any iteration.

#### 3.1.2 *High-Level Idea of* BalSep

As in RLE, the algorithm BalSep iteratively removes unbalanced cuts of conductance $O(\sqrt{\gamma})$ at every iteration until an $\Omega(b)$-balanced cut of conductance $O(\sqrt{\gamma})$ is found or a certificate is produced that no $b$-balanced cut of conductance less than $\gamma$ exists. However, BalSep overcomes the limitation of RLE by adopting a different cut-finding procedure and a different way of removing unbalanced cuts of low conductance. The goal of both these two modifications is to make it easier to track the progress of the algorithm from one iteration to the next in terms of a simple potential function. In turn, this allows us to show that BalSep removes unbalanced cuts in only $O(\log n)$ iterations.

First, rather than working with the vertex embedding given by the eigenvector, at iteration $t$ we will consider the multi-dimensional vector embedding represented by the transition probability matrix $P^{(t)}$ of a certain continuous-time random walk over the graph. The embedding given by $P^{(t)}$ can be interpreted as a probabilistic version of the second eigenvector of the Laplacian, as it consists of a distribution over eigenvectors, with eigenvectors of low eigenvalue assigned more weight. Hence, the embedding captures not only the cut corresponding to the second eigenvector, but also cuts associated with other eigenvectors of eigenvalue less than $\gamma$ or close to $\gamma$. This first modification of RLE will enable our algorithm to find many different low-conductance unbalanced cuts at once.

Secondly, if at iteration $t$ an unbalanced cut $S^{(t)}$ of conductance less than $O(\sqrt{\gamma})$ is found, BalSep departs from RLE by only performing a *soft* removal of $S^{(t)}$, rather than eliminating $S^{(t)}$ and all its adjacent edges from the graph. This soft removal is achieved by increasing the transition rates of $P^{(t)}$ across $(S^{(t)}, \overline{S}^{(t)})$. This increase ensures that at the next iteration the new random walk $P^{(t+1)}$ will mix more quickly across $(S^{(t)}, \overline{S}^{(t)})$ and the same cut will be unlikely likely to be detected again.

The random walks and soft cut-removals, used in BalSep in place of the eigenvectors and hard cut-removals of RLE, share a common goal: to make $P^{(t+1)}$ a more stable object with respect to $P^{(t)}$. In particular, this will allow us to precisely quantify how the mixing of $P^{(t)}$ changes from one iteration to the next. In fact, our potential function $\Psi^{(t)}$ at time $t$ will be a natural measure of the mixing of $P^{(t)}$. We will show that every time an unbalanced cut is found the mixing $\Psi^{(t)}$ improves by a constant multiplicative factor. This potential reduction will then yield that $O(\log n)$ iterations suffice to either find a low-conductance balanced cut or output a certificate that no such cut exists.

#### 3.1.3 *Sketch of the Algorithm and Its Analysis*

In our algorithm BalSep, the random walk process described by $P^{(t)}$ is a type of continuous-time random walk, which we refer to as an *Accelerated Heat Kernel Walk* (AHK) and described more completely in Section 4.2. AHK random walks are a variation of the heat-kernel random walk [9], i.e. the continuous-time natural random walk over $G$, in which we now allow some vertices to have increased transition rates to all vertices in $V$. This has the effect of accelerating the converge to stationary from such vertices. We use AHK random walks to accelerate the convergence to the stationary distribution of random walks starting at vertices inside the low-conductance unbalanced cuts $\{S^{(t)}\}$ detected by BalSep. As discussed above, accelerating the mixing across $(S^{(t)}, \overline{S^t})$ plays the role of a soft removal of $S^{(t)}$.

Hence, at iteration $t$, $P^{(t)}$ can be thought of as a heat-kernel random walk, modified by accelerating the convergence over cuts $S^{(1)}, \ldots, S^{(t-1)}$ and run for $\tau \stackrel{\text{def}}{=} O(\log n/\gamma)$ time. This choice of $\tau$ ensures that the walk must mix across all cuts of conductance much larger than $\gamma$, hence emphasizing cuts of the desired conductance.

*The Algorithm.*

At iteration $t$, the algorithm considers the vector embedding given by the columns of $P^{(t)}$, translated so that the stationary distribution, i.e. the uniform distribution, takes the place of the origin. If all vectors in this embedding are short, $P^{(t)}$ has mixed well from all vertices in $V$. Using our potential function, we will be able to show that in this case we have a certificate that no $b$-balanced cut has conductance less than $\gamma$ in $G$.

Conversely, if a vector[4] $v_i \stackrel{\text{def}}{=} P^{(t)}e_i$ has large length, the random walk started at $i$ must be far from mixing, which allows us to conclude that $i$ must be contained in some cut of conductance close to $\gamma$. In this case, BalSep uses SDP-techniques from OV [22] to find a cut $S^{(t)}$ of conductance at most $O(\sqrt{\gamma})$. As in RLE, if $S^{(t)}$ is $\Omega(b)$-balanced or if the union $\cup_{i=1}^{t} S^{(i)}$ is $\Omega(b)$-balanced, BalSep can output an $\Omega(b)$-balanced cut of the required conductance. Otherwise, BalSep constructs $P^{(t+1)}$ by accelerating the convergence of $S^{(t)}$ and proceeds to the next iteration.

*Potential Analysis.*

For our analysis, the potential $\Psi^{(t)}$ is chosen to be the sum over all vertices $i$ of the $\ell_2^2$-distance between $P^{(t)}e_i$ and the stationary distribution over $G$. This potential function measures how well $P^{(t)}$ mixes, so that high values of $\Psi^{(t)}$ demonstrate the existence of low-conductance cuts and low values of $\Psi^{(t)}$ can be used to certify that no $b$-balanced cut of conductance $\gamma$ exists.

As long as $\Psi^{(t)}$ is sufficiently large, the random walks started at some vertices are not mixing, and we can use the cut-finding procedure of OV to find an unbalanced cut $S^{(t)}$ of conductance at most $O(\sqrt{\gamma})$. Using an argument from OV, we are also able to show that most of the distance from stationary $\Psi^{(t)}$ of $P^{(t)}$ can in fact be attributed to vertices in $S^{(t)}$. In words, we can think of $S^{(t)}$ as the main reason why $P^{(t)}$ is not mixing. Moreover, we can show that modifying $P^{(t)}$ into $P^{(t+1)}$ has the effect of removing from $\Psi^{(t)}$ a large fraction of the potential due to $S^{(t)}$. Hence, we can show the total potential $\Psi^{(t+1)}$ of $P^{(t+1)}$ is at most just a constant fraction of $\Psi^{(t)}$.

The potential reduction at every iteration allows us to argue that after $T = O(\log n)$ iterations, we must either find an $\Omega(b)$-balanced cut of conductance $O(\sqrt{\gamma})$ or a transition matrix $P^{(t)}$ which has small distance from the stationary distribution. We claim that, in the latter case, we can turn $P^{(t)}$ into a certificate that $G$ contains no $b$-balanced cuts of conductance less than $\gamma$. To show this, we suppose that such a low-conductance balanced cut $T$ existed. Then, the heat-kernel random walk, run for time $\tau$, would not mix across $T$. Hence, the only way $P^{(t)}$ could mix across $T$ is by having the increased rates across the cuts $\{S^{(i)}\}$ help the random walk mix over $(T, \overline{T})$. However, as $T$ is $b$-balanced, we can show that many vertices must have their convergence accelerated for $P^{(t)}$ to mix across $T$. This yields a contradiction as the only vertices accelerated in $P^{(t)}$ are those $\cup_{i=1}^{t-1} S^{(i)}$, and $\cup_{i=1}^{t-1} S^{(i)}$ must have small volume, or BalSep would have returned it as an $\Omega(b)$-balanced cut of the required conductance.

[4]Here $e_i$ denotes the $i$th standard unit vector in $\mathbb{R}^n$.

*Running time.*

To ensure that each iteration requires only $\tilde{O}(m)$ time, we use the Johnson-Lindenstrauss Lemma to compute a $O(\log n)$-dimensional approximation to the embedding $P^{(t)}$. To compute this approximation, we perform $O(\log n)$ matrix-vector multiplications of the form $P^{(t)}u$ where $P^{(t)}$ is a matrix exponential and $u$ is a unit vector. To complete these computations in time $\tilde{O}(m)$, we rely on our results on the fast computation of matrix-exponentials.

## 3.2 Our Algorithms for Computing an Approximation to $\exp(-A)v$

In this section, we give an overview of the algorithms and proofs for Theorems 3 and 5, deferring formal proof sketches to Section 5. The algorithm for Theorem 4 is very similar to the one for Theorem 3, and the details are given in Section 5.2.2. A few quick definitions: A matrix $M$ is called *Upper Hessenberg* if, $(M)_{ij} = 0$ for $i > j + 1$. $M$ is called *tridiagonal* if $M_{ij} = 0$ for $i > j + 1$ and for $j > i + 1$. Let $\lambda_1(M)$ and $\lambda_n(M)$ denote the largest and smallest eigenvalues of $M$, respectively.

As mentioned in the introduction, the matrices that we need to exponentiate for the BS algorithm are neither sparse nor SDD. Thus, Theorem 3 is insufficient for our application. Fortunately, the following theorem suffices and its proof is not very different from that of Theorem 3, which is explained below. Details of the proof of the following theorem appears in Section 5.2.3.

Theorem 7. (Matrix Exponential Computation Beyond SDD). *There is an algorithm that, given a vector $v$, a parameter $0 < \delta \leq 1$ and an $n \times n$ symmetric matrix $A = \Pi HMH\Pi$ where $M$ is SDD, $H$ is a diagonal matrix with strictly positive entries and $\Pi$ is a rank $(n-1)$ projection matrix, $\Pi \stackrel{def}{=} I - ww^\top$ ($w$ is explicitly known and $\|w\| = 1$), computes a vector $u$ such that $\|\exp(-A)v - u\| \leq \delta \|v\|$ in time $\tilde{O}((m_M + n)\log(2 + \|HMH\|))$. The tilde hides* $\text{poly}(\log n)$ *and* $\text{poly}(\log 1/\delta)$ *factors.*

As we will show later (see Section 4), our algorithm for BS requires us to compute $\exp(-A)v$ for a matrix $A$ of the form $D^{-1/2}(L + \sum_i \beta_i L(\mathsf{Star}_i))D^{-1/2}$, where $D$ is the diagonal matrix with the $(i,i)$ entry being the degree of vertex $i$, $L$ is the Laplacian of the input graph $G$, $\beta_i \geq 0$ and $L(\mathsf{Star}_i)$ is the Laplacian of the star rooted at vertex $i$. If we let $\mathbf{1}$ denote the all 1's vector, and $\Pi \stackrel{\text{def}}{=} I - 1/2m \cdot (D^{1/2}\mathbf{1})(D^{1/2}\mathbf{1})^\top$, the projection onto the space orthogonal to $1/\sqrt{2m} \cdot D^{1/2}\mathbf{1}$, then the way we define $L(\mathsf{Star}_i)$, it will turn out that $\forall i$, $D^{-1/2}L(\mathsf{Star}_i)D^{-1/2} = \Pi(d_i/2m \cdot I + e_i e_i^\top)\Pi$. Since $D^{1/2}\mathbf{1}$ is an eigenvector of $D^{-1/2}LD^{-1/2}$ with eigenvalue 0, we have, $\Pi D^{-1/2}LD^{-1/2}\Pi = D^{-1/2}LD^{-1/2}$. Thus,

$$A = \Pi D^{-1/2}LD^{-1/2}\Pi + \sum_i \beta_i \Pi(d_i/2m \cdot I + e_i e_i^\top)\Pi$$

$$= \Pi D^{-1/2}(L + \sum_i \beta_i d_i/2m \cdot D + \sum_i \beta_i d_i \cdot e_i e_i^\top)D^{-1/2}\Pi.$$

This is of the form $\Pi HMH\Pi$, where $H \stackrel{\text{def}}{=} D^{-1/2}$ is diagonal and $M$ is SDD. The proof of the above theorem uses the Sherman-Morrison formula to extend the SDD solver to fit our requirement. Moreover, to obtain a version of Theorem 5 for such matrices, we do not have to do much additional work since multiplication by $H$ and $\Pi$ take $O(n)$ steps and hence,

$t_A$ is still $O(m_M + n)$. The details appear in Section 5.2.3. Finally, note that in our application, $\|HMH\|$ is poly$(n)$.

We now give an overview of the proofs of Theorem 3 and Theorem 5. First, we explain a general method known as the Lanczos method, which is pervasive in numerical linear algebra. We then show how suitable adaptations of this can be combined with (old and new) structural results in approximation theory to obtain our results.

### 3.2.1 Lanczos Method

Given an $n \times n$ symmetric PSD matrix $B$ and a function $f : \mathbb{R} \mapsto \mathbb{R}$, we can define $f(B)$ as follows: Let $u_1, \ldots, u_n$ form an eigenbasis for $B$ with eigenvalues $\lambda_1, \ldots, \lambda_n$. Thus, $B = \sum_i \lambda_i u_i u_i^\top$. Define $f(B) \stackrel{\text{def}}{=} \sum_i f(\lambda_i) u_i u_i^\top$. We will reduce both our algorithms to computing $f(B)v$ for a given vector $v$, albeit with different $f$'s and $B$'s. We point out the $f$'s and $B$'s required for Theorems 3 and 5 in Sections 3.2.2 and 3.2.3 respectively.

Since exact computation of $f(B)$ usually requires diagonalization of $B$, which could take as much as $O(n^3)$ time (see [23]), we seek an approximation to $f(B)v$. The Lanczos method allows us to do exactly that: It looks for an approximation to $f(B)v$ of the form $p(B)v$, where $p$ is a polynomial of small degree, say $k$. Before we describe how, we note that it computes this approximation in roughly $O((t_B + n)k)$ time, plus the time it takes to compute $f(\cdot)$ on a $(k+1) \times (k+1)$ tridiagonal matrix, which can often be upper bounded by $O(k^2)$ (see [23]). Hence, the time is reduced to $O((t_B+n)k+k^2)$. What one has lost in this process is accuracy: The candidate vector $u$ output by the Lanczos method, is now only an approximation to $f(B)v$. The error in the approximation, $\|f(B)v - u\|$, can be upper bounded by the *uniform error* of the best degree $k$ polynomial approximating $f$ in the interval $[\lambda_n(B), \lambda_1(B)]$. Roughly,

$$\|f(B)v - u\| \lessapprox \min_{p_k \in \Sigma_k} \sup_{\lambda \in [\lambda_n(B), \lambda_1(B)]} |f(\lambda) - p_k(\lambda)|.$$

Here $\Sigma_k$ is the collection of all real polynomials of degree at most $k$. Surprisingly, one does not need to know the best polynomial and proving *existence* of good polynomials is sufficient. By increasing $k$, one can reduce this error. Indeed, for $k = n$, there is no error. Thus, the task is reduced to proving existence of low degree polynomials that approximate $f$ within the error limits required for the applications.

*Computing the Best Polynomial Approximation.*

Now, we describe in detail, the Lanczos method and how it achieves the error guarantee claimed above. Notice that for any polynomial $p$ of degree at most $k$, the vector $p(B)v$ lies in $\mathcal{K} \stackrel{\text{def}}{=} \mathsf{Span}\{v, Bv, \ldots, B^k v\}$ – called the *Krylov subspace*. The Lanczos method iteratively creates an orthonormal basis $\{v_i\}_{i=0}^k$ for $\mathcal{K}$, such that for all $i \leq k$, $\mathsf{Span}\{v_0, \ldots, v_i\} = \mathsf{Span}\{v, \ldots, B^i v\}$. Let $V_k$ be the $n \times (k+1)$ matrix with $\{v_i\}_{i=0}^k$ as its columns. Thus, $V_k V_k^\top$ denotes the projection onto the Krylov subspace. Let $T_k$ be the $(k+1) \times (k+1)$ matrix $T_k \stackrel{\text{def}}{=} V_k^\top B V_k$, expressing $B$ as an operator restricted to $\mathcal{K}$ in the basis $\{v_i\}_{i=0}^k$. Note that this is not just a change of basis, since vectors in $\mathcal{K}$ can be mapped by $B$ to vectors outside $\mathcal{K}$. From this discussion, it can be shown that $p(B)v = V_k p(T_k) V_k^\top v$, for any polynomial $p$ of degree at most $k$.

Hence, a natural approximation for $f(B)v$ is $V_k f(T_k) V_k^\top v$. Writing $r_k(x) \stackrel{\text{def}}{=} f(x) - p_k(x)$, where $p_k$ is any degree $k$ approximation to $f(x)$, the error in the approximation is $f(B)v - V_k f(T_k) V_k^\top v = r_k(B)v - V_k r_k(T_k) V_k^\top v$, *for any choice of* $p_k$. Hence, the norm of the error vector is at most $(\|r_k(B)\| + \|r_k(T_k)\|)\|v\|$, which is bounded by the value of $r_k$ on the eigenvalues of $B$ and $T_k$. Since the eigenvalues of $T_k$ are bounded by the eigenvalues of $B$, the norm of the error is bounded by $2\|v\| \cdot \max_{\lambda \in [\lambda_n(B), \lambda_1(B)]} |f(\lambda) - p_k(\lambda)|$. Minimizing over $p_k$ gives the error bound claimed above. Note that we do not explicitly need the approximating polynomial. It suffices to prove that there exists a degree $k$ polynomial that uniformly approximates $f$ well on an interval containing the spectrum of $B$ and $T_k$.

Finally, observe that $(T_k)_{ij} = v_i^\top B v_j$. If we construct the basis iteratively as above, we have $Bv_j \in \mathsf{Span}\{v_0, \ldots, v_{j+1}\}$ by construction, and if $i > j + 1$, $v_i$ is orthogonal to this subspace and hence $v_i^\top (Bv_j) = 0$. Thus, $T_k$ is Upper Hessenberg. Moreover, if $B$ is symmetric, $v_j^\top (Bv_i) = v_i^\top (Bv_j)$, and hence $T_k$ is symmetric and tridiagonal. This means that while constructing the basis, at step $i + 1$, it needs to orthonormalize $Bv_i$ only w.r.t. $v_{i-1}$ and $v_i$. Thus the total time required is $O((t_B + n)k)$, plus the time required for the computation of $f(T_k)$, which can typically be bounded by $O(k^2)$ for a tridiagonal matrix (using [23]). This completes an overview of the Lanczos method. The Lanczos procedure, described in Figure 2 in Section 5, implements the Lanczos method. We next describe how we apply the Lanczos method to obtain our two algorithms.

### 3.2.2 Approximating $\exp(-A)v$ Using a Rational Approximation to $e^{-x}$

*Our Algorithm.*

The starting point of the algorithm underlying Theorem 3 is a rather surprising result by Saff, Schönhage and Varga (SSV) [28], which says that for any positive integer $k$, there exists a degree $k$ polynomial $p_k^\star$ such that, $p_k^\star((1+x/k)^{-1})$ approximates $e^{-x}$ up to an error of $O(k \cdot 2^{-k})$ over the interval $[0, \infty)$ (Theorem 12). Then, to approximate $\exp(-A)v$, one could apply the Lanczos method with $B \stackrel{\text{def}}{=} (I + A/k)^{-1}$ and $f(x) \stackrel{\text{def}}{=} e^{k(1-1/x)}$. Essentially, this was the method suggested by Eshof and Hochbruck (EH) [11]. The strong approximation guarantee of the SSV result, along with the guarantee of the Lanczos method from the previous section, would imply that the order of the Krylov subspace for $B$ required would be roughly $\log 1/\delta$, and hence, independent of $\|A\|$. The running time is then dominated by the computation $Bv = (I + A/k)^{-1} v$.

EH note that the computation of exact matrix inverse is a costly operation ($O(n^3)$ time in general), and all known faster methods for inverse computation incur some error. They suggest using the Lanczos method with faster iterative methods, e.g. Conjugate Gradient, for approximating the inverse (or rather the product of the inverse with a given vector) as a *heuristic*. They make no attempt to give a theoretical justification of why approximate computation suffices. Also note that, even if the computation was error-free, a method such as Conjugate Gradient will have running time which varies with $\sqrt{\lambda_1(A)/\lambda_n(A)}$ in general. Thus, the EH method falls substantially short of resolving the hypothesis mentioned in the introduction.

To be able to prove Theorem 3 using the SSV guarantee, we have to modify the Lanczos method in several ways, and hence, deviate from the method suggested by EH: 1)

EH construct $T_k$ as a tridiagonal matrix as in the Lanczos method, but since the computation is no longer exact, the basis $\{v_i\}_{i=0}^k$ is no longer guaranteed to be orthonormal. As a result, the proofs of the Lanczos method break down. Our algorithm, instead, builds an orthonormal basis, which means that $T_k$ becomes an Upper Hessenberg matrix instead of tridiagonal, and we need to compute $k^2$ dot products in order to compute $T_k$. 2) With $T_k$ being asymmetric, several nice spectral properties are lost, *e.g.* real eigenvalues and an orthogonal set of eigenvectors. We overcome this issue by symmetrizing $T_k$ to construct $\widehat{T}_k = \frac{T_k + T_k^\top}{2}$, and computing our approximation with $\widehat{T}_k$. Since $\widehat{T}_k$ is symmetric, we can bound the quality of a polynomial approximation applied to $\widehat{T}_k$ by the behavior of the polynomial on the eigenvalues of $\widehat{T}_k$. 3) Our analysis is based on the SSV approximation result, which is better than the variant proved and used by EH. Moreover, for their *shifting* technique, which is the source of the $\|\exp(-A)\|$ factor in the hypothesis, the given proof in EH is incorrect and it is not clear if the given bound could be achieved even under exact computation[5]. 4) Most importantly, since $A$ is SDD, we are able to employ the Spielman-Teng solver (Theorem 14) to approximate $(I + A/k)^{-1}v$. Our procedure, called ExpRational, has been described in Figure 3 in Section 5.

*Error Analysis.*

To complete the proof of Theorem 3, we need to analyze the role of the error that creeps in due to approximate matrix inversion. The problem is that this error, generated in each iteration of the Krylov basis computation, propagates to the later steps. Thus, small errors in the inverse computation may lead to the basis $V_k$ computed by our algorithm to be quite far from the $k$-th order Krylov basis for $B, v$.

At a very high-level, the proof follows the outline of the proof for Lanczos method. We first show that, assuming the error in computing the inverse is small, $\widehat{T}_k$ can be used to approximate $B^i v$ for small $i$, *i.e.* for all $i \leq k$, we have, $\|B^i v - V_k \widehat{T}_k^i V_k^\top v\| \leq \varepsilon_2$, for some small $\varepsilon_2$. This implies that, for any polynomial $p$ of degree at most $k$,

$$\|p(B)v - V_k p(\widehat{T}_k) V_k^\top v\| \leq \varepsilon_2 \|p\|_1 ,$$

where if $p \stackrel{\text{def}}{=} \sum_{i=0}^k a_i \cdot x^i$, $\|p\|_1 = \sum_{i \geq 0}^k |a_i|$. This is the most technical part of the error analysis, and unfortunately, the only way we know of proving the error bound above is by a *brute-force* calculation. A part of this proof is to show that the spectrum of $\widehat{T}_k$ cannot shift far from the spectrum of $B$.

To bound the error in the candidate vector output by the algorithm, *i.e.* $\|f(B)v - V_k f(\widehat{T}_k) V_k^\top v\|$, we start by expressing $e^{-x}$ as the sum of a degree $k$-polynomial $p_k$ in $(1 + x/k)^{-1}$ and a remainder function $r_k$. We use the analysis from the previous paragraph to upper bound the error in the polynomial part by $\|p\|_1 \varepsilon_2$. We bound the contribution of the remainder term to the error by bounding $\|r_k(B)\|$ and $\|r_k(\widehat{T}_k)\|$. This step uses the fact that eigenvalues of $r_k(\widehat{T}_k)$ are $\{r_k(\lambda_i)\}_i$, where $\{\lambda_i\}_i$ are eigenvalues of $\widehat{T}_k$. This is the reason our algorithm symmetrizes $T_k$ to $\widehat{T}_k$. To complete the error analysis, we use the polynomials $p_k^\star$ from SSV and bound $\|p_k^\star\|_1$. Even though we do not know $p_k^\star$ explicitly, we can bound its coefficients indirectly by writing it as an interpolation polynomial. All these issues make the error analysis highly technical. However, since the error analysis is crucial for our algorithms, a more illuminating proof is highly desirable.

[5]EH show the existence of degree $k$ polynomials in $(1+\nu x)^{-1}$ for any *constant* $\nu \in (0,1)$, that approximate $e^{-x}$ up to an error of $\exp(1/2\nu - \Theta(\sqrt{k(\nu^{-1}-1)}))$. In order to deduce the claimed hypothesis, it needs to be used for $\nu \approx 1/\lambda_n(A)$, in which case, there is a factor of $e^{\lambda_n(A)}$ in the error, which could be huge.

### 3.2.3 Approximation Using Our Polynomial Approximation to $e^{-x}$

More straightforwardly, combining the Lanczos method with the setting $B \stackrel{\text{def}}{=} A$ and $f(x) \stackrel{\text{def}}{=} e^{-x}$, along with the polynomial approximation to $e^{-x}$ that we prove in Theorem 6, we get that setting $k \approx \sqrt{\lambda_1(A) - \lambda_n(A)} \cdot \text{poly}(\log 1/\delta)$ suffices to obtain a vector $u$ that satisfies $\|\exp(-A)v - u\| \leq \delta \|v\| \|\exp(-A)\|$. This gives us our second method for approximating $\exp(-A)v$. Note that this algorithm avoids any inverse computation and, as a result, the procedure and the proofs are simpler and the algorithm practical.

## 3.3 Our Uniform Approximation for $e^{-x}$

In this section, we give a brief overview of the proof of Theorem 6. More details appear in Section 6.

A straightforward approach to approximating $e^{-x}$ over $[a, b]$ is to truncate its series expansion around $\frac{a+b}{2}$. With a degree of the order of $(b-a) + \log 1/\delta$, these polynomials achieve an error of $\delta \cdot e^{-(b+a)/2}$, for any constant $\delta > 0$. This approach is equivalent to approximating $e^\lambda$ over $[-1, 1]$, for $\lambda \stackrel{\text{def}}{=} (b-a)/2$, by polynomials of degree $O(\lambda + \log 1/\delta)$. On the flip side, it is known that if $\lambda$ is constant, the above result is optimal (see e.g. [27]). Instead of polynomials, one could consider approximations by rational functions, as in [10, 36]. However, the author in [27] shows that, if both $\lambda$ and the degree of the denominator of the rational function are constant, the required degree of the numerator is only an additive constant better than that for polynomials. It might seem that the question of approximating the exponential has been settled and one cannot do much better. However, the result by SSV mentioned before, seems surprising in this light. The lower bound does not apply to their result, since the denominator of their rational function is unbounded. In a similar vein, we ask the following question: If we are looking for weaker error bounds, *e.g.* $\delta \cdot e^{-a}$ instead of $\delta \cdot e^{-(b+a)/2}$ (recall $b > a$), can we improve on the degree bound of $O((b-a) + \log 1/\delta)$? Theorem 6 answers this question in the affirmative and gives a new upper bound and an almost matching lower bound. We give an overview of the proofs of both these results next.

*Upper Bound.*

We wish to show that there exists a polynomial of degree of the order of $\sqrt{b-a} \cdot \text{poly}(\log 1/\delta)$ that approximates $e^{-x}$ on the interval $[a, b]$, up to an error of $\delta \cdot e^{-a}$, for any $\delta > 0$. Our approach is to approximate $(1 + x/k)^{-1}$ on the interval $[a, b]$, by a polynomial $q$ of degree $l$, and then compose the polynomial $p_k^\star$ from the SSV result with $q$, to obtain $p_k^\star(q(x))$ which is a polynomial of degree $k \cdot l$ approximating $e^{-x}$ over $[a, b]$. Thus, we are looking for polynomials $q$ that minimize $|q(x) - 1/x|$ over $[1 + a/k, 1 + b/k]$. Slightly modifying the optimization, we consider polynomials $q$ that minimize $|x \cdot$

$q(x)-1|$ over $[1+a/k, 1+b/k]$. We can show that the solution to this modified optimization can be derived from the well-known Chebyshev polynomials. For the right choice of $k$ and $l$, the composition of the two polynomials approximates $e^{-x}$ to within an error of $\delta \cdot e^{-a}$ over $[a,b]$, and has degree $\sqrt{b-a} \cdot \text{poly}(\log 1/\delta)$ . To bound the error in the composition step, we need to bound the sum of absolute values of coefficients of $p_k^\star$, which we achieve by rewriting $p_k^\star$ as an interpolation polynomial. More details appear in Section 6.

*Lower Bound.*

As already noted, since we consider a weaker error bound $\delta \cdot e^{-a}$ and $\lambda \stackrel{\text{def}}{=} (b-a)/2$ isn't a constant for our requirements, the lower bounds mentioned above no longer hold. Nevertheless, we prove that the square-root dependence of the required degree, on $(b-a)$, is optimal. The proof is simple and we sketch it here: Using a theorem of Markov from approximation theory (see [8]), we show that, any polynomial approximating $e^{-x}$ over the interval $[a,b]$ up to an error of $\delta \cdot e^{-a}$, for some constant $\delta$ small enough, must have degree at least of the order of $\sqrt{b-a}$. Markov's theorem states that for a univariate polynomial $p$ of degree $k$, which lives in a box of height $h$ over an interval of width $w$, the absolute value of the derivative $p'$ in the interval is at most $d^2h/w$. Let $p_k$ be a polynomial of degree $k$ that $\delta \cdot e^{-a}$-approximates $e^{-x}$ in the interval $[a,b]$. If $b-a$ is large enough and $\delta$ a small enough constant, then one can get a lower bound of $\Omega(e^{-a})$ on the derivative of $p_k$ using the Mean Value Theorem. Also, one can obtain an upper bound of $O(e^{-a})$ on the height of the box in which $p_k$ lives. Both these bounds use the fact that $p_k$ approximates $e^{-x}$ and is $\delta \cdot e^{-a}$ close to it. Since the width of the box is $b-a$, these two facts, along with Markov's theorem, immediately imply a lower bound of $\Omega(\sqrt{b-a})$ on $k$. This shows that our upper bound is tight up to a factor of $\text{poly}(\log 1/\delta)$.

# 4. THE ALGORITHM FOR BALANCED SEPARATOR

In this section, we provide a description of our algorithm BalSep and an overview of the proofs of Theorem 1 and Theorem 2. We start with some preliminaries.

## 4.1 Basic Preliminaries

*Instance Graph and Edge Volume.* We denote by $G=(V,E)$ the unweighted instance graph, where $V=[n]$ and $|E|=m$. We assume $G$ is connected. We let $d \in \mathbb{R}^n$ be the degree vector of $G$, i.e. $d_i$ is the degree of vertex $i$. For a subset $S \subseteq V$, we define the edge volume as $\mathsf{vol}(S) \stackrel{\text{def}}{=} \sum_{i \in S} d_i$. The total volume of $G$ is $2m$. The conductance of a cut $(S,\bar{S})$ is defined to be $\phi(S) \stackrel{\text{def}}{=} |E(S,\bar{S})|/\min\{\mathsf{vol}(S),\mathsf{vol}(\bar{S})\}$, where $\mathsf{vol}(S)$ is the sum of the degrees of the vertices in the set $S$. Moreover, a cut $(S,\bar{S})$ is $b$-balanced if $\min\{\mathsf{vol}(S),\mathsf{vol}(\bar{S})\} \geq b \cdot \mathsf{vol}(S)$.

*Special Graphs.* We denote by $K_V$ the complete graph with weight $d_i d_j/2m$ between every pair $i,j \in V$. For $i \in V$, $\mathsf{Star}_i$ is the star graph rooted at $i$, with edge weight of $d_i d_j/2m$ between $i$ and $j$, for all $j \in V$.

*Graph matrices.* For an undirected graph $H=(V,E_H)$, let $A(H)$ denote the adjacency matrix of $H$, and $D(H)$ the diagonal matrix of degrees of $H$. The (combinatorial) Laplacian of $H$ is defined as $L(H) \stackrel{\text{def}}{=} D(H) - A(H)$. Note that for all $x \in \mathbb{R}^V$, $x^\top L(H) x = \sum_{\{i,j\} \in E_H} (x_i - x_j)^2$. By $D$ and $L$, we denote $D(G)$ and $L(G)$ respectively for the input graph $G$. Finally, the natural random walk over $G$ has transition matrix $W \stackrel{\text{def}}{=} AD^{-1}$.

*Vector and Matrix Notation.* We are working within the vector space $\mathbb{R}^n$. We will denote by $I$ the identity matrix over this space. For a symmetric matrix $A$, we will use $A \succeq 0$ to indicate that $A$ is positive semi-definite. The expression $A \succeq B$ is equivalent to $A - B \succeq 0$. For two matrices $A, B$ of equal dimensions, let $A \bullet B \stackrel{\text{def}}{=} \text{Tr}(A^\top B) = \sum_{ij} A_{ij} \cdot B_{ij}$. We denote by $\{e_i\}_{i=1}^n$ the standard basis for $\mathbb{R}^n$. $\mathbf{0}$ and $\mathbf{1}$ will denote the all 0s and all 1s vectors respectively.

*Fact 1.* $L(K_V) = D - 1/2m \cdot D\mathbf{1}\mathbf{1}^\top D = D^{1/2}(I - 1/2m \cdot D^{1/2}\mathbf{1}\mathbf{1}D^{1/2})D^{1/2}$.

*Embedding Notation.* We will deal with vector embeddings of $G$, where each vertex $i \in V$ is mapped to a vector $v_i \in \mathbb{R}^d$, for some $d \leq n$. For such an embedding $\{v_i\}_{i \in V}$, we denote by $v_{\mathsf{avg}}$ the mean vector, i.e. $v_{\mathsf{avg}} \stackrel{\text{def}}{=} \sum_{i \in V} d_i/2m \cdot v_i$. Given a vector embedding $\{v_i \in \mathbb{R}^d\}_{i \in V}$, recall that $X$ is the Gram matrix of the embedding if $X_{ij} = v_i^\top v_j$. A Gram matrix $X$ is always PSD, *i.e.*, $X \succeq 0$. For any $X \in \mathbb{R}^{n \times n}, X \succeq 0$, we call $\{v_i\}_{i \in V}$ the *embedding corresponding to* $X$ if $X$ is the Gram matrix of $\{v_i\}_{i \in V}$. For $i \in V$, we denote by $R_i$ the matrix such that $R_i \bullet X = \|v_i - v_{\mathsf{avg}}\|^2$.

*Fact 2.* $\sum_{i \in V} d_i R_i \bullet X = \sum_{i \in V} d_i \|v_i - v_{\mathsf{avg}}\|^2 = 1/2m \cdot \sum_{i<j} d_j d_i \|v_i - v_j\|^2 = L(K_V) \bullet X$.

## 4.2 AHK Random Walks

The random-walk processes used by our algorithm are continuous-time Markov processes [24] over the vertices of $G$. The simplest such process is the *heat kernel* process, which is defined as having transition rate matrix $Q = -(I-W) = -LD^{-1}$ and probability transition matrix $e^{-\tau LD^{-1}}$ at time $\tau$. [6]

For the construction of our algorithm, we generalize the concept of heat kernel to a larger class of continuous-time Markov processes, which we name *Accelerated Heat Kernel* (AHK) processes. A process $\mathcal{H}(\beta)$ in this class is defined by a non-negative vector $\beta \in \mathbb{R}^n$ and the transition rate matrix of $\mathcal{H}(\beta)$ is $Q(\beta) \stackrel{\text{def}}{=} -(L + \sum_{i \in V} \beta_i L(\mathsf{Star}_i))D^{-1}$. As this is the negative of a sum of Laplacian matrices, it is easy to verify that it is a valid transition rate matrix. The effect of adding the star terms to the transition rate matrix is that of accelerating the convergence of the process to stationary at vertices $i$ with large value of $\beta_i$, as a large fraction of the probability mass that leaves these vertices is distributed uniformly over the edges. We denote by $P_\tau(\beta)$ the probability-transition matrix of $\mathcal{H}(\beta)$ between time 0 and $\tau$, i.e. $P_\tau(0) = e^{\tau Q(\beta)}$.

[6] The heat kernel can also be interpreted as the probability transition matrix of the following discrete-time random walk: sample a number of steps $i$ from a Poisson distribution with mean $\tau$ and perform $i$ steps of the natural random walk over $G$.

We remark that all the AHK random walks in BalSep are run for time $\tau \stackrel{\text{def}}{=} {}^{O(\log n)}/_{\gamma}$. This choice ensures that the walks must mix across all cuts of conductance much larger than $\gamma$, so that the embedding $P^{(t)}$ emphasizes cuts of the desired conductance.

*Embedding View.* A useful matrix to study $\mathcal{H}(\beta)$ will be $D^{-1}P_{2\tau}(\beta)$. This matrix describes the probability distribution over the edges of $G$ and has the advantage of being symmetric and positive semidefinite:

$$D^{-1}P_{2\tau}(\beta) = D^{-1/2}e^{-(2\tau)D^{-1/2}(L+\sum_{i\in V}\beta_i L(\mathsf{Star}_i))D^{-1/2}}D^{-1/2}.$$

Moreover, we have the following fact:

*Fact 3.* $D^{-1/2}P_\tau(\beta)$ is a square root of $D^{-1}P_{2\tau}(\beta)$.

Hence, $D^{-1}P_{2\tau}(\beta)$ is the Gram matrix of the embedding given by the columns of its square root $D^{-1/2}P_\tau(\beta)$. This property will enable us to use geometric SDP techniques to analyze $\mathcal{H}(\beta)$.

*Mixing.* Spectral methods for finding low-conductance cuts are based on the idea that random walk processes mix slowly across sparse cuts, so that it is possible to detect such cuts by considering the starting vertices for which the probability distribution of the process strongly deviates from stationary. We measure this deviation for vertex $i$ at time $t$ by the $\ell_2^2$-norm of the distance between $P_\tau(\beta)e_i$ and the uniform distribution *over the edges of* $G$. We denote it by $\Psi(P_\tau(\beta), i)$ :

$$\Psi(P_\tau(\beta), i) \stackrel{\text{def}}{=} d_i \sum_{j\in V} d_j \left(\frac{e_j^\top P_\tau(\beta) e_i}{d_j} - \frac{1}{2m}\right)^2$$

A fundamental quantity for our algorithm will be the *total deviation* from stationarity over a subset $S \subseteq V$. We will denote $\Psi(P_t(\beta), S) \stackrel{\text{def}}{=} \sum_{i\in S}\Psi(P_t(\beta), i)$. In particular, $\Psi(P_\tau(\beta), V)$ will play the role of potential function in our algorithm. The following facts express these mixing quantities in the geometric language of the embedding corresponding to $D^{-1}P_{2\tau}(\beta)$.

*Fact 4.* $\Psi(P_\tau(\beta), i) = d_i R_i \bullet D^{-1}P_{2\tau}(\beta)$.

*Fact 5.*

$$\Psi(P_\tau(\beta), V) = \sum_{i\in V} d_i R_i \bullet D^{-1}P_{2\tau}(\beta) = L(K_V) \bullet D^{-1}P_{2\tau}(\beta).$$

## 4.3 Algorithm Description

We are now ready to describe BalSep, which outputs a $c$-balanced cut of conductance $O(\sqrt{\gamma})$ or the string NO, if it finds a certificate that no $b$-balanced cut of conductance less than $\gamma$ exists. BalSep can also fail and output the string Fail. We will show that this only happens with small probability. The algorithm BalSep is formally presented [7] in Figure 1. We will consider embeddings given by the columns of $D^{-1/2}P_\tau(\beta)$ for some choice of $\beta$.

[7]The constants in this presentation are not optimized and are likely to be higher than what is necessary in practice. They can also be modified to obtain different trade-offs between the approximation guarantee and the output balance.

As we are only interested in Euclidean distances between vectors in the embedding, we use the Johnson-Lindenstrauss Lemma (see the full version for details) to obtain an $O(\log n)$-dimensional embedding approximately preserving distances between columns of $D^{-1/2}P_\tau(\beta)$ up to a factor of $(1+\varepsilon)$, where $\varepsilon$ is a constant such that ${}^{1+\varepsilon}/_{1-\varepsilon} \le {}^4/_3$.

The algorithm BalSep calls two subroutines FindCut and ExpV. FindCut is an SDP-rounding algorithm that uses random projections and radial sweeps to find a low-conductance cut, that is either $c$-balanced, for some constant $c = \Omega(b) \le {}^b/_{100}$ defined in OV, or obeys a strong guarantee stated in Theorem 8. Such algorithm is implicit in [22] and is described precisely in the full version of this paper. ExpV is a generic algorithm that approximately computes products of the form $P_\tau(\beta)u$ for unit vectors $u$. Expv can be chosen to be either the algorithm implied by Thereom 7, which makes use of the Spielman-Teng solver, or that in Theorem 5, which just applies the Lanczos method.

At iteration $t = 1$, we have $\beta^{(1)} = \mathbf{0}$, so that $P^{(1)}$ is just the probability transition matrix of the heat kernel on $G$ for time $\tau$. In general at iteration $t$, BalSep runs ExpV to compute $O(\log n)$ random projections of $P^{(t)}$ and constructs an approximation $\{v_i^{(t)}\}_{i\in V}$ to the embedding given by the columns of $D^{-1/2}P^{(t)}$. This approximate embedding has Gram matrix $X^{(t)}$.

In Step 2, BalSep computes $L(K_V) \bullet X^{(t)}$, which is an estimate of the total deviation $\Psi(P^{(t)}, V)$ by Fact 5. If this deviation is small, the AHK walk $P^{(t)}$ has mixed sufficiently over $G$ to yield a certificate that $G$ cannot have any $b$-balanced cut of conductance less than $\gamma$. This is shown in Lemma 1. If the AHK walk $P^{(t)}$ has not mixed sufficiently, we can use FindCut to find a cut $S^{(t)}$ of low conductance $O(\sqrt{\gamma})$, which is an obstacle for mixing. If $S^{(t)}$ is $c$-balanced, we output it and terminate. Similarly, if $S \cup S^{(t)}$ is $c$-balanced, as $\phi(S \cup S^{(t)}) \le O(\sqrt{\gamma})$, we can also output $S \cup S^{(t)}$ and exit. Otherwise, $S^{(t)}$ is unbalanced and is potentially preventing BalSep from detecting balanced cuts in $G$. We then proceed to modify the AHK walk, by increasing the values of $\beta^{(t+1)}$ for the vertices in $S^{(t)}$. This change lets $P^{(t+1)}$ mix faster from the vertices in $S^{(t)}$ and, in particular, mix across $S^{(t)}$. This ensure that in the next iterations, $S^{(t)}$ will be unlikely to be an obstacle to detecting more balanced cuts in $G$. We remark that, at any given iteration $t$, the support of $\beta^{(t)}$ is $\cup_{r=1}^{t-1} S^{(r)}$, which is an unbalanced set. Hence, the AHK walk $P^{(t)}$ looks like a heat-kernel random walk, whose convergence is accelerated only on a set of small volume.

In conclusion, the BalSep algorithm exactly parallels the RLE algorithm, introducing only two fundamental changes. First, we use the embedding given by the AHK random walk $P^{(t)}$ in place of the eigenvector to find cuts in $G$ or in a residual graph. Secondly, rather than fully removing unbalanced low-conductance cuts from the graph, we modify $\beta^{(t)}$ at every iteration $t$, so $P^{(t+1)}$ at the next iteration mixes across the unbalanced cuts found so far.

## 4.4 Analysis

The analysis of BalSep is at heart a modification of the MMWU argument in OV, stated in a random-walk language. This modification allows us to deal with the different embedding used by BalSep at every iteration with respect to OV. In this analysis, the quantity $\Psi(P^{(t)}, V)$ plays the

**Input:** An unweighted connected instance graph $G = (V, E)$, a constant balance value $b \in (0, 1/2]$, a conductance value $\gamma \in [1/n^2, 1)$.

Let $S = 0, \bar{S} = V$. Set $\tau \stackrel{\text{def}}{=} \log n/12\gamma$ and $\beta^{(1)} \stackrel{\text{def}}{=} \mathbf{0}$.

At iteration $t = 1, \ldots, T \stackrel{\text{def}}{=} 12 \log n$ :

1. Denote $P^{(t)} \stackrel{\text{def}}{=} P_\tau(\beta^{(t)})$. Pick $k \stackrel{\text{def}}{=} O(\log n/\varepsilon^2)$ random unit vectors $\{u_1^{(t)}, u_2^{(t)}, \ldots, u_k^{(t)} \in \mathbb{R}^n\}$ and use the subroutine ExpV to compute the embedding $\{v_i^{(t)} \in \mathbb{R}^k\}_{i \in V}$ defined as

$$\left(v_i^{(t)}\right)_j \stackrel{\text{def}}{=} \sqrt{\frac{n}{k}} u_j^\top D^{-1/2} P^{(t)} e_i.$$

   Let $X^{(t)}$ be the Gram matrix corresponding to this embedding.

2. If $L(K_V) \bullet X^{(t)} = \sum_{i \in V} d_i ||v_i^{(t)} - v_{\mathsf{avg}}^{(t)}||^2 \leq \frac{1+\varepsilon}{n}$, output NO and terminate.

3. Otherwise, run FindCut$(G, b, \gamma, \{v_i^{(t)}\}_{i \in V})$. FindCut outputs a cut $S^{(t)}$ with $\phi(S^{(t)}) \leq O(\sqrt{\gamma})$ or fails, in which case we also output Fail and terminate.

4. If $S^{(t)}$ is $c$-balanced, output $S^{(t)}$ and terminate. If not, update $S \stackrel{\text{def}}{=} S \cup S^{(t)}$. If $S$ is $c$-balanced, output $S$ and terminate.

5. Otherwise, update $\beta^{(t+1)} = \beta^{(t)} + \frac{72\gamma}{T} \sum_{i \in S^{(t)}} e_i$ and proceed to the next iteration.

Output NO and terminate.

**Figure 1: The BalSep Algorithm**

role of potential function. We start by showing that if the potential function is small enough, we obtain a certificate that no $b$-balanced cut of conductance at most $\gamma$ exists. In the second step, we show that, if an unbalanced cut $S^{(t)}$ of low conductance is found, the potential decreases by a constant fraction. Together, these two facts allow us to prove that $T = O(\log n)$ iterations suffices to either find a $\Omega(b)$-balanced cut of conductance $O(\sqrt{\gamma})$ or to give a certificate that no $b$-balanced cut of conductance less than $\gamma$ exists. The remaining proofs are included in the full version.

*Potential Guarantee.*

We argue that, if $\Psi(P^{(t)}, V)$ is sufficiently small, i.e. the walk $P^{(t)}$ mixes sufficiently from all starting points, it must be the case that $G$ has no $b$-balanced cut of conductance less than $\gamma$. A similar result is implicit in OV.

Lemma 1. *Let $S = \cup_{i=1}^t S^{(i)}$. If $\Psi(P^{(t)}, V) \leq \frac{4}{3n}$, and $\mathsf{vol}(S) \leq c \cdot 2m \leq b/100 \cdot 2m$, then*

$$L + \sum_{i \in V} \beta_i^{(t)} L(\mathsf{Star}_i) \succeq 3\gamma \cdot L(K_V).$$

*Moreover, this implies that no $b$-balanced cut of conductance less than $\gamma$ exists in $G$.*

For $t = 1$, the current random walk $P^{(1)}$ is just the heat-kernel of the graph. Then, the proof of Lemma 1 is a consequence of our choice of $\tau$, as $\Psi(P^{(1)}, V) \leq 4/\mathrm{poly}(3n)$ implies that

$$L \succeq \frac{1}{\tau} \cdot \log\left(\frac{3n}{4}\right) L(K_V) \succeq \gamma L(K_V).$$

In words, the fact that $P^{(1)}$ mixes well implies that the spectral gap of $G$ is large, so that no cut of conductance $\gamma$ can exists. For $t > 1$, a similar argument yields the first part of Lemma 1. The certificate is obtained by noticing that $P^{(t)}$ is accelerated on a small unbalanced set $S$, so that the term $\sum_{i \in V} \beta_i^{(t)} L(\mathsf{Star}_i)$ has little impact on the conductance of a balanced cut in $G$. In other words, if a balanced cut of conductance less than $\gamma$ existed, its convergence could not be greatly helped by the acceleration over $S$. Then, if $P^{(t)}$ is still mixing very well, no such balanced cut can exist.

*The Deviation of an Unbalanced Cut.*

In the next step, we show that, if the walk has not mixed sufficiently, w.h.p. the embedding $\{v_i^{(t)}\}_{i \in V}$, computed by BalSep, has low quadratic form with respect to the Laplacian of $G$. From a SDP-rounding perspective, this means that the embedding can be used to recover cuts of value close to $\gamma$. This part of the analysis departs from that of OV, as we use our modified definition of the embedding.

Lemma 2. *If $\Psi(P^{(t)}, V) \geq \frac{1}{n}$, then w.h.p. $L \bullet X^{(t)} \leq O(\gamma) \cdot L(K_V) \bullet X^{(t)}$.*

This guarantee on the embedding allows us to apply SDP-rounding techniques in the subroutine FindCut. The following result is implicit in [22]. Its proof appears in the full version.

Theorem 8. *Consider an embedding $\{v_i \in \mathbb{R}^d\}_{i \in V}$ with Gram matrix $X$ such that $L \bullet X^{(t)} \leq \alpha L(K_V) \bullet X^{(t)}$, for $\alpha > 0$. On input $(G, b, \alpha, \{v_i\}_{i \in V})$, FindCut runs in time $\tilde{O}(md)$ and w.h.p. outputs a cut $C$ with $\phi(C) \leq O(\sqrt{\alpha})$. Moreover, there is a constant $c = \Omega(b) \leq b/100$ such that either $C$ is $c$-balanced or*

$$\sum_{i \in C} d_i R_i \bullet X \geq 2/3 \cdot L(K_V) \bullet X.$$

The following corollary is a simple consequence of Lemma 2 and Theorem 8:

COROLLARY 1. *At iteration $t$ of* BALSEP, *if* $\Psi(P^{(t)}, V) \geq \frac{1}{n}$ *and* $S^{(t)}$ *is not $c$-balanced, then w.h.p.* $\Psi(P^{(t)}, S) \geq 1/2 \cdot \Psi(P^{(t)}, V)$.

In words, at the iteration $t$ of BALSEP, the cut $S^{(t)}$ must either be $c$-balanced or be an unbalanced cut that contributes a large constant fraction of the total deviation of $P^{(t)}$ from the stationary distribution. In this sense, $S^{(t)}$ is the main reason for the failure of $P^{(t)}$ to achieve better mixing. To eliminate this obstacle and drive the potential further down, $P^{(t)}$ is updated to $P^{(t+1)}$ by accelerating the convergence to stationary from all vertices in $S^{(t)}$. Formally, this is achieved by adding weighted stars rooted at all vertices over $S^{(t)}$ to the transition-rate matrix of the AHK random walk $P^{(t)}$.

*Potential Reduction.*

The next theorem crucially exploits the stability of the process $\mathcal{H}(\beta^{(t)})$ and Corollary 1 to show that the potential decreases by a constant fraction at every iteration in which an unbalanced cut is found. More precisely, the theorem shows that accelerating the convergence from $S^{(t)}$ at iteration $t$ of BALSEP has the effect of eliminating at least a constant fraction of the total deviation due to $S^{(t)}$. The proof is a simple application of the Golden-Thompson inequality [7] and mirrors the main step in the MMWU analysis.

THEOREM 9. *At iteration $t$ of* BALSEP, *if* $\Psi(P^{(t)}, V) \geq \frac{1}{n}$ *and* $S^{(t)}$ *is not $c$-balanced, then w.h.p.*

$$\Psi(P^{(t+1)}, V) \leq \Psi(P^{(t)}, V) - 1/3 \cdot \Psi(P^{(t)}, S^{(t)}) \leq 5/6 \cdot \Psi(P^{(t)}, V).$$

To complete the proof of Theorem 1 and Theorem 2, it now suffices to notice that $\Psi(P^{(1)}, V) \leq n$. Hence, by Theorem 9, after $O(\log n)$ iterations, if no low-conductance balanced cut has been found, we can apply Lemma 1 to give a certificate that no $b$-balanced cut of conductance less than $\gamma$ exists. To obtain the running-time guarantee, we notice that at each iteration BALSEP only needs to compute $k = O(\log n)$ products of the form $D^{-1/2}P^{(t)}u$, where $u$ is a unit vector. For this purpose, we apply the algorithms of Theorem 7 and Theorem 5. Detailed proofs of Theorem 1 and Theorem 2 are included in the full version.

## 4.5 SDP Interpretation

In this section we compare our algorithm with that of OV [22] who employed an SDP formulation for BS together with the Matrix Multiplicative Weight Update (MMWU) of [5]. We show how BALSEP has a natural interpretation in terms of the OV SDP and how it implies a new width reduction method.

OV designed an algorithm that outputs either a $\Omega(b)$-balanced cut of conductance $O(\sqrt{\gamma})$ or a certificate that no $b$-balanced cut of conductance $\gamma$ exists in time $\tilde{O}(m/\gamma^2)$. This algorithm uses the MMWU of Arora and Kale [5] to approximately solve an SDP formulation of the BS problem. The main technical contribution of their work is the routine FINDCUT (implicit in their ORACLE), which takes the role of an approximate separation oracle for their SDP. In an iteration of their algorithm, OV use the MMWU update to produce a candidate SDP-solution $Y^{(t)}$. In one scenario, $Y^{(t)}$ does not have sufficiently low Laplacian objective value:

$$L \bullet Y^{(t)} \geq \Omega(\gamma) L(K_V) \bullet Y^{(t)}. \quad (1)$$

In this case, the MMWU uses Equation 1 to produce a candidate solution $Y^{(t+1)}$ with lower objective value. Otherwise, FINDCUT is run on the embedding corresponding to $Y^{(t)}$. By Theorem 8, this yields either a cut of the required balance or a dual certificate that $Y^{(t)}$ is infeasible. This certificate has the form

$$\gamma \cdot \sum_{i \in S^{(t)}} d_i R_i \bullet Y^{(t)} \geq \Omega(\gamma) L(K_V) \bullet Y^{(t)} \quad (2)$$

and is used by the update to construct the next candidate $Y^{(t+1)}$. The number of iterations necessary is determined by the width of the two possible updates described above. A simple calculation shows that the width of the update for Equation 1 is $\Theta(1)$, while for Equation 2, it is only $O(\gamma)$. Hence, the overall width is $\Theta(1)$, implying that $O(\log n/\gamma)$ iteration are necessary for the algorithm of OV to produce a dual certificate that the SDP is infeasible and therefore no $b$-balanced cut of conductance $\gamma$ exists.

Our modification of the update is based on changing the starting candidate solutions from $Y^{(1)} \propto D^{-1}$ to $X^{(1)} \propto D^{-1/2}e^{-2\tau D^{-1/2}LD^{-1/2}}D^{-1/2}$. In Lemma 1 and Lemma 2, we show that this modification implies that all $X^{(t)}$ must now have $L \bullet X^{(t)} \leq O(\gamma) \cdot L(K_V) \bullet X^{(t)}$ or else we find a dual certificate that the SDP is infeasible. This additional guarantee effectively allows us to bypass the update of Equation 1 and only work with updates of the form given in Equation 2. As a result, our width is now $O(\gamma)$ and we only require $O(\log n)$ iterations.

Another way to interpret our result is that all possible $\tau \approxeq \log n/\gamma$ updates of the form of Equation 1 in the algorithm of OV are regrouped into a single step, which is performed at the beginning of the algorithm.

# 5. APPROXIMATING $\exp(-A)v$

In this section, we describe the algorithms for Theorems 3, 4 and 5 and give the steps involved in their proofs. We first describe the Lanczos method in Section 5.1, and then show how to use it to deduce Theorem 5. We then describe how to modify the Lanczos method to obtain the procedure EXPRATIONAL, in Section 5.2, and then show how to use EXPRATIONAL to deduce Theorems 3, 4 and 7. Finally, we give a proof sketch for our main theorem about the procedure EXPRATIONAL in Section 5.3. We give formal statements of various theorems and lemmas, but defer most proofs to the full version [20] due to space restrictions.

For this section, we will assume the upper bound from Theorem 6, regarding polynomials approximating $e^{-x}$. Please see Section 6 for more details about the proof of this theorem. We first state the basic definitions used in this section. Some of these definitions have been presented before, but are being reproduced here for completeness.

*Definitions.*

We work with square $n \times n$ matrices over $\mathbb{R}$. For a symmetric PSD matrix $M$, we define the $M$-norm of a vector $x$ as $\|x\|_M \stackrel{\text{def}}{=} (x^\top M x)^{1/2}$. For a matrix $M$, abusing notation, we denote its exponential by $\exp(-M)$, which is defined as $\sum_{i \geq 0} \frac{(-1)^i}{i!} M^i$. $\|M\| \stackrel{\text{def}}{=} \sup_{\|x\|=1} \|Mx\|$ denotes the spectral norm of $M$. $M$ is said to be *Symmetric and Diagonally Dominant* (SDD) if, $M_{ij} = M_{ji}$, for all $i, j$ and $M_{ii} \geq \sum_{j \neq i} |M_{ij}|$, for all $i$. $M$ is called *Upper Hessenberg* if, $(M)_{ij} = 0$ for $i > j+1$. $M$ is called *tridiagonal* if $M_{ij} = 0$

for $i > j+1$ and for $j > i+1$. Let $\lambda_1(M)$ and $\lambda_n(M)$ denote the largest and the smallest eigenvalues of $M$ respectively, and, let $\Lambda(M)$ denote the smallest interval containing the spectrum of $M$, *i.e.*, $[\lambda_n(M), \lambda_1(M)]$. For a matrix $M$, let $m_M$ denote the number of non-zero entries in $M$. Further, let $t_M$ denote the time required to multiply the matrix $M$ with a given vector $w$. In general $t_M$ depends on how $M$ is given as an input and can be $\Theta(n^2)$. However, it is possible to exploit the special structure of $M$ if given as an input appropriately: It is possible to just multiply the non-zero entries of $M$, giving $t_M = O(m_M)$. Also, if $M$ is a rank one matrix $ww^\top$, where $w$ is known, we can multiply by $M$ in $O(n)$ time. For any positive integer $k$, let $\Sigma_k$ denote the set of all polynomials with degree at most $k$. Given a degree $k$ polynomial $p \stackrel{\text{def}}{=} \sum_{i=0}^{k} a_i \cdot x^i$, the $\ell_1$ norm of $p$, denoted as $\|p\|_1$ is defined as $\|p\|_1 = \sum_{i\geq 0}^{k} |a_i|$.

## 5.1 Lanczos Method – From Scalars to Matrices

We give a description of the Lanczos method (*e.g.* see [26]) in Figure 2 and give a proof of a well-known theorem about the approximation guarantee and the running time of the method (Theorem 10). We then show how to deduce Theorem 5 (Simple algorithm for exponentiating PSD matrices) using Theorem 10.

Given an $n \times n$ symmetric matrix $B$, a function $f : \mathbb{R} \to \mathbb{R}$, and a vector $v$, the Lanczos method allows us to approximate $f(B)v$. Without loss of generality, we assume that $\|v\| = 1$. The Lanczos method looks for an approximation to $f(B)v$ of the form $p(B)v$, where $p$ is a polynomial of small degree, say $k$. Notice that $p(B)v$ lies in the subspace $\mathsf{Span}\{v, Bv, \ldots, B^k v\}$ – called the *Krylov subspace*. Before we describe the method, we recall the definition of a Krylov subspace.

*Definition 1. (Krylov Subspace).* Given a matrix $B$ and a vector $v$, the Krylov subspace of order $k$, denoted by $\mathcal{K}(B, v, k)$, is defined as the subspace spanned by the vectors $v, Bv, \ldots, B^k v$.

It will be convenient to work with an orthonormal basis for $\mathcal{K} \stackrel{\text{def}}{=} \mathcal{K}(B, v, k)$. Let $\{v_i\}_{i=0}^{k}$ be an orthonormal basis for $\mathcal{K}$. Let $V_k$ be the $n \times (k+1)$ matrix with $\{v_i\}_{i=0}^{k}$ as its columns. Thus, $V_k V_k^\top$ denotes the projection onto $\mathcal{K}$. Let $T_k$ be the $(k+1)\times(k+1)$ matrix expressing $B$ as an operator restricted to $\mathcal{K}$ in the basis $\{v_i\}_{i=0}^{k}$, *i.e.*, $T_k \stackrel{\text{def}}{=} V_k^\top B V_k$. Now, since $v, Bv \in \mathcal{K}$, we have,

$$Bv = (V_k V_k^\top) B (V_k V_k^\top) v = V_k (V_k^\top B V_k) V_k^\top v = V_k T_k V_k^\top v.$$

Similarly, for all $i \leq k$, $B^i v = V_k T_k^i V_k^\top v$, and hence, by linearity, $p(B)v = V_k p(T_k) V_k^\top v$, for any polynomial $p$ of degree at most $k$. We summarize this in the following lemma.

LEMMA 3. (Exact Computation with Polynomials. See *e.g.* [26]). *Let $V_k$ and $T_k$ be as defined above. For any polynomial $p$ of degree at most $k$,*

$$p(B)v = V_k p(T_k) V_k^\top v.$$

Thus, $T_k$ can be used to compute $p(B)v$ exactly for any degree $k$ polynomial $p$. This lemma suggests that a natural candidate for approximating $f(B)v$ is the vector $V_k f(T_k) V_k^\top v$, even when $f$ is not a degree $k$ polynomial. To determine the quality of this approximation, we will bound the norm of the error, *i.e.*, $\left\| f(B)v - V_k f(T_k) V_k^\top v \right\|$.

Writing $r_k(x) \stackrel{\text{def}}{=} f(x) - p_k(x)$, where $p_k$ is any degree $k$ approximation to $f(x)$, and using Lemma 3, we get,

$$f(B)v - V_k f(T_k) V_k^\top v = r_k(B)v - V_k r_k(T_k) V_k^\top v.$$

Hence, the norm of the error vector is at most $(\|r_k(B)\| + \|r_k(T_k)\|)\|v\|$, which is bounded by the value of $r_k$ on the eigenvalues of $B$ and $T_k$. To obtain the best bound, we can minimize this error bound over polynomials $p_k$ of degree at most $k$. This is summarized in the following lemma.

LEMMA 4. (Approximation by Best Polynomial. See *e.g.* [26]). *Let $V_k$ and $T_k$ be as defined above. Let $f : \mathbb{R} \to \mathbb{R}$ be any function such that $f(B)$ and $f(T_k)$ are well-defined. Then,*

$$\left\| f(B)v - V_k f(T_k) V_k^\top v \right\| \leq \min_{p_k \in \Sigma_k} \left( \max_{\lambda \in \Lambda(B)} |f(\lambda) - p_k(\lambda)| + \max_{\lambda \in \Lambda(T_k)} |f(\lambda) - p_k(\lambda)| \right).$$

Thus, $V_k f(T_k) V_k^\top v$ approximates $f(B)v$ as well as the *best* degree $k$ polynomial that uniformly approximates $f$. It remains to show how to compute $V_k$ and $T_k$. We address this next.

### 5.1.1 Efficiently Computing a Basis for the Krylov Subspace

We now show how to compute $V_k$ and $T_k$ efficiently. This is done iteratively as follows: Let $v_0 \stackrel{\text{def}}{=} v$. For $i = 0, \ldots, k-1$, we compute $Bv_i$ and remove the components along the vectors $\{v_0, \ldots, v_i\}$ to obtain a new vector that is orthogonal to $v_0, \ldots, v_i$. This vector, scaled to norm 1, is defined to be $v_{i+1}$ (similar to Gram-Schmidt orthonormalization). By construction, these vectors satisfy, for all $i \leq k$, $\mathsf{Span}\{v_0, \ldots, v_i\} = \mathsf{Span}\{v, Bv, \ldots, B^i v\}$.

Since $T_k = V_k^\top B V_k$, we have $(T_k)_{ij} = v_i^\top B v_j$. By construction, $Bv_j \in \mathsf{Span}\{v_0, \ldots, v_{j+1}\}$, and if $i > j+1$, $v_i$ is orthogonal to this subspace, and hence $v_i^\top (Bv_j) = 0$. Thus, $T_k$ is Upper Hessenberg, *i.e.*, $(T_k)_{ij} = 0$ for $i > j+1$. Moreover, if $B$ is symmetric, $v_j^\top (Bv_i) = v_i^\top (Bv_j)$, and hence $T_k$ is symmetric and tridiagonal. This implies that we need to orthogonalize $Bv_i$ only w.r.t $v_i$ and $v_{i-1}$. Thus, we need to compute only $O(k)$ dot-products while computing the basis, instead of $O(k^2)$; and time required for computing the dot-products is $O(nk)$, instead of $O(nk^2)$. This is crucial for the proof of Theorem 5 and Theorem 2.

We summarize the procedure below. A formal description of the LANCZOS algorithm is given in Figure 2.

1. Compute the basis $\{v_i\}_{i=0}^{k}$ – Start with $v_0 \stackrel{\text{def}}{=} v$. For $i = 0, \ldots, k-1$, compute $Bv_i$ and orthogonalize it to $v_i$ and $v_{i-1}$. Scale the vector to unit norm to get $v_{i+1}$.
2. Construct the matrices $V_k, T_k$, and return the vector $V_k f(T_k) V_k^\top v$ as the candidate approximation to $f(B)v$.

The following theorem summarizes the main result about this procedure.

THEOREM 10. (LANCZOS Theorem. See *e.g.* [26]) *Given a symmetric PSD matrix $B$, a vector $v$ with $\|v\| = 1$, a function $f$, and a positive integer parameter $k$ as inputs, the procedure* LANCZOS *computes a vector $u$ such that,*

$$\|f(B)v - u\| \leq 2 \cdot \min_{p_k \in \Sigma_k} \max_{\lambda \in \Lambda(B)} |f(\lambda) - p_k(\lambda)|.$$

**Input**: A symmetric matrix $B \succeq 0$, a vector $v$ such that $\|v\| = 1$, a positive integer $k$, and a function $f : \mathbb{R} \to \mathbb{R}$.
**Output**: A vector $u$ that is an approximation to $f(B)v$.

1. Initialize $v_0 \stackrel{\text{def}}{=} v$.
2. For $i = 0$ to $k-1$, (Construct an orthonormal basis to Krylov subspace of order $k$)
   a. If $i = 0$, compute $w_0 \stackrel{\text{def}}{=} Bv_0$. Else, compute $w_i \stackrel{\text{def}}{=} Bv_i - \beta_i v_{i-1}$. (Orthogonalize w.r.t. $v_{i-1}$)
   b. Define $\alpha_i \stackrel{\text{def}}{=} v_i^\top w_i$ and $w_i' \stackrel{\text{def}}{=} w_i - \alpha_i v_i$ *. (Orthogonalize w.r.t. $v_i$)
   c. Define $\beta_{i+1} \stackrel{\text{def}}{=} \|w_i'\|$ and $v_{i+1} \stackrel{\text{def}}{=} w_i'/\beta_{i+1}$. (Scaling it to norm 1)
3. Let $V_k$ be the $n \times (k+1)$ matrix whose columns are $v_0, \ldots, v_k$ respectively.
4. Let $T_k$ be the $(k+1) \times (k+1)$ matrix such that for all $i$, $(T_k)_{ii} = v_i^\top B v_i = \alpha_i, (T_k)_{i,i+1} = (T_k)_{i+1,i} = v_{i+1}^\top B v_i = \beta_{i+1}$ and all other entries are 0. (Compute $T_k \stackrel{\text{def}}{=} V_k^\top B V_k$)
5. Compute $\mathcal{B} \stackrel{\text{def}}{=} f(T_k)$ exactly via eigendecomposition. Output the vector $V_k \mathcal{B} V_k^\top v$.

* If $w_i' = 0$, compute the approximation with the matrices $T_{i-1}$ and $V_{i-1}$, instead of $T_k$ and $V_k$. The error bounds still hold.

**Figure 2: The** Lanczos **algorithm for approximating** $f(B)v$

*The time taken by* Lanczos *is* $O\left((n + t_B)k + k^2\right)$.

Proof. The algorithm Lanczos implements the Lanczos method we've discussed here. The error guarantee follows from Lemma 4 and the fact that $\Lambda(T_k) \subseteq \Lambda(B)$. The total running time is dominated by $k$ multiplications of $B$ with a vector, $O(k)$ dot-products and the eigendecomposition of the tridiagonal matrix $T_k$ to compute $f(T_k)$ (which can be done in $O(k^2)$ time [23]), giving a total running time of $O\left((n + t_B)k + k^2\right)$. □

Combining the approximation guarantee of the Lanczos algorithm given by Theorem 10 for the setting $B \stackrel{\text{def}}{=} A$ and $f(x) \stackrel{\text{def}}{=} e^{-x}$, along with the polynomial approximation to $e^{-x}$ that we prove in Theorem 6, we obtain a proof of the following theorem which is easily seen to imply Theorem 5.

Theorem 11. (Running Time Using Lanczos). *Given a symmetric PSD matrix $A$, a vector $v$ with $\|v\| = 1$ and a parameter $0 < \delta \leq 1$, for $k$ that is*

$$O\left(\sqrt{\max\{\log 1/\delta, (\lambda_1(A) - \lambda_n(A))\}}\,(\log 1/\delta)^{3/2} \cdot \log\log 1/\delta\right),$$

*and $f(x) = e^{-x}$, the procedure* Lanczos *computes a vector $u$ such that $\|\exp(-A)v - u\| \leq \|\exp(-A)\|\,\delta$. The time taken by* Lanczos *is $O\left((n + t_A)k + k^2\right)$.*

This completes our description of the Lanczos method, and how to deduce Theorem 5 using the method.

## 5.2 Approximating $\exp(-A)v$ Using a Rational Approximation to $e^{-x}$

In this section, we describe the steps involved in the proofs of Theorem 3 (Exponentiating SDD matrices), Theorem 4 (Exponentiating PSD matrices) and Theorem 7 (Matrix exponentials required for BalSep). We show how to modify the Lanczos method to obtain an algorithm, which we call ExpRational, that underlies these theorems. The starting point is the following result by Saff, Schönhage and Varga (SSV) [28].

Theorem 12. (Rational Approximation [28]). *There exists constants $c_1 \geq 1$ and $k_0$ such that, for any positive integer $k \geq k_0$, there exists a polynomial $p_k^\star(x)$ of degree $k$ such that $p_k^\star(0) = 0$, and,*

$$\sup_{t \in (0,1]} \left|e^{-k/t+k} - p_k^\star(t)\right| = \sup_{x \in [0,\infty)} \left|e^{-x} - p_k^\star\left((1 + x/k)^{-1}\right)\right| \leq c_1 k \cdot 2^{-k}.$$

This result implies that for any positive integer $k$, there exists a degree $k$ polynomial $p_k^\star$ such that, $p_k^\star((1 + x/k)^{-1})$ approximates $e^{-x}$ up to an error of $O(k \cdot 2^{-k})$ over the interval $[0, \infty)$. This suggests we can approximate $\exp(-A)v$, using the Lanczos method with $B \stackrel{\text{def}}{=} (I + A/k)^{-1}$ and $f(x) \stackrel{\text{def}}{=} e^{k(1-1/x)}$. However, exact inverse computation is expensive, and we will have to work with approximate inverse computation. We abstract out the required inversion procedure as $\mathsf{Invert}_A$ and require the following guarantee: given a vector $y$, a positive integer $k$, and an $\varepsilon_1 > 0$, $\mathsf{Invert}_A(y, k, \varepsilon_1)$ returns a vector $u_1$ such that, $\left\|(I + A/k)^{-1} y - u_1\right\| \leq \varepsilon_1 \|y\|$. We will use $\mathsf{Invert}_A$ to approximately multiply a given vector with $(I + A/k)^{-1}$ during each iteration of the Lanczos algorithm. Due to approximate computation, two more changes to the Lanczos method are required:

1. At every step, we need to orthonormalize $w_i \approx (I + A/k)^{-1} v_i$ w.r.t all vectors $v_0, \ldots, v_i$.
2. We compute the candidate approximation using the symmetrized matrix $\widehat{T}_k \stackrel{\text{def}}{=} 1/2 \cdot (T_k + T_k^\top)$.

We now summarize our procedure below. A formal description of the ExpRational procedure is given in Figure 3.

1. Compute the basis $\{v_i\}_{i=0}^k$ – Start with $v_0 \stackrel{\text{def}}{=} v$. For $i = 0, \ldots, k-1$,
   (a) Use $\mathsf{Invert}_A$ to obtain $w_i$, an approximation to $(I + A/k)^{-1} v_i$.
   (b) Orthogonalize $w_i$ to $v_0, \ldots, v_i$. Scale the vector to unit norm to get $v_{i+1}$.

**Input**: A Matrix $A \succeq 0$, a vector $v$ such that $\|v\| = 1$, and an approximation parameter $0 < \delta \leq 1$.
**Output**: A vector $u$ such that $\|\exp(-A)v - u\| \leq \delta$.
**Parameters:** Let $k \stackrel{\text{def}}{=} O(\log 1/\delta)$ and $\varepsilon_1 \stackrel{\text{def}}{=} \exp(-\Theta(k \log k + \log(1 + \|A\|)))$.

1. Initialize $v_0 \stackrel{\text{def}}{=} v$.
2. For $i = 0$ to $k - 1$, (Construct an orthonormal basis to Krylov subspace of order $k$)
   a. Call the procedure $\mathsf{Invert}_A(v_i, k, \varepsilon_1)$. The procedure returns a vector $w_i$, such that, $\left\|(I + A/k)^{-1} v_i - w_i\right\| \leq \varepsilon_1 \|v_i\|$. (Approximate $(I + A/k)^{-1} v_i$)
   b. For $j = 0, \ldots, i$,
      i. Let $\alpha_{j,i} \stackrel{\text{def}}{=} v_j^\top w_i$. (Compute projection onto $w_i$)
   c. Define $w_i' \stackrel{\text{def}}{=} w_i - \sum_{j=0}^{i} \alpha_{j,i} v_j$. (Orthogonalize w.r.t. $v_j$ for $j \leq i$)
   d. Let $\alpha_{i+1,i} \stackrel{\text{def}}{=} \|w_i'\|$ * and $v_{i+1} \stackrel{\text{def}}{=} w_i'/\alpha_{i+1,i}$. (Scaling it to norm 1)
   e. For $j = i + 2, \ldots, k$,
      i. Let $\alpha_{j,i} \stackrel{\text{def}}{=} 0$.
3. Let $V_k$ be the $n \times (k + 1)$ matrix whose columns are $v_0, \ldots, v_k$ respectively.
4. Let $T_k$ be the $(k + 1) \times (k + 1)$ matrix $(\alpha_{i,j})_{i,j \in \{0,\ldots,k\}}$ and $\widehat{T}_k \stackrel{\text{def}}{=} 1/2(T_k^\top + T_k)$. (Symmetrize $T_k$)
5. Compute $\mathcal{B} \stackrel{\text{def}}{=} \exp\left(k \cdot (I - \widehat{T}_k^{-1})\right)$ exactly and output the vector $V_k \mathcal{B} V_k^\top v$.

* If $w_i' = 0$, compute the approximation with the matrices $T_{i-1}$ and $V_{i-1}$, instead of $T_k$ and $V_k$. The error bounds still hold.

**Figure 3: The** ExpRational **algorithm for approximating** $\exp(-A)v$

2. Construct the matrices $V_k, T_k$. Compute $\widehat{T}_k$ and return the vector $V_k f(\widehat{T}_k) V_k^\top v$ as the candidate approximation to $f(B)v$.

The following theorem summarizes the main result about this procedure.

Theorem 13. (Running Time of ExpRational). *Given a symmetric PSD matrix $A \succeq 0$, a vector $v$ with $\|v\| = 1$, an error parameter $0 < \delta \leq 1$ and oracle access to $\mathsf{Invert}_A$, for parameters $k \stackrel{def}{=} O(\log 1/\delta)$ and $\varepsilon_1 \stackrel{def}{=} \exp(-\Theta(k \log k + \log(1+\|A\|)))$, ExpRational computes a vector $u$ such that $\|\exp(-A)v - u\| \leq \delta$, in time $O(T^{inv}_{A,k,\varepsilon_1} \cdot k + n \cdot k^2 + k^3)$, where $T^{inv}_{A,k,\varepsilon_1}$ is the time required by $\mathsf{Invert}_A(\cdot, k, \varepsilon_1)$.*

*Remark 1.* Note that there is an $n \cdot k^2$ term in the running time for ExpRational, in contrast with $n \cdot k$ in the running time for Lanczos. This is because, in ExpRational, we can no longer guarantee that the matrix $T_k$ is tridiagonal, in contrast with Lanczos. For ExpRational, $k$ is small ($O(\log n)$ for our application) and hence the term $n \cdot k^2$ does not hurt. Whereas, for Lanczos, $k$ is large ($\tilde{O}(1/\sqrt{\varepsilon})$ for our application), and a term of $n \cdot k^2$ in the running time would be prohibitive.

*Remark 2.* Note that there is a $k^3$ term in the running time for ExpRational, and a $k^2$ term in the running time for Lanczos. This corresponds to the time required for computing the eigendecomposition of a $(k + 1) \times (k + 1)$ symmetric matrix. While this process requires $O(k^3)$ time in general; in case of Lanczos, the matrix is tridiagonal and hence the time required is $O(k^2)$ (see [23]).

The proof of Theorem 13 is the most technical part of the paper. The main issue is that the error in approximating the matrix inverse at each iteration, propagates to later steps. We need to bound the error introduced in the output vector because of the error at each iteration. We give the steps involved in the proof in Section 5.3

We now give different implementations of $\mathsf{Invert}_A$ in order to prove Theorems 3, 4 and 7 in Sections 5.2.1, 5.2.2 and 5.2.3, respectively.

### 5.2.1 SDD Matrices – Theorem 3

For Theorem 3 about exponentiating SDD matrices, we implement the $\mathsf{Invert}_A$ procedure in ExpRational using the Spielman-Teng SDD solver [34]. Here, we state an improvement on the Spielman-Teng result by Koutis, Miller and Peng [18].

Theorem 14. (SDD Solver [18]). *Given a system of linear equations $Mx = b$, where the matrix $M$ is SDD, and an error parameter $\varepsilon > 0$, it is possible to obtain a vector $u$ that is an approximate solution to the system, in the sense that*

$$\|M^{-1}b - u\|_M \leq \varepsilon \|M^{-1}b\|_M \ .$$

*The time required for computing $u$ is $\tilde{O}(m_M \log n \log 1/\varepsilon)$, where $M$ is an $n \times n$ matrix. (The tilde hides $\log \log n$ factors.)*

In order to prove Theorem 3, we use the ExpRational procedure to approximate the exponential. At every call to $\mathsf{Invert}_A(y, k, \varepsilon_1)$, we call the SDD solver with the matrix $(I + A/k)$, vector $y$ and error parameter $\varepsilon_1$, and return the vector $u_1$ returned by the solver. Note that the matrix $(I +$

$A/k$) is SDD. Also note that the guarantee on $u$ provided by the SDD solver, in terms of $\|\cdot\|_{I+A/k}$, is different from the guarantee required by $\mathsf{Invert}_A$, which is in terms of $\|\cdot\|$. However,

$$\|(I + A/k)^{-1}y - u_1\| \leq \left\|(I + A/k)^{-1}y - u_1\right\|_{(I+A/k)} \leq \varepsilon_1 \cdot \|(I + A/k)^{-1}y\|_{(I+A/k)} \leq \varepsilon_1 \cdot \|y\|.$$

Thus, the vector $u_1$ satisfies guarantee required for $\mathsf{Invert}_A$. Thus, Theorem 13 implies that the procedure ExpRational computes a vector $u$ approximating $e^{-A}v$, as desired.

Using Theorem 14, the time required by each call to $\mathsf{Invert}_A$, *i.e.*, $T^{\text{inv}}_{A,k,\varepsilon_1}$ is $\tilde{O}\left((m_A + n)\log n \log 1/\varepsilon_1\right)$, and hence, from Theorem 13, the total running time is

$$\tilde{O}\left((m_A + n)\log n(\log 1/\delta + \log(1+\|A\|))\log 1/\delta + (\log 1/\delta)^3\right),$$

where the tilde hides polynomial factors in $\log\log n$ and $\log\log 1/\delta$. This completes the proof of Theorem 3.

### 5.2.2 General PSD Matrices – Theorem 4

The proof of Theorem 4 is identical to that of Theorem 3 except that we replace the SDD solver by the Conjugate Gradient method for implementing the $\mathsf{Invert}_A$ procedure. We use the following theorem.

Theorem 15. (Conjugate Gradient Method. See [31]). *Given a system of linear equations $Mx = b$ and an error parameter $\varepsilon > 0$, it is possible to obtain a vector $u$ that is an approximate solution to the system, in the sense that*

$$\|u - M^{-1}b\|_M \leq \varepsilon\|M^{-1}b\|_M.$$

*The time required for computing $u$ is $O\left(t_M\sqrt{\kappa(M)}\log 1/\varepsilon\right)$, – where $\kappa(M)$ denotes the condition number of $M$.*

*Remark 3.* Note that, in comparison to the SDD solver, the Conjugate Gradient method has a significantly larger running time in general, because of the $\sqrt{\kappa(M)}$ factor. However, the Conjugate Gradient method only requires multiplication by the matrix $M$, hence the factor $t_M$ in the running time, which could be smaller than the $m_M$ factor in the running time for the SDD Solver.

As with the SDD solver, the guarantee on the returned vector is in terms of $\|\cdot\|_{I+A/k}$, but as observed in Section 5.2.1, this implies the guarantee required by $\mathsf{Invert}_A$. Using Theorem 15, the time required by each call to $\mathsf{Invert}_A$, $T^{\text{inv}}_{A,k,\varepsilon_1}$ is, $O\left(t_A\sqrt{\kappa(I + A/k)}\log 1/\varepsilon_1\right) = O\left(t_A\sqrt{1 + \|A\|}\log 1/\varepsilon_1\right)$, and hence, from Theorem 13, the total running time is

$$\tilde{O}\left(t_A\sqrt{1 + \|A\|}(\log 1/\delta + \log(1 + \|A\|))\log 1/\delta + (\log 1/\delta)^2\right),$$

where the tilde hides polynomial factors in $\log\log n$ and $\log\log 1/\delta$. This completes the proof of Theorem 4.

### 5.2.3 Beyond SDD - Theorem 7

Theorem 7 requires us to approximate $\exp(-A)v$, for a given vector $v$ and matrix $A = \Pi HMH\Pi$, where $M$ is an SDD matrix, $H$ is a diagonal matrix with strictly positive entries and $\Pi$ is a rank $(n-1)$ projection matrix, $\Pi \stackrel{\text{def}}{=} I - ww^\top$ ($w$ is explicitly known and $\|w\| = 1$). Since $A$ may be neither SDD, nor sparse, Theorem 3 does not suffice, whereas the running times in Theorems 4 and 5 are slow for our requirements.

Approximating the exponential of matrices of the form $\Pi HMH\Pi$ is required by the algorithm for our main result (Theorem 1). The SDD solver (Theorem 14) does not suffice to implement the $\mathsf{Invert}_A$ procedure for such matrices. Fortunately, Lemma 5 given below implements the $\mathsf{Invert}_A$ procedure for such matrices.

Lemma 5. ($\mathsf{Invert}_A$ Procedure for Theorem 7). *Given a positive integer $k$, vector $y$, an error parameter $\varepsilon_1$, a rank $(n-1)$ projection matrix $\Pi = I - ww^\top$ (where $\|w\| = 1$ and $w$ is explicitly known), a diagonal matrix $H$ with strictly positive entries, and an invertible SDD matrix $M$ with $m_M$ non-zero entries, let $M_1$ denote the matrix $(I + 1/k \cdot \Pi HMH\Pi)$. We can compute a vector $u$ such that*

$$\left\|M_1^{-1}y - u\right\| \leq \varepsilon_1 \left\|M_1^{-1}y\right\|,$$

*in time $\tilde{O}((m_M + n)\log n \log \frac{1+\|HMH\|}{\varepsilon_1})$. (The tilde hides $\text{poly}(\log\log n)$ factors.)*

Before we sketch a proof of Lemma 5, we complete the proof of Theorem 7 assuming Lemma 5. We use the ExpRational procedure with the $\mathsf{Invert}_A$ procedure given by Lemma 5. The time required for each call to $\mathsf{Invert}_A$ is $T^{\text{inv}}_{A,k,\varepsilon_1} \stackrel{\text{def}}{=} \tilde{O}((m_M + n)\log n \log \frac{1+\|HMH\|}{\varepsilon_1})$, and hence from Theorem 13, we get that we can compute the desired vector $u$ approximating $e^{-A}v$ in total time

$$\tilde{O}\left((m_M + n)\log n(\log 1/\delta + \log(1 + \|HMH\|))\log 1/\delta + (\log 1/\delta)^3\right),$$

where the tilde hides polynomial factors in $\log\log n$ and $\log\log 1/\delta$. This completes the proof of Theorem 7. We now give a proof sketch for Lemma 5, which is proven by combining the SDD Solver with the Sherman-Morrison formula.

*Proof Sketch for Lemma 5.* Using the fact that $w$ is an eigenvector of our matrix, we will split $y$ into two components – one along $w$ and one orthogonal (denote it $z$). Along $w$, we can easily compute the component of the required vector. Among the orthogonal component, we will write our matrix as the sum of $I + 1/k \cdot HMH$ and a rank one matrix, and use the Sherman-Morrison formula to express its inverse. Since we can write

$$(I + 1/k \cdot HMH)^{-1} = H^{-1}(H^{-2} + 1/k \cdot M)^{-1}H^{-1},$$

where $(H^{-2} + M/k)$ is an SDD matrix, we can estimate $(I + 1/k \cdot HMH)^{-1}z$ and $(I + 1/k \cdot HMH)^{-1}w$ by combining the SDD Solver with the observation that $\left\|H^{-1}u\right\|_{HM_1H} = \|u\|_{M_1}$, for any vector $u$, and any PSD matrix $M_1$. The procedure is described in Figure 4. We upper bound the error in the above estimation procedure. Observe that the number of non-zero entries in $H^{-2} + M$ is at most $m_M + n$, and multiplying a vector with $H$ takes time $O(n)$. Thus, the time required is dominated by two calls to the SDD solver, giving a total time of $\tilde{O}((m_M + n)\log n \log \frac{1+\|HMH\|}{\varepsilon_1})$.

## 5.3 Error Analysis for ExpRational – Proof of Theorem 13

In this section, we give the steps involved in the proof of Theorem 13. This is the most technically challenging part of our proof, and is crucial for the proof of Theorem 13, and in turn, Theorem 7, which is required to prove our main result, Theorem 1. We restate Theorem 13 here for completeness.

**Input**: An SDD matrix $M$, a diagonal matrix $H$ with positive entries, a unit vector $w$ and a vector $y$.
**Output**: A vector $u$ that approximates $(I + 1/k \cdot \Pi HMH\Pi)^{-1}y$, where $\Pi \stackrel{\text{def}}{=} 1 - ww^\top$.

1. Compute $z \stackrel{\text{def}}{=} y - (w^\top y)w$.
2. Call SDD solver with matrix $(H^{-2} + M/k)$, vector $H^{-1}z$ and error parameter $\frac{\varepsilon_1}{6(1+1/k\cdot\|HMH\|)}$. Denote the vector returned by $\beta_1$.
3. Call SDD solver with matrix $(H^{-2} + M/k)$, vector $H^{-1}w$ and error parameter $\frac{\varepsilon_1}{6(1+1/k\cdot\|HMH\|)}$. Denote the vector returned by $\beta_2$.
4. Compute

$$u_1 \stackrel{\text{def}}{=} H^{-1}\beta_1 - \frac{1/k \cdot w^\top HM\beta_1}{1 + 1/k \cdot w^\top HM\beta_2} H^{-1}\beta_2 + (w^\top y)w. \tag{3}$$

Return $u_1$.

**Figure 4: The $\mathsf{Invert}_A$ procedure for Theorem 7**

THEOREM 16. (Theorem 13 Restated). *Given a symmetric PSD matrix $A \succeq 0$, a vector $v$ with $\|v\| = 1$, an error parameter $0 < \delta \leq 1$ and oracle access to $\mathsf{Invert}_A$, for parameters $k \stackrel{def}{=} O(\log 1/\delta)$ and $\varepsilon_1 \stackrel{def}{=} \exp(-\Theta(k \log k + \log(1+\|A\|)))$, ExpRational computes a vector $u$ such that $\|\exp(-A)v - u\| \leq \delta$, in time $O(T^{inv}_{A,k,\varepsilon_1} \cdot k + n \cdot k^2 + k^3)$, where $T^{inv}_{A,k,\varepsilon_1}$ is the time required by $\mathsf{Invert}_A(\cdot, k, \varepsilon_1)$.*

*Proof Sketch for Theorem 13.* For notational convenience, define $B \stackrel{\text{def}}{=} (I + A/k)^{-1}$. Let $\varepsilon_1 > 0$ and $k$ be the parameters used by ExpRational.

Recall that, in Lemma 4 (Section 5.1), we showed that for the Lanczos procedure, the error in the output vector can be bounded using degree $k$ polynomials. In ExpRational, however, the computation of the vector $Bv_i$, at step $i$, is approximate. Hence, the guarantee given by Lemma 4 for Lanczos does not hold for ExpRational. Nonetheless, we can prove the following analogous lemma for ExpRational.

LEMMA 6. (ExpRational - Polynomial Approximation). *Let $V_k$ and $\widehat{T}_k$ be the matrices generated by ExpRational. Let $f$ be any function such that $f(B)$ and $f(T_k)$ are defined. Define $r_k(x) \stackrel{def}{=} f(x) - p(x)$. Then,*

$$\left\| f(B)v_0 - V_k f(\widehat{T}_k) V_k^\top v_0 \right\| \leq \min_{p \in \Sigma_k} \left( \varepsilon_2 \|p\|_1 + \max_{\lambda \in \Lambda(B)} |r_k(\lambda)| + \max_{\lambda \in \Lambda(\widehat{T}_k)} |r_k(\lambda)| \right), \tag{4}$$

*where $0 < \varepsilon_2 \leq 1$ is such that $\varepsilon_1 \leq \varepsilon_2/(2(k+1)^{3/2})$.*

Thus, we can still bound the error for ExpRational in terms of degree $k$ polynomial approximations to the function $f$. We give a proof sketch for this lemma later in this section. Assuming Lemma 6, we give a proof sketch for Theorem 13.

For our application, $f(t) = f_k(t) \stackrel{\text{def}}{=} \exp\left(k \cdot (1 - 1/t)\right)$, so that $f_k((1 + x/k)^{-1}) = \exp(-x)$. We use the polynomials $p^\star_k$ from the SSV result (Theorem 12) in Lemma 6, and show how to bound each term in Equation (4) to complete the proof of Theorem 13.

We pick,

$$k \stackrel{\text{def}}{=} \max\{k_0, \log_2 8c_1/\delta + 2\log_2 \log_2 8c_1/\delta\} = O\left(\log 1/\delta\right),$$

where $k_0, c_1$ are the constants given by Theorem 12. Theorem 12 implies that $p^\star_k(0) = 0$ and $\sup_{t\in(0,1]} |p^\star_k(t) - f_k(t)| \leq \delta/8$. Since $A \succeq 0$, we get $0 \prec (I + A/k)^{-1} \preceq I$. Thus, $\Lambda(B) \subseteq (0, 1]$, this implies that the second error term in Equation (4) is bounded by $\delta/8$.

To bound the first term in Equation (4), we use the following lemma to bound $\|p^\star_k\|_1$. This lemma is proved by expressing $p^\star_k$ as an interpolation polynomial on $k+1$ equally spaced points in $[0, 1]$.

LEMMA 7. ($\ell_1$-norm Bound). *Given a polynomial $p$ of degree $k$ such that $p(0) = 0$ and*

$$\sup_{t\in(0,1]} \left| e^{-k/t+k} - p(t) \right| = \sup_{x\in[0,\infty)} \left| e^{-x} - p\left((1 + x/k)^{-1}\right) \right| \leq 1,$$

*we must have $\|p\|_1 \leq (2k)^{k+1}$.*

Using this lemma, we deduce that for the value of $k$ chosen, $\|p^\star_k\|_1 \leq (2k)^{k+1}$. This bounds the first term in Equation (4).

In order to bound the third term in Equation (4), we need to give bounds on the spectrum of $\widehat{T}_k$. Due to approximate computation, the eigenvalues of $\widehat{T}_k$ need not lie within $\Lambda(B)$. The following lemma bounds the spectrum of $\widehat{T}_k$, by expressing $\widehat{T}_k$ as $V_k^\top BV_k + E_1$, where $E_1$ is an error matrix. The norm of $E_1$ is bounded by using the guarantee of the $\mathsf{Invert}_A$ procedure.

LEMMA 8. (Eigenvalues of $\widehat{T}_k$). *The eigenvalues of $\widehat{T}_k$ lie in the interval $\left[\lambda_n(B) - \varepsilon_1\sqrt{k+1}, \lambda_1(B) + \varepsilon_1\sqrt{k+1}\right]$.*

Thus, the eigenvalues of $\widehat{T}_k$ do not lie significantly outside $\Lambda(B)$. However, since $f_k(t)$ is discontinuous at $t = 0$, and goes to infinity for small negative values, in order to get a reasonable approximation to $f$, we need that the eigenvalues of $\widehat{T}_k$ are strictly positive. Our choice of $\varepsilon_1$ will ensure that $\varepsilon_1\sqrt{k+1} < \lambda_n(B)$, and hence, $\lambda_n(\widehat{T}_k) > 0$. Thus, from Lemma 8, we get that $\Lambda(\widehat{T}_k) \subseteq (0, \beta]$ for $\beta \stackrel{\text{def}}{=} 1 + \varepsilon_1\sqrt{k+1}$. Hence, we can bound the third term in Equation (4) by $\sup_{t\in(0,\beta]} |r_k(t)|$.

Since $\beta > 1$, we need to bound we the error in approximating $f_k(t)$ by $p^\star_k(t)$ over $(0, \beta]$. The following lemma, gives us the required error bound. This proof for this lemma bounds the error over $[1, \beta]$ by applying triangle inequality and bounding the change in $f_k$ and $p$ over $[1, \beta]$ separately.

LEMMA 9. (Approximation on Extended Interval). *For any $\beta \geq 1$, any degree $k$ polynomial $p$ satisfies,*

$$\sup_{t\in(0,\beta]} |p(t) - f_k(t)| \leq \|p\|_1 \cdot (\beta^k - 1) + (f_k(\beta) - f_k(1)) + \sup_{t\in(0,1]} |p(t) - f_k(t)| .$$

Now, we can use Lemma 9 to bound the third term in Equation (4).

We now know how to bound each term in Equation (4). We plug in the following parameters,

$$\varepsilon_2 \stackrel{\text{def}}{=} \delta/4 \cdot (2k)^{-k-1} \cdot (1+1/k \cdot \lambda_1(A))^{-1},\ \varepsilon_1 \stackrel{\text{def}}{=} 1/8 \cdot (k+1)^{-5/2}\varepsilon_2.$$

Note that these parameters satisfy the condition $\varepsilon_1\sqrt{k+1} < \lambda_n(B)$. After simplifying the expressions in Lemma 9, we get that the total error $\left\| f(B)v_0 - V_k f(\widehat{T}_k)V_k^\top v_0 \right\|$ is bounded by $\delta$, as required by Theorem 13.

*Running Time.*

The running time for the procedure is dominated by $k$ calls to the $\mathsf{Invert}_A$ procedure with parameters $k$ and $\varepsilon_1$, computation of at most $k^2$ dot-products and the exponentiation of $\widehat{T}_k$. The exponentiation of $\widehat{T}_k$ can be done in time $O(k^3)$ [23]. Thus the total running time is $O(T^{\text{inv}}_{A,k,\varepsilon_1} \cdot k + n \cdot k^2 + k^3)$.

This completes the proof sketch for Theorem 13. It remains to give a proof sketch of Lemma 6.

### *Proof Sketch for Lemma 6*

In order to prove Lemma 6, we first bound the error in approximating $p(B)v$, where $p$ is a polynomial of degree at most $k$, by the vector $V_k p(\widehat{T}_k)V_k^\top v_0$. The following lemma achieves this.

LEMMA 10. (Approximate Polynomial Computation). *For any polynomial $p$ of degree at most $k$ and $0 < \varepsilon_2 \leq 1$ satisfying $\varepsilon_1 \leq \varepsilon_2/(2(k+1)^{3/2})$, we have,*

$$\left\| p(B)v_0 - V_k p(\widehat{T}_k)V_k^\top v_0 \right\| \leq \varepsilon_2 \|p\|_1 .$$

This lemma shows that we can bound the error in computation of the polynomial, in terms of the $\ell_1$ norm of the polynomial being computed. This is in contrast with Lemma 4 for the Lanczos method, which states that for computation of polynomials of degree at most $k$, the Lanczos method incurs no error.

Assuming Lemma 10, we can prove Lemma 6 by writing $r_k(x) \stackrel{\text{def}}{=} f(x) - p_k(x)$, where $p_k$ is any degree $k$ approximation to $f(x)$. Thus,

$$f(B)v - V_k f(T_k)V_k^\top v = (p_k(B)v_0 - V_k p_k(\widehat{T}_k)V_k^\top v_0) + r_k(B)v - V_k r_k(T_k)V_k^\top v,$$

*for any $p_k$*. We get Lemma 6 by applying triangle inequality, using Lemma 10, and minimizing over $p_k$.

Lemma 10 follows if we show that, when $p(x) = x^t$ for any $t \leq k$, the error in approximating $p(B)v$ is at most $\varepsilon_2$. The following lemma achieves this.

LEMMA 11. (Approximate Computation with $\widehat{T}_k$). *For any $t \leq k$ and $0 < \varepsilon_2 \leq 1$ satisfying $\varepsilon_1 \leq \varepsilon_2/(8(k+1)^{5/2})$, we have, $\left\| B^t v_0 - V_k \widehat{T}_k^t V_k^\top v_0 \right\| \leq \varepsilon_2$ .*

In order to prove this lemma, we express $BV_k$ in terms of $T_k$ and an error matrix $E$. This allows us to express $BV_k - V_k\widehat{T}_k$ in terms of $E$ and $V_k$. Now, we write the telescoping sum $B^tV_k - V_k\widehat{T}_k^t = \sum_{j=1}^t B^{t-j}(BV_k - V_k\widehat{T}_k)\widehat{T}_k^{j-1}$, and use triangle inequality and a *brute-force* calculation to bound each term. The calculations involved are rather tedious. Please see [20] for details.

## 6. UNIFORM APPROXIMATIONS TO $e^{-x}$

In this section, we give a proof sketch for the upper bound in Theorem 6. The lower bound in Theorem 6 was sketched in detail in Section 3.3. Please see [20] for a discussion about known results and a complete proof of Theorem 6.

We first give a few preliminaries.

*Preliminaries*

A function $g$ is called a $\delta$-approximation to a function $f$ over an interval $\mathcal{I}$, if, $\sup_{x\in\mathcal{I}} |f(x) - g(x)| \leq \delta$. Note that $\mathcal{I}$ can be either finite or infinite. Such approximations are known as *uniform approximations* in approximation theory.

Recall that $\Sigma_k$ denote the set of all degree $k$ polynomials, and, given a degree $k$ polynomial $p \stackrel{\text{def}}{=} \sum_{i=0}^k a_i \cdot x^i$, the $\ell_1$ norm of $p$, is defined as $\|p\|_1 = \sum_{i\geq 0}^k |a_i|$.

### 6.1 Upper Bound in Theorem 6

In this section, we give the lemmas and steps involved in the proof of the upper bound in Theorem 6.

*Proof Sketch for Upper Bound in Theorem 6.* The starting point for this theorem is the result of Saff, Schönhage and Varga [28] (Theorem 12). Recall that it states that, for any positive integer $k$, there exists a degree $k$ polynomial $p_k^\star$ such that, $p_k^\star((1 + x/k)^{-1})$ approximates $e^{-x}$ up to an error of $O(k \cdot 2^{-k})$ over the interval $[0, \infty)$. Thus, it suffices to pick $k = \Theta(\log 1/\delta)$, for the polynomial $p_k^\star\left((1 + x/k)^{-1}\right)$ to approximate $e^{-x}$ up to an error of $\delta/2$.

The main idea is to construct a polynomial $q^\star(x)$ approximating $(1 + x/k)^{-1}$ on the interval $[0, b - a]$. Given such a polynomial $q^\star$, a natural candidate polynomial approximating $e^{-x}$ on $[0, b - a]$ is the polynomial $p_k^\star(q^\star(x))$. Or equivalently, the polynomial $e^{-a} \cdot p_k^\star(q^\star(x - a))$ is a candidate polynomial approximating $e^{-x}$ on $[a, b]$. We can then bound the approximation error as,

$$\begin{aligned}\sup_{x\in[a,b]} &|e^{-a} \cdot p_k^\star(q^\star(x-a)) - e^{-x}| \\ &\leq e^{-a} \cdot \sup_{y\in[0,b-a]} |p_k^\star((1+y/k)^{-1}) - e^{-y}| \\ &\quad + e^{-a} \cdot \sup_{y\in[0,b-a]} |p_k^\star(q^\star(y)) - p_k^\star((1+y/k)^{-1})| \quad (5)\end{aligned}$$

We know that the first term is at most $\delta/2 \cdot e^{-a}$. We just need to bound the second term.

The following lemma gives the required polynomial $q^\star$, using Chebyshev polynomials. For convenience, we define $\nu \stackrel{\text{def}}{=} 1/k$.

LEMMA 12. (Approximating $(1 + \nu x)^{-1}$). *For every $\nu > 0, \varepsilon > 0$ and $d > c \geq 0$, there exists a polynomial $q^\star_{\nu,c,d,\varepsilon}(x)$ of degree $\left\lceil \sqrt{\frac{1+\nu d}{1+\nu c}} \log \frac{2}{\varepsilon} \right\rceil$ such that*

$$\sup_{x\in[c,d]} |(1 + \nu x) \cdot q^\star_{\nu,c,d,\varepsilon}(x) - 1| \leq \varepsilon.$$

In order to bound the second term in Equation (5), we need to bound the error in approximating $p((1+\nu x)^{-1})$ by $p(q^\star(x))$ for polynomials $p$ of small degree. Lemma 12 implies that the expression $(1+\nu x)\cdot q^\star$ is within $1\pm\varepsilon$ on $[c,d]$. If $\varepsilon$ is small, for a small positive integer $t$, $[(1+\nu x)\cdot q^\star]^t$ will be at most $1\pm O(t\varepsilon)$. Combining this observation with the fact that $(1+\nu x)^{-1}\le 1$ for $x\in[c,d]$ with $c>0$, we show the following lemma that proves an error bound to small degree polynomials.

LEMMA 13. (Error in Polynomial). *For all real $\varepsilon>0$, $d>c\ge 0$ and a polynomial $p$ of degree $k$; if $k\varepsilon\le 1$, then,*

$$\sup_{x\in[c,d]} |p(q^\star_{\nu,c,d,\varepsilon}(x)) - p((1+\nu x)^{-1})| \le 2k\varepsilon \left\|p\right\|_1,$$

*where $\nu = 1/k$, and $q^\star_{\nu,c,d,\varepsilon}$ is the polynomial from Lemma 12.*

Thus, we can bound the error in approximating $p((1+\nu x)^{-1})$ by $p(q^\star_{\nu,c,d,\varepsilon}(x))$, for small degree polynomials $p$, in terms of $\left\|p\right\|_1$.

Thus, if we can bound the $\ell_1$ norm of $p^\star_k$, we would be done. Lemma 7 already shows such a bound, and allows us to bound $\left\|p^\star_k\right\|_1$ by $(2k)^{k+1}$. Now, we can complete the proof.

Let $\varepsilon \stackrel{\text{def}}{=} \frac{\delta}{4k(2k)^{k+1}}$. Our candidate polynomial is $p_{a,b,\delta}(x) \stackrel{\text{def}}{=} e^{-a}\cdot p^\star_k\left(q^\star_{\nu,0,b-a,\varepsilon}(x-a)\right)$, where $q^\star_{\nu,0,b-a,\varepsilon}$ is the polynomial of degree $\left\lceil \sqrt{1+\nu(b-a)}\log\frac{2}{\varepsilon}\right\rceil$ given by Lemma 12. Since $\varepsilon k<1$, we can use Lemma 13, and deduce that the second term in Equation (5) is at most $e^{-a}\cdot 2k\varepsilon\left\|p^\star_k\right\|_1 \le e^{-a}\cdot\delta/2$.

The degree of $p_{a,b,\delta}(x)$ is the product of the degrees of $p^\star_k$ and $q^\star_{\nu,0,b-a,\varepsilon}$, which is

$$k\left\lceil\sqrt{1+\nu(b-a)}\log\frac{2}{\varepsilon}\right\rceil = O\left(\sqrt{\max\{\log 1/\delta,(b-a)\}}\cdot(\log 1/\delta)^{3/2}\cdot\log\log 1/\delta\right)$$

This completes a proof sketch for the upper bound in Theorem 6.

# 7. DISCUSSION AND OPEN PROBLEMS

In this paper, using techniques from disparate areas such as random walks, SDPs, numerical linear algebra and approximation theory, we have settled the question of designing an asymptotically optimal $\tilde{O}(m)$ spectral algorithm for BS (Theorem 1) and alongwith provided a simple and practical algorithm (Theorem 2). However, there are several outstanding problems that emerge from our work.

The main remaining open question regarding the design of spectral algorithms for BS is whether it is possible to obtain stronger certificates that no sparse balanced cuts exist, in nearly-linear time. This question is of practical importance in the construction of decompositions of the graph into induced graphs that are near-expanders, in nearly-linear time [35]. OV show that their certificate, which is of the same form as that of BALSEP, is stronger than the certificate of Spielman and Teng [35]. In particular, our certificate can be used to produce decompositions into components that are guaranteed to be subsets of induced expanders in $G$. However, this form of certificate is still much weaker than that given by RLE, which actually outputs an induced expander of large volume.

With regards to approximating the Matrix exponential, a computation which plays an important role in SDP-based algorithms, random walks, numerical linear algebra and quantum computing, settling the hypothesis remains the main open question. Further, as noted earlier, the error analysis plays a crucial role in making Theorem 3 and, hence, Theorem 1 work, but its proof is rather long and difficult. A more illuminating proof of this would be highly desirable. It is possible that some of the proofs here can be simplified if we assume that the procedure $\mathsf{Invert}_A(\cdot,k,\varepsilon_1)$ is linear. This is true for the SDD solver and simplifies some proofs in [33]. However, for Theorem 7, which is the main theorem for our application, we combine the SDD solver with the Sherman-Morrison formula and the resulting procedure is no longer linear.

Another question is to close the gap between the upper and lower bounds on polynomial approximations to $e^{-x}$ over an interval $[a,b]$ in Theorem 6.

# 8. REFERENCES


[1] N. Alon and V. D. Milman. $\lambda_1$, Isoperimetric inequalities for graphs, and superconcentrators. *J. Comb. Theory, Ser. B*, 38(1):73–88, 1985.

[2] R. Andersen, F. R. K. Chung, and K. J. Lang. Local graph partitioning using pagerank vectors. In *FOCS'06: Proc. 47th Ann. IEEE Symp. Foundations of Computer Science*, pages 475–486, 2006.

[3] R. Andersen and K. J. Lang. An algorithm for improving graph partitions. In *SODA'08: Proc. 19th Ann. ACM-SIAM Symp. Discrete Algorithms*, SODA '08, pages 651–660, 2008.

[4] R. Andersen and Y. Peres. Finding sparse cuts locally using evolving sets. In *STOC '09: Proc. 41st Ann. ACM Symp. Theory of Computing*, pages 235–244, 2009.

[5] S. Arora and S. Kale. A combinatorial, primal-dual approach to semidefinite programs. In *STOC '07: Proc. 39th Ann. ACM Symp. Theory of Computing*, pages 227–236, 2007.

[6] S. Arora, S. Rao, and U. Vazirani. Expander flows, geometric embeddings and graph partitioning. In *STOC '04: Proc. 36th Ann. ACM Symp. Theory of Computing*, pages 222–231. ACM, 2004.

[7] R. Bhatia. *Matrix Analysis (Graduate Texts in Mathematics)*. Springer, 1996.

[8] E. W. Cheney. *Introduction to approximation theory*. McGraw-Hill, New York, 1966.

[9] F. R. Chung. *Spectral Graph Theory (CBMS Regional Conference Series in Mathematics, No. 92)*. American Mathematical Society, 1997.

[10] B. L. Ehle. A-stable methods and Padé approximations to the exponential. *Siam J. on Mathematical Analysis*, 4(4):671–680, 1973.

[11] J. v. Eshof and M. Hochbruck. Preconditioning Lanczos approximations to the matrix exponential. *SIAM J. Sci. Comput.*, 27:1438–1457, November 2005.

[12] M. Hochbruck and C. Lubich. On krylov subspace approximations to the matrix exponential operator. *SIAM J. Numer. Anal.*, 34(5):1911–1925, Oct. 1997.

[13] G. Iyengar, D. J. Phillips, and C. Stein. Approximation algorithms for semidefinite packing problems with applications to maxcut and graph coloring. In *IPCO'05: Proc. 11th Conf. Integer*

*Programming and Combinatorial Optimization*, pages 152–166, 2005.

[14] G. Iyengar, D. J. Phillips, and C. Stein. Approximating semidefinite packing programs. *SIAM Journal on Optimization*, 21(1):231–268, 2011.

[15] S. Kale. Efficient algorithms using the multiplicative weights update method. Technical report, Princeton University, Department of Computer Science, 2007.

[16] R. Kannan, S. Vempala, and A. Vetta. On clusterings - good, bad and spectral. In *FOCS'00: Proc. 41th Ann. IEEE Symp. Foundations of Computer Science*, page 367. IEEE Computer Society, 2000.

[17] R. Khandekar, S. Rao, and U. Vazirani. Graph partitioning using single commodity flows. In *STOC '06: Proc. 38th Ann. ACM Symp. Theory of Computing*, pages 385–390. ACM, 2006.

[18] I. Koutis, G. L. Miller, and R. Peng. Approaching optimality for solving SDD linear systems. In *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, FOCS '10, pages 235–244. IEEE Computer Society, 2010.

[19] A. Madry. Fast approximation algorithms for cut-based problems in undirected graphs. In *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, FOCS '10, pages 245–254. IEEE Computer Society, 2010.

[20] L. Orecchia, S. Sachdeva, and N. K. Vishnoi. Approximating the exponential, the lanczos method and an $\tilde{O}(m)$-time spectral algorithm for balanced separator. *CoRR*, abs/1111.1491, 2011.

[21] L. Orecchia, L. J. Schulman, U. V. Vazirani, and N. K. Vishnoi. On partitioning graphs via single commodity flows. In *STOC '08: Proc. 40th Ann. ACM Symp. Theory of Computing*, pages 461–470, 2008.

[22] L. Orecchia and N. K. Vishnoi. Towards an sdp-based approach to spectral methods: A nearly-linear-time algorithm for graph partitioning and decomposition. In *SODA'11: Proc. 22nd Ann. ACM-SIAM Symp. Discrete Algorithms*, pages 532–545, 2011.

[23] V. Y. Pan and Z. Q. Chen. The complexity of the matrix eigenproblem. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, STOC '99, pages 507–516. ACM, 1999.

[24] E. Parzen. *Stochastic Processes*. Society for Industrial and Applied Mathematics, 1999.

[25] H. Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, STOC '08, pages 255–264. ACM, 2008.

[26] Y. Saad. Analysis of some Krylov subspace approximations to the matrix exponential operator. *SIAM J. Numer. Anal.*, 29:209–228, February 1992.

[27] E. Saff. On the degree of best rational approximation to the exponential function. *Journal of Approximation Theory*, 9(2):97 – 101, 1973.

[28] E. B. Saff, A. Schönhage, and R. S. Varga. Geometric convergence to $e^{-z}$ by rational functions with real poles. *Numerische Mathematik*, 25:307–322, 1975.

[29] K. Schloegel, G. Karypis, and V. Kumar. Sourcebook of parallel computing. chapter Graph partitioning for high-performance scientific simulations, pages 491–541. Morgan Kaufmann Publishers Inc., 2003.

[30] J. Sherman. Breaking the multicommodity flow barrier for $O(\sqrt{\log n})$-approximations to sparsest cut. In *FOCS'09: Proc. 50th Ann. IEEE Symp. Foundations of Computer Science*, 2009.

[31] J. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. 1994. `http://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf`.

[32] D. B. Shmoys. Approximation algorithms for NP-hard problems. chapter Cut problems and their application to divide-and-conquer, pages 192–235. PWS Publishing Co., 1997.

[33] D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *STOC '04: Proc. 36th Ann. ACM Symp. Theory of Computing*, pages 81–90. ACM, 2004.

[34] D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *CoRR*, abs/cs/0607105, 2006.

[35] D. A. Spielman and S.-H. Teng. A local clustering algorithm for massive graphs and its application to nearly-linear time graph partitioning. *CoRR*, abs/0809.3232, 2008.

[36] C. Underhill and A. Wragg. Convergence properties of Padé approximants to $\exp(z)$ and their derivatives. *IMA Journal of Applied Mathematics*, 11(3):361–367, 1973.