JMLR: Workshop and Conference Proceedings vol 40:1–27, 2015

Efficient Sampling for Gaussian Graphical Models via Spectral Sparsification

Dehua Cheng[1] DEHUA.CHENG@USC.EDU
Yu Cheng[1] YU.CHENG.1@USC.EDU
Yan Liu[1] YANLIU.CS@USC.EDU
Richard Peng[2] RPENG@MIT.EDU
Shang-Hua Teng[1] SHANGHUA@USC.EDU
[1] *University of Southern California,* [2] *Massachusetts Institute of Technology*

Abstract

Motivated by a sampling problem basic to computational statistical inference, we develop a toolset based on spectral sparsification for a family of fundamental problems involving Gaussian sampling, matrix functionals, and reversible Markov chains. Drawing on the connection between Gaussian graphical models and the recent breakthroughs in spectral graph theory, we give the first nearly linear time algorithm for the following basic matrix problem: Given an $n \times n$ Laplacian matrix $\mathbf{M}$ and a constant $-1 \leq p \leq 1$, provide efficient access to a sparse $n \times n$ linear operator $\tilde{\mathbf{C}}$ such that

$$\mathbf{M}^p \approx \tilde{\mathbf{C}}\tilde{\mathbf{C}}^\top,$$

where $\approx$ denotes spectral similarity. When p is set to -1, this gives the first parallel sampling algorithm that is essentially optimal both in total work and randomness for Gaussian random fields with symmetric diagonally dominant (SDD) precision matrices. It only requires *nearly linear work* and $2n$ *i.i.d. random univariate Gaussian samples* to generate an n-dimensional *i.i.d. Gaussian random sample* in polylogarithmic depth.

The key ingredient of our approach is an integration of spectral sparsification with multilevel method: Our algorithms are based on factoring $\mathbf{M}^p$ into a product of well-conditioned matrices, then introducing powers and replacing dense matrices with sparse approximations. We give two sparsification methods for this approach that may be of independent interest. The first invokes Maclaurin series on the factors, while the second builds on our new nearly linear time spectral sparsification algorithm for random-walk matrix polynomials. We expect these algorithmic advances will also help to strengthen the connection between machine learning and spectral graph theory, two of the most active fields in understanding large data and networks.

Keywords: Gaussian Sampling, Spectral Sparsification, Matrix Polynomials, Random Walks

1. Introduction

Sampling from a multivariate probability distribution is one of the most fundamental problems in statistical machine learning and statistical inference. In the sequential computation framework, the Markov Chain Monte Carlo (MCMC) algorithms (e.g., Gibbs sampling) and their theoretical underpinning built on the celebrated Hammersley-Clifford Theorem have provided the algorithmic and mathematical foundation for analyzing graphical models (Jordan, 1998; Koller and Friedman, 2009). However, unless there are significant independence among the variables, the Gibbs sampling process tends to be highly sequential: Many distributions require a significant number of iterations,

e.g., $\Omega(n)$, for a reliable estimate (Jordan, 1998; Liu et al., 2013). Gibbs sampling may also use $\Omega(n^2)$ random numbers (n per iteration) to generate its first sample, although in practice it usually performs better than these worst case bounds.

Despite active efforts by various research groups (Gonzalez et al., 2011; Johnson et al., 2013; Ahn et al., 2014; Ihler et al., 2003), the design of a *provably* scalable parallel Gibbs sampling algorithm remains a challenging problem. The "holy grail" question in parallel sampling for graphical models can be characterized as:

> *Is it possible to obtain a characterization of the family of Markov random fields from which one can draw a sample in polylogarithmic depth with nearly-linear total work?*

Formally, the performance of a parallel sampling algorithm can be characterized using the following three parameters: (1) *Time complexity (work)*: the total amount of operations needed by the sampling algorithm to generate the first and subsequent random samples. (2) *Parallel complexity (depth)*: the length of the longest critical path of the algorithm in generating the first and subsequent random samples. (3) *Randomness complexity*: total number of random scalars needed by the sampling algorithm to generate a random sample. In view of the holy grail question, we say a sampling algorithm is *efficient* if it runs in nearly linear time, polylogarithmic depth, and uses only $O(n)$ random scalars to generate a random sample from the Markov random fields with n variables.

GAUSSIAN RANDOM FIELDS WITH SDD PRECISION MATRICES

The basic Gibbs sampling method follows a randomized process that iteratively resamples each variable conditioned on the value of other variables. This method is analogous to the Gauss-Seidel method for solving linear systems, and this connection is most apparent on an important class of multivariate probability distributions: *multivariate normal distributions* or *multivariate Gaussian distributions*. In the framework of graphical models, such a distribution is commonly specified by its *precision matrix* $\mathbf{\Lambda}$ and potential vector $\boldsymbol{h}$, i.e., $\Pr(\mathbf{x}|\mathbf{\Lambda}, \boldsymbol{h}) \propto \exp(-\frac{1}{2}\mathbf{x}^\top \mathbf{\Lambda}\mathbf{x} + \boldsymbol{h}^\top \mathbf{x})$, which defines a *Gaussian random field*.

Our study is partly motivated by the recent work of Johnson, Saunderson, and Willsky (2013) that provides a mathematical characterization of the *Hogwild Gaussian Gibbs sampling* heuristics (see also Niu et al., 2011). Johnson *et al.* proved that if the precision matrix $\mathbf{\Lambda}$ is *symmetric generalized diagonally dominant*, aka an *H-matrix*, then Hogwild Gibbs sampling converges with the correct mean for any variable partition (among parallel processors) and for any number of locally sequential Gibbs sampling steps with old statistics from other processors. The SDD condition defines a natural subclass of Gaussian random fields which arises in applications such as image denoising (Bishop, 2006). It can also be used in the rejection/importance sampling framework (Andrieu et al., 2003) as a proposed distribution, likely giving gains over the commonly used diagonal precision matrices in this framework.

This connection to H-matrices leads us to draw from the recent breakthroughs in nearly-linear work solvers for linear systems in symmetric diagonally dominant matrices (Spielman and Teng, 2004; Koutis et al., 2010; Kelner et al., 2013; Peng and Spielman, 2014), which are algorithmically inter-reducible to H-matrices (see Daitch and Spielman, 2008). We focus our algorithmic solution on a subfamily of H-matrices known as the *SDDM* matrices; these are SDD matrices with non-positive off-diagonal entries. Our sampling result can be simply extended to Gaussian random field with SDD precision matrices (see Appendix E).

Our algorithm is based on the following three-step numerical framework for sampling a Gaussian random field with parameters $(\mathbf{\Lambda}, \boldsymbol{h})$: $\Pr(\mathbf{x}|\mathbf{\Lambda}, \boldsymbol{h}) \propto \exp(-\frac{1}{2}\mathbf{x}^\top \mathbf{\Lambda}\mathbf{x} + \boldsymbol{h}^\top \mathbf{x})$. (1) Compute

the mean $\boldsymbol{\mu}$ of the distribution from $(\boldsymbol{\Lambda}, \boldsymbol{h})$ by solving the linear system $\boldsymbol{\Lambda}\boldsymbol{\mu} = \boldsymbol{h}$. (2) Compute a factor of the covariance matrix by finding an inverse square-root factor of the precision matrix $\boldsymbol{\Lambda}$, i.e., $\boldsymbol{\Lambda}^{-1} = \mathbf{C}\mathbf{C}^\top$. (3) If $\mathbf{C}$ is an $n \times n'$ matrix, then we can obtain a sample of $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Lambda})$ by first drawing a random vector $\boldsymbol{z} = (z_1, z_2, \ldots, z_{n'})^\top$, where $z_i, i = 1, 2, \ldots, n'$ are i.i.d. standard Gaussian random variables, and return $\boldsymbol{x} = \mathbf{C}\boldsymbol{z} + \boldsymbol{\mu}$.

If $\mathbf{C}$ has a sparse representation, then Step 3 has a scalable parallel implementation with parallel complexity polylogarithmic in n. Polylogarithmic depth and nearly-linear work solvers for SDDM linear systems (Peng and Spielman, 2014) also imply a scalable parallel solution to Step 1. So this framework allows us to concentrate on Step 2: factoring the covariance matrix $\boldsymbol{\Lambda}$.

In fact, because each SDD matrix has a graph-theoretical factor analog of the *signed edge-vertex incidence matrix*, we can compose this factor with the inverse of $\boldsymbol{\Lambda}$ provided by the solver to obtain a *semi-efficient* sampling algorithm for Gaussian field with SDD precision matrix (See Appendix G). While the resulting algorithm meets the work and depth objectives we set for efficient sampling, the algorithm is inefficient in randomness complexity: it generates a sample of $(\boldsymbol{\Lambda}, \boldsymbol{h})$ by first taking random univariate Gaussian samples, one per non-zero entry in $\boldsymbol{\Lambda}$, which could be higher than the existing numerical approach to this problem (Chow and Saad, 2014).

The classic Newton's method gives another *semi-efficient* sampling algorithm that is efficient in both depth and randomness complexity, but not in total work. This is because direct use of Newton's method to compute $\boldsymbol{\Lambda}^{-1/2}$ may lead to dense matrix multiplication even when the original graphical model is sparse.

OUR CONTRIBUTION

The goal of achieving optimal randomness leads us to the question whether algorithms for SDDM matrices can be improved to produce a *square* $n \times n$ factorization, i.e., a *"canonical" square-root*, of the precision matrix. This in turn leads us to a more general and fundamental problem in numerical algorithms, particularly in the analysis of Markov models (Leang Shieh et al., 1986; Higham and Lin, 2011) – the problem of computing q^{th} root of a matrix for some $q \geq 1$. For example, in Nick Higham's talk for Brain Davies' 65 Birthday conference (2009), he quoted an email from a power company regarding the usage of an electricity network to illustrate the practical needs of taking the q^{th}-root of a Markov transition.

> *"I have an Excel spreadsheet containing the transition matrix of how a company's [Standard & Poor's] credit rating charges from on year to the next. I'd like to be working in eighths of a year, so the aim is to find the* **eighth root of the matrix.***"*

However, for some sparse SDDM matrix $\mathbf{M}$, its q^{th}-root could be dense. Our approaches are crucially based on factorizing $\mathbf{M}^{1/q}$ into a product of well-conditioned, easy-to-evaluate, sparse matrices, called a *sparse factor chain*. These are matrices generated via sparsification (Spielman and Teng, 2011), which is a general tool that takes a dense SDD matrix $\mathbf{B}$ and error bound ϵ, generates $\tilde{\mathbf{B}} \approx_\epsilon \mathbf{B}$ ($\approx_\epsilon$ is defined in Section 2) with $O(n \log n/\epsilon^2)$ entries. Our result is as follows:

Theorem 1 (Sparse Factor Chain for $\mathbf{M}^p$) *For any $n \times n$ SDDM matrix $\mathbf{M}$ with m non-zero entries and condition number κ [1], for any constant $-1 \leq p \leq 1$ and precision parameter $0 < \epsilon \leq 1$, we can construct a sparse factor chain of $\mathbf{M}^p$ (with precision ϵ) in $O(m \cdot \log^{c_1} n \cdot \log^{c_2}(\kappa/\epsilon) \cdot \epsilon^{-4})$*

1. As shown in Spielman and Teng (2014) (Lemma 6.1), for an SDDM matrix M, $\kappa(\mathbf{M})$ is always upper bounded by $\text{poly}(n)/\epsilon_M$, where ϵ_M denotes the machine precision. Note also, the running time of any numerical algorithm for factoring $\mathbf{M}^{-1}$ is $\Omega(m + \log(\kappa(\mathbf{M})/\epsilon))$ where $\Omega(\log(\kappa(\mathbf{M})/\epsilon))$ is the bit precision required in the worst case.

work for modest[2] *constants* c_1 *and* c_2*. Moreover, our algorithm can be implemented to run in polylogarithmic depth on a parallel machine with* m *processors.*

Prior to our work, no algorithm could provide such a factorization in nearly-linear time. As factoring an SDD matrix can be reduced to factoring an SDDM matrix twice its dimensions (See Appendix E), our efficient sampling algorithm can be directly extended to Gaussian random fields with SDD precision matrices. Moreover, for the case when $p = -1$ (i.e., the factorization problem needed for Gaussian sampling), we can remove the polynomial dependency of ϵ, which leads to the complexity of $O(m \cdot \log^{c_1} n \cdot \log^{c_2}(\kappa) \cdot \log(1/\epsilon))$.

Putting the running time aside, we would like to point out the subtle difference between the quality of our sampling algorithm and the quality of classic Gibbs method as both generators have their inherent approximation errors: The samples generated by our algorithm are *i.i.d.*, but they are from a Gaussian random field whose covariance matrix approximates the one given in the input. Fortunately, with $\log(1/\epsilon)$ dependence on the approximation parameter ϵ, we can essentially set ϵ to machine precision. In contrast, samples generated from Gibbs process are only approximately *i.i.d.* In addition, while the Gibbs process gives the exact mean and covariance in the limit, it needs to produce a sample in a finite steps, which introduces errors to both mean and covariance.

Our approaches work upon a representation of $\mathbf{M}$ as $\mathbf{I} - \mathbf{X}$, which is known as splitting in numerical analysis. This splitting allows the interpretation of $\mathbf{X}$ as a random walk transition probability matrix. The key step is to reduce the factorization problem for $\mathbf{I} - \mathbf{X}$ to one on $\mathbf{I} - f(\mathbf{X})$, where f is a low degree polynomial. This framework is closely related to multi-scale analysis of graphs (see e.g., Coifman et al., 2005), which relates $\mathbf{I} - \mathbf{X}^2$ to $\mathbf{I} - \mathbf{X}$. This transformation also plays a crucial role in the parallel solver for SDDM linear systems by Peng and Spielman (2014), which relied on an efficient sparsification routine for $\mathbf{I} - \mathbf{X}^2$ to keep all intermediate matrices sparse.

It can be checked that matrix sparsification extends naturally to a wide range of functionals that include p^{th} powers. Meanwhile, the need to produce a factorization means only products can be used in the reduction step. We obtain such a factorization in two ways: a more involved reduction step so that we can use simpler f, or basing factorization on more intricate functions f. The former uses the Maclaurin series for the reduction. The latter is effective when $1/p$ is an integer; it only performs matrix-vector multiplications for the reduction steps, but reduces to a more general matrices, e.g., $\mathbf{I} - \frac{3}{4}\mathbf{X}^2 - \frac{1}{4}\mathbf{X}^3$ when $p = -1$. Incorporating these higher degree polynomials leads to a basic problem that is of interest on its own: whether a random-walk polynomial can be efficiently sparsified. Suppose $\mathbf{A}$ and $\mathbf{D}$ are respectively the adjacency matrix and diagonal weighted degree matrix of a weighted undirected graph G. For a non-negative vector $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_t)$ with $\sum_{r=1}^{t} \alpha_r = 1$, the matrix

$$\mathbf{L}_{\boldsymbol{\alpha}}(G) = \mathbf{D}\left(\mathbf{I} - \sum_{r=1}^{t} \alpha_r \cdot \left(\mathbf{D}^{-1}\mathbf{A}\right)^r\right) \tag{1}$$

generalizes graph Laplacian to one involving a t-degree *random-walk matrix polynomial* of G.

In order to effectively utilize random-walk matrix polynomials in our algorithms, we give the first nearly linear time spectral sparsification algorithm for them. Our sparsification algorithm is built on the following key mathematical observation: One can obtain good and succinctly representable sampling probabilities for $\mathbf{I} - \mathbf{X}^r$ via $\mathbf{I} - \mathbf{X}$ and $\mathbf{I} - \mathbf{X}^2$. This allows us to design an

2. Currently, both constants are at most 5 – these constants and the exponent 4 for $(1/\epsilon)$ can be further reduced with the improvement of spectral sparsification algorithms (Spielman and Teng, 2011; Spielman and Srivastava, 2011; Koutis, 2014; Miller et al., 2013) and tighter numerical analysis of our approaches.

efficient path sampling algorithm which generalizes the works of Spielman and Teng (2011); Spielman and Srivastava (2011); Peng and Spielman (2014).

Theorem 2 (Sparsification of Random-Walk Matrix Polynomials) *For any weighted undirected graph G with n nodes and m edges, for every non-negative vector $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_t)$ with $\sum_{r=1}^{t} \alpha_r = 1$, for any $\epsilon > 0$, we can construct in time $O(t^2 \cdot m \cdot \log^2 n/\epsilon^2)$ a spectral sparsifier with $O(n \log n/\epsilon^2)$ non-zeros and approximation parameter ϵ for the random-walk matrix polynomial $\mathbf{L}_{\boldsymbol{\alpha}}(G)$.*

The application of our results to sampling is a step toward understanding the holy grail question in parallel sampling for graphical models. It provides a natural and non-trivial sufficient condition for Gaussian random fields that leads to parallel sampling algorithms that run in polylogarithmic parallel time, nearly-linear total work, and have optimal randomness complexity. While we are mindful that many classes of precision matrices are not reducible to SDD matrices, we believe our approach is applicable to a wider range of matrices. The key structural property that we use is that matrices related to the square of the off-diagonal entries of SDDM matrices have sparse representations. We believe incorporating such structural properties is crucial for overcoming the $\Omega(mn)$-barrier for sampling from Gaussian random fields, and that our construction of the sparse factor chain may lead to efficient parallel sampling algorithms for other popular graphical models. We hope the nature of our algorithms also help to illustrate the connection between machine learning and spectral graph theory, two active fields in understanding large data and networks.

2. Background and Notation

In this section, we introduce the definitions and notations that we will use in this paper. We use $\rho(\mathbf{M})$ to denote the spectral radius of the matrix $\mathbf{M}$, which is the maximum absolute value of the eigenvalues of $\mathbf{M}$. We use $\kappa(\mathbf{M})$ to denote its condition number corresponding to the spectral norm, which is the ratio between the largest and smallest singular values of $\mathbf{M}$.

We work with SDDM matrices in our algorithms, and denote them by $\mathbf{M}$. SDDM matrices are positive-definite SDD matrices with nonnegative off-diagonal entries. The equivalence between solving linear systems in such matrices and graph Laplacians, weakly-SDD matrices, and M-matrices are well known (see Spielman and Teng, 2004; Daitch and Spielman, 2008). In Appendix E, we rigorously prove that this connection carries over to inverse factors.

SDDM matrices can be split into diagonal and off-diagonal entries. We will show in Appendix F that by allowing diagonal entries in both matrices of this splitting, we can ensure that the diagonal matrix is $\mathbf{I}$ without loss of generality.

Lemma 3 (Commutative Splitting) *For an SDDM matrix $\mathbf{M} = \mathbf{D} - \mathbf{A}$ with diagonal $\mathbf{D} = diag\,(d_1, d_2, \ldots, d_n)$, and let $\kappa = \max\{2, \kappa\,(\mathbf{D} - \mathbf{A})\}$. We can pick $c = (1 - 1/\kappa)\,/(\max_i d_i)$, so that all eigenvalues of $c\,(\mathbf{D} - \mathbf{A})$ lie between $\frac{1}{2\kappa}$ and $2 - \frac{1}{2\kappa}$. Moreover, we can rewrite $c\,(\mathbf{D} - \mathbf{A})$ as an SDDM matrix $\mathbf{I} - \mathbf{X}$, where all entries of $\mathbf{X} = \mathbf{I} - c\,(\mathbf{D} - \mathbf{A})$ are nonnegative, and $\rho\,(\mathbf{X}) \leq 1 - \frac{1}{2\kappa}$.*

Since all of our approximation guarantees are multiplicative, we can ignore the constant scaling factor, and use the splitting $\mathbf{M} = \mathbf{I} - \mathbf{X}$ for the rest of our analysis.

For a symmetric positive definite matrix $\mathbf{M}$ with spectral decomposition $\mathbf{M} = \sum_{i=1}^{n} \lambda_i \boldsymbol{u}_i \boldsymbol{u}_i^\top$, its p^{th} power for any $p \in \mathbb{R}$ is: $\mathbf{M}^p = \sum_{i=1}^{n} \lambda_i^p \boldsymbol{u}_i \boldsymbol{u}_i^\top$. We say that $\mathbf{C}$ is an *inverse square-root factor*

$\tilde{G} = \text{GRAPHSAMPLING}(G = \{V, E, w\}, \tau_e, M)$

1. Initialize graph $\tilde{G} = \{V, \emptyset\}$.
2. For i from 1 to M:
   Sample an edge e from E with $p_e = \tau_e / (\sum_{e \in E} \tau_e)$. Add e to $\tilde{G}$ with weight $w(e)/(M\tau_e)$.
3. Return graph $\tilde{G}$.

Figure 1: Pseudocode for Sampling by Effective Resistances

of $\mathbf{M}$, if $\mathbf{C}\mathbf{C}^\top = \mathbf{M}^{-1}$. When the context is clear, we sometime refer to $\mathbf{C}$ as an inverse factor of $\mathbf{M}$. Note that the inverse factor of a matrix is not unique.

In our analysis, we will make extensive use of *spectral approximation*. We say $\mathbf{X} \approx_\epsilon \mathbf{Y}$ when

$$\exp(\epsilon)\,\mathbf{X} \succcurlyeq \mathbf{Y} \succcurlyeq \exp(-\epsilon)\,\mathbf{X}, \tag{2}$$

where the *Loewner ordering* $\mathbf{Y} \succcurlyeq \mathbf{X}$ means $\mathbf{Y} - \mathbf{X}$ is positive semi-definite. We use the following standard facts about the Loewner positive definite order and approximation.

Fact 4 *For positive semi-definite matrices* $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{W}$ *and* $\mathbf{Z}$,

a. if $\mathbf{Y} \approx_\epsilon \mathbf{Z}$, *then* $\mathbf{X} + \mathbf{Y} \approx_\epsilon \mathbf{X} + \mathbf{Z}$;

b. if $\mathbf{X} \approx_\epsilon \mathbf{Y}$ *and* $\mathbf{W} \approx_\epsilon \mathbf{Z}$, *then* $\mathbf{X} + \mathbf{W} \approx_\epsilon \mathbf{Y} + \mathbf{Z}$;

c. if $\mathbf{X} \approx_{\epsilon_1} \mathbf{Y}$ *and* $\mathbf{Y} \approx_{\epsilon_2} \mathbf{Z}$, *then* $\mathbf{X} \approx_{\epsilon_1+\epsilon_2} \mathbf{Z}$;

d. if $\mathbf{X}$ *and* $\mathbf{Y}$ *are positive definite matrices such that* $\mathbf{X} \approx_\epsilon \mathbf{Y}$, *then* $\mathbf{X}^{-1} \approx_\epsilon \mathbf{Y}^{-1}$;

e. if $\mathbf{X} \approx_\epsilon \mathbf{Y}$ *and* $\mathbf{V}$ *is a matrix, then* $\mathbf{V}^\top \mathbf{X} \mathbf{V} \approx_\epsilon \mathbf{V}^\top \mathbf{Y} \mathbf{V}$.

The Laplacian matrix is related to electrical flow (Spielman and Srivastava, 2011; Christiano et al., 2011). For an edge with weight $w(e)$, we view it as a resistor with resistance $r(e) = 1/w(e)$. Recall that the *effective resistance* between two vertices u and v $R(u, v)$ is defined as the potential difference induced between them when a unit current is injected at one and extracted at the other. Let $\boldsymbol{e}_i$ denote the vector with 1 in the i-th entry and 0 everywhere else, the effective resistance $R(u, v)$ equals to $(\boldsymbol{e}_u - \boldsymbol{e}_v)^\top \mathbf{L}^\dagger (\boldsymbol{e}_u - \boldsymbol{e}_v)$, where $\mathbf{L}^\dagger$ is the *Moore-Penrose pseudoinverse* of $\mathbf{L}$. From this expression, we can see that effective resistance obeys triangle inequality. Also note that adding edges to a graph does not increase the effective resistance between any pair of nodes.

Spielman and Srivastava (2011) showed that oversampling the edges using upper bounds on the effective resistance suffices for constructing spectral sparsifiers. The theoretical guarantees for this sampling process were strengthened in Kelner and Levin (2013). A pseudocode of this algorithm is given in Figure 1. The following theorem provides a guarantee on its performance:

Theorem 5 *Given a weighted undirected graph* G, *and upper bounds on its effective resistance* $Z(e) \geq R(e)$. *For any approximation parameter* $\epsilon > 0$, *there exists* $M = O((\sum_{e \in E} \tau_e) \cdot \log n / \epsilon^2)$, *with* $\tau_e = w(e)Z(e)$, *such that with probability at least* $1 - \frac{1}{n}$, $\tilde{G} = \text{GRAPHSAMPLING}(G, w, \tau_e, M)$ *has at most* M *edges, and satisfies*

$$(1 - \epsilon)\mathbf{L}_G \preccurlyeq \mathbf{L}_{\tilde{G}} \preccurlyeq (1 + \epsilon)\mathbf{L}_G. \tag{3}$$

3. Symmetric Factor Reduction: Overview

We start with a brief outline of our approach which utilizes the following symmetric factor-reduction scheme: In order to construct a sparse factor chain for $\mathbf{M}^p = (\mathbf{I} - \mathbf{X})^p$, ideally we would like to design two matrix functions f and g so that we can iteratively generate a sequence of matrices $\mathbf{X}_0, \mathbf{X}_1, ..., \mathbf{X}_d$ where $\mathbf{X}_0 = \mathbf{X}$ and $\mathbf{X}_{k+1} = f(\mathbf{X}_k)$ that satisfies a symmetric reduction formula:

$$(\mathbf{I} - \mathbf{X}_k)^p = g(\mathbf{X}_k) \cdot (\mathbf{I} - f(\mathbf{X}_k))^p \cdot g(\mathbf{X}_k). \tag{4}$$

To obtain an *efficient* iterative algorithm, it suffices to construct f and g such that (1) the spectral radius of $\mathbf{X}_{k+1}$ is reducing, i.e., $\rho(\mathbf{X}_{k+1})/\rho(\mathbf{X}_k) \leq \alpha$ for a parameter $\alpha < 1$ that depends on $1 - \rho(\mathbf{X}_0)$; (2) the function $g(\mathbf{X}_k)$ has a sparse representation whose matrix-vector product can be efficiently evaluated; (3) $\mathbf{I} - f(\mathbf{X}_k)$ itself is sparse.

Condition (1) guarantees that after a logarithmic number of iterations, $\rho(\mathbf{X}_d)$ approaches 0, and thus $(\mathbf{I} - \mathbf{X}_d)^p$ can be approximated by $\mathbf{I}$, while conditions (2) and (3) guarantee that no densification occurs during the iterative reduction so that they can be performed in nearly linear time. Unfortunately, it is difficult to obtain all three of these conditions simultaneously. We remedy this by introducing sparsification to the middle term $\mathbf{I} - f(\mathbf{X}_k)$ and set $\mathbf{X}_{k+1} \approx f(\mathbf{X}_k)$. This is where numerical approximation occurs. Note that this replacement works under p^{th} power due to Fact 4.e on spectral approximation and Lemma 14, which states that spectral approximation is preserved under the p^{th} power for $p \in [-1, 1]$. This allows us to replace the middle term by its spectral sparsifier and maintain a multiplicative overall error.

To utilizing this idea for our purpose, we need to make strategic tradeoffs between the complexity of $g(\mathbf{X}_k)$ and the complexity for maintaining a sparse form for $\mathbf{I} - \mathbf{X}_{k+1}$. If g is limited to linear functions, then intuitively $\mathbf{I} - f(\mathbf{X}_k)$ has to make up for the accuracy, making sparsification more challenging. Alternatively, by allowing g to be non-linear, we have more freedom in designing f.

In Section 4, we limit f to be a quadratic function of $\mathbf{X}_k$ and use the sparsification algorithm from Peng and Spielman (2014) for the middle term. We need to approximate the corresponding non-linear matrix function g by a sparse representation. For this approach to work, we also need to ensure that the approximator of g commutes with $\mathbf{X}_k$, in order to bound the overall multiplicative error (See Appendix B). We show that the matrix version of Maclaurin Series for well-conditioned matrices, i.e., a polynomial of $\mathbf{X}_k$, can be used to seek a desirable approximation of g.

In Section 5, we limit g to be linear, the same form as in the classic Newton's method. This leads to a problem of sparsifying a random-walk matrix polynomial of degree 3 or higher. We prove Theorem 2 in this section, which provides the technical subroutine for this approach.

4. Symmetric Factor Reduction via Maclaurin Series

To illustrate our algorithm, we start with the case $p = -1$, i.e., inverse square root factor. One candidate expression for reducing from $(\mathbf{I} - \mathbf{X})^{-1}$ to $(\mathbf{I} - \mathbf{X}^2)^{-1}$ is the identity

$$(\mathbf{I} - \mathbf{X})^{-1} = (\mathbf{I} + \mathbf{X}) \left(\mathbf{I} - \mathbf{X}^2\right)^{-1}. \tag{5}$$

It reduces computing $(\mathbf{I} - \mathbf{X})^{-1}$ to a matrix-vector multiplication and computing $(\mathbf{I} - \mathbf{X}^2)^{-1}$. As $\|\mathbf{X}^2\|_2 < \|\mathbf{X}\|_2$ when $\|\mathbf{X}\|_2 < 1$, $\mathbf{I} - \mathbf{X}^2$ is closer to $\mathbf{I}$ than $\mathbf{I} - \mathbf{X}$. Formally, it can be shown that after $\log \kappa$ steps, where $\kappa = \kappa(\mathbf{I} - \mathbf{X})$, the problem becomes well-conditioned. This low iteration count coupled with the low cost of matrix-vector multiplications then gives a low-depth algorithm.

To perform the symmetric reduction as required in Equation (4), we directly symmetrize the identity in Equation (5) by moving a half power of the first term onto the other side, based on the fact that polynomials of $\mathbf{I}$ and $\mathbf{X}$ commute with each other:

$$(\mathbf{I}-\mathbf{X})^{-1} = (\mathbf{I}+\mathbf{X})^{1/2}\left(\mathbf{I}-\mathbf{X}^2\right)^{-1}(\mathbf{I}+\mathbf{X})^{1/2}. \tag{6}$$

This has the additional advantage that the terms can be naturally incorporated into the factorization $\tilde{\mathbf{C}}\tilde{\mathbf{C}}^\top$. Of course, this leads to the issue of evaluating half powers, or more generally p^{th} powers. Here our key idea, which we will discuss in depth in Appendix B, is to use *Maclaurin expansions*. These are low-degree polynomials which give high quality approximations when the matrices are well-conditioned. We show that for any well-conditioned matrix $\mathbf{B}$, i.e., $\mathbf{B} \approx_{O(1)} \mathbf{I}$, and for any constant power $|p| \leq 1$, there exists a (polylogarithmic) degree t polynomial $T_{p,t}(\cdot)$ such that $T_{p,t}(\mathbf{B}) \approx_\epsilon \mathbf{B}^p$. In such case, the Maclaurin expansion provides a good sparse approximation.

The condition that $\rho(\mathbf{X}) < 1$ means the expression $\mathbf{I}+\mathbf{X}$ is almost well-conditioned: the only problematic case is when $\mathbf{X}$ has an eigenvalue close to -1. We resolve this issue by halving the coefficient in front of $\mathbf{X}$, leading to the following formula for symmetric factorization:

$$(\mathbf{I}-\mathbf{X})^{-1} = (\mathbf{I}+\mathbf{X}/2)^{1/2}\left(\mathbf{I}-\mathbf{X}/2-\mathbf{X}^2/2\right)^{-1}(\mathbf{I}+\mathbf{X}/2)^{1/2}, \tag{7}$$

where $g(\mathbf{X}) = (\mathbf{I}+\mathbf{X}/2)^{1/2}$ can be ϵ-approximated by $T_{1/2,t}(\mathbf{I}+\mathbf{X}/2)$ with $t = O(\log(1/\epsilon))$. We defer the error analysis in Appendix A (Lemma 18).

Equation (7) allows us to reduce the problem to one involving $\mathbf{I} - \frac{1}{2}\mathbf{X} - \frac{1}{2}\mathbf{X}^2$, which is the average of $\mathbf{I}-\mathbf{X}$ and $\mathbf{I}-\mathbf{X}^2$. Note that $\mathbf{X}^2$ is usually a dense matrix. Thus we need to sparsify it during intermediate steps. By Peng and Spielman (2014), $\mathbf{I}-\mathbf{X}^2$ is an SDDM matrix that corresponds to two-step random walks on $\mathbf{I}-\mathbf{X}$, and can be efficiently sparsified, which provides us with a sparsifier for $\mathbf{I} - \frac{1}{2}\mathbf{X} - \frac{1}{2}\mathbf{X}^2$. The application of this sparsification step inside Equation (7) leads to a chain of matrices akin to the sparse inverse chain defined in Peng and Spielman (2014). In Appendix A (Lemma 16) we prove the existence of the following *sparse inverse factor chain*.

Definition 6 (Sparse Inverse Factor Chain) *For a sequence of approximation parameters $\boldsymbol{\epsilon} = (\epsilon_0, \ldots, \epsilon_d)$, an $\boldsymbol{\epsilon}$-sparse factor chain $(\mathbf{X}_1, \ldots, \mathbf{X}_d)$ for $\mathbf{M} = \mathbf{I} - \mathbf{X}_0$ satisfies the following conditions:*

1. *For $i = 0, 1, 2, \ldots, d-1$, $\mathbf{I} - \mathbf{X}_{i+1} \approx_{\epsilon_i} \mathbf{I} - \frac{1}{2}\mathbf{X}_i - \frac{1}{2}\mathbf{X}_i^2$;*
2. *$\mathbf{I} \approx_{\epsilon_d} \mathbf{I} - \mathbf{X}_d$.*

We summarize our first technical lemma below. In this lemma, and the theorems that follow, we will assume our SDDM matrix $\mathbf{M}$ is $n \times n$ and has m nonzero entries. We also use κ to denote $\kappa(\mathbf{M})$, and assume that $\mathbf{M}$ is expressed in the form $\mathbf{M} = \mathbf{I} - \mathbf{X}_0$ for a non-negative matrix $\mathbf{X}_0$.

Lemma 7 (Efficient Chain Construction) *Given an SDDM matrix $\mathbf{M} = \mathbf{I} - \mathbf{X}_0$, and approximation parameter ϵ, there exists an $\boldsymbol{\epsilon}$-sparse factor chain $(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_d)$ for $\mathbf{M}$ with $d = O(\log(\kappa/\epsilon))$ such that (1) $\sum_{i=0}^{d} \epsilon_i \leq \epsilon$, and (2) the total number of nonzero entries in $(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_d)$ is upper bounded by $O(n \log^c n \cdot \log^3(\kappa/\epsilon)/\epsilon^2)$, for a constant c.*

Moreover, we can construct such an $\boldsymbol{\epsilon}$-sparse factor chain in work $O(m \cdot \log^{c_1} n \cdot \log^{c_2}(\kappa/\epsilon)/\epsilon^4)$ and parallel depth $O(\log^{c_3} n \cdot \log(\kappa/\epsilon))$, for some constants c_1, c_2 and c_3.

The length of the chain is logarithmic in both the condition number κ and $\sum \epsilon_i$. We will analyze the error propagation along this chain in Appendix A. Using this Maclaurin based chain, we obtain our main algorithmic result for computing the inverse square root. In its application to Gaussian sampling, we can multiply a random vector $\boldsymbol{x}$ containing i.i.d. standard Gaussian random variable through the chain to obtain a random sample $\boldsymbol{y}$:

$$\boldsymbol{y} = T_{1/2,t}\left(\mathbf{I} + \mathbf{X}_0/2\right) \cdot T_{1/2,t}\left(\mathbf{I} + \mathbf{X}_1/2\right) \cdots T_{1/2,t}\left(\mathbf{I} + \mathbf{X}_{d-1}/2\right) \boldsymbol{x}. \tag{8}$$

Theorem 8 (Maclaurin Series Based Parallel Inverse Factorization) *Given an SDDM matrix* $\mathbf{M} = \mathbf{I} - \mathbf{X}_0$ *and a precision parameter* ϵ*, we can construct a representation of an* $n \times n$ *matrix* $\tilde{\mathbf{C}}$*, such that* $\tilde{\mathbf{C}}\tilde{\mathbf{C}}^\top \approx_\epsilon (\mathbf{I} - \mathbf{X}_0)^{-1}$ *in work* $O(m \cdot \log^{c_1} n \cdot \log^{c_2} \kappa)$ *and parallel depth* $O(\log^{c_3} n \cdot \log \kappa)$ *for some constants* c_1, c_2*, and* c_3*.*

Moreover, for any $\boldsymbol{x} \in \mathbb{R}^n$*, we can compute* $\tilde{\mathbf{C}}\boldsymbol{x}$ *in work* $O((m + n \cdot \log^c n \cdot \log^3 \kappa) \cdot \log\log\kappa \cdot \log(1/\epsilon))$ *and depth* $O(\log n \cdot \log\kappa \cdot \log\log\kappa \cdot \log(1/\epsilon))$ *for some other constant* c*.*

Proof For computing the inverse square root factor ($p = -1$), we first use Lemma 7 with $\sum \epsilon_i = 1$ to construct a crude sparse inverse factor chain with length $d = O(\log\kappa)$. Then for each i between 0 and $d-1$, we use Maclaurin expansions from Lemma 24 to obtain $O(\log\log\kappa)$-degree polynomials that approximate $(\mathbf{I} + \frac{1}{2}\mathbf{X}_i)^{1/2}$ to precision $\epsilon_i = \Theta(1/d)$. Finally we apply a refinement procedure from Theorem 22 to reduce error to the final target precision ϵ, using $O(\log(1/\epsilon))$ multiplication with the crude approximator. The number of operations needed to compute $\tilde{\mathbf{C}}\boldsymbol{x}$ follows the complexity of Horner's method for polynomial evaluation. ■

The factorization formula from Equation 7 can be extended to any $p \in [-1, 1]$:

$$(\mathbf{I} - \mathbf{X})^p = (\mathbf{I} + \mathbf{X}/2)^{-p/2} \left(\mathbf{I} - \mathbf{X}/2 - \mathbf{X}^2/2\right)^p (\mathbf{I} + \mathbf{X}/2)^{-p/2}.$$

This allows us to construct an ϵ-approximate factor of $\mathbf{M}^p$ for general $p \in [-1, 1]$, with work nearly linear in m/ϵ^4. Because the refinement procedure in Theorem 22 does not apply to the general p^{th} power case, we need to have a longer chain with length $O(\log(\kappa/\epsilon))$ and $O(\log(\log\kappa/\epsilon))$-degree order Maclaurin expansion. It remains open if we can reduce the dependency on ϵ to $\log(1/\epsilon)$.

Theorem 9 (Factoring $\mathbf{M}^p$) *For any* $p \in [-1, 1]$*, and an SDDM matrix* $\mathbf{M} = \mathbf{I} - \mathbf{X}_0$*, we can construct a representation of an* $n \times n$ *matrix* $\tilde{\mathbf{C}}$ *such that* $\tilde{\mathbf{C}}\tilde{\mathbf{C}}^\top \approx_\epsilon \mathbf{M}^p$ *in work* $O(m \cdot \log^{c_1} n \cdot \log^{c_2}(\kappa/\epsilon)/\epsilon^4)$ *and depth* $O(\log^{c_3} n \cdot \log(\kappa/\epsilon))$ *for some constants* c_1, c_2*, and* c_3*. Moreover, for any* $\boldsymbol{x} \in \mathbb{R}^n$*, we can compute* $\tilde{\mathbf{C}}\boldsymbol{x}$ *in work* $O((m + n \cdot \log^c n \cdot \log^3(\kappa/\epsilon)/\epsilon^2) \cdot \log(\log(\kappa)/\epsilon))$ *and depth* $O(\log n \cdot \log(\kappa/\epsilon) \cdot \log(\log(\kappa)/\epsilon))$ *for some other constant* c*.*

5. Sparse Newton's Method

Newton's method is a powerful method for numerical approximation. Under desirable conditions, it may converge rapidly, which provides a framework for designing not only sequential but also parallel algorithms. However, one of the major drawbacks of Newton's method when applying to matrix functions, such as the ones consider in this paper, is that it may introduce dense matrix multiplications, even if the original matrix is sparse. Thus, when dealing with large-scale matrices, the intermediate computation for Newton's method could be prohibitively expensive.

In this section, we show how spectral sparsification can be used to speedup Newton's method for the computation of matrix roots. In doing so, we obtain an alternative symmetric factor reduction that leads to the fundamental problem for the sparsification of random-walk matrix polynomials. To motive our reduction formula, we first review Newton's method for computing the inverse square root of the scalar function $1-x$, which takes the form $y_{k+1} = y_k - f(y)/f'(y) = y_k\left[1+(1-(1-x)y_k^2)/2\right]$, and $f(y) = y^{-2} - (1-x)$, with $y_0 = 1$. Define the residue $r_i = 1-(1-x)y_k^2$. Then we can rewrite $y_{k+1} = \prod_{i=0}^{k}\left(1+\frac{r_i}{2}\right)$. With $y_0 = 1$, we have $r_0 = x$, and $(1-r_k)^{-1/2} = \left(1-3r_k^2/4-r_k^3/4\right)^{-1/2}(1+r_k/2)$. Replacing scalars with matrices with appropriate normalization leads to a different yet efficient representation of the classic Newton's method, i.e., reconstructing the final solution from the residues at each step. After symmetrization, it admits

$$(\mathbf{I}-\mathbf{X})^{-1} = (\mathbf{I}+\mathbf{X}/2)\left(\mathbf{I}-3\mathbf{X}^2/4-\mathbf{X}^3/4\right)^{-1}(\mathbf{I}+\mathbf{X}/2), \tag{9}$$

Equation (9) suggests that we can reduce the factorization of $(\mathbf{I}-\mathbf{X})^{-1}$ to the factorization of $\left(\mathbf{I}-\frac{3}{4}\mathbf{X}^2-\frac{1}{4}\mathbf{X}^3\right)^{-1}$. In general, when $q=-1/p$ is a positive integer, i.e., when calculating the inverse q^{th} root, the factorization formula from Equation (9) becomes

$$(\mathbf{I}-\mathbf{X})^{-1/q} = (\mathbf{I}+\mathbf{X}/2q)\left[(\mathbf{I}+\mathbf{X}/2q)^{2q}(\mathbf{I}-\mathbf{X})\right]^{-1/q}(\mathbf{I}+\mathbf{X}/2q), \tag{10}$$

where the middle term is closely connected to a random-walk polynomial of degree $(2q+1)$: We observe that all possible middle polynomials for different q share the same trait, that the coefficients (except that for the zero-order term) are non-positive and sum to 1. Here $\mathbf{X}$ is closely connected with the random-walk transition matrices on the underlying graph, and the polynomial represents a superposition of random walks of different steps. Thus, if these matrix polynomials can be efficiently sparsified, then our symmetric factor reduction scheme will yield an efficient algorithm for computing the inverse q^{th} root.

The need to "sparsify" Newton's method motivated us to solve the basic problem of computing a *spectral sparsifier* for *random-walk matrix polynomial* of the form

$$\mathbf{L}_{\boldsymbol{\alpha}}(G) = \mathbf{D} - \sum_{r=1}^{t}\alpha_r\mathbf{D}\cdot\left(\mathbf{D}^{-1}\mathbf{A}\right)^r, \tag{11}$$

given by an adjacency matrix $\mathbf{A}$ of a weighted, undirected graph G and a distribution vector $\boldsymbol{\alpha} = (\alpha_1,...,\alpha_t)$ satisfying $\sum_{r=1}^{t}\alpha_r = 1$, where recall $\mathbf{D}$ is the diagonal matrix of weighted degrees. Note that $\mathbf{D}^{-1}\mathbf{A}$ is the transition matrix of the reversible Markov chain of the random walks on the graph defined by $\mathbf{A}$. Below we focus our discussion on the sparsification of random-walk matrix polynomials because the result can be extended to polynomials involving $\mathbf{X}$ when $\mathbf{I}-\mathbf{X}$ is an SDDM matrix (see Appendix D for more details).

We sparsify $\mathbf{L}_{G_r} = \mathbf{D} - \mathbf{D}(\mathbf{D}^{-1}\mathbf{A})^r$ in two steps. In the first and critical step, we obtain an initial sparsifier with $O(r\cdot m\log n/\epsilon^2)$ non-zeros for $\mathbf{L}_{G_r}$. In the second step, we apply the standard spectral sparsification algorithms to further reduce the number of nonzeros to $O(n\log n/\epsilon^2)$.

In the first step sparsification, we found that for all positive odd integer r, we have $\frac{1}{2}\mathbf{L}_G \preceq \mathbf{L}_{G_r} \preceq r\mathbf{L}_G$, and for all positive even integers r we have $\mathbf{L}_{G_2} \preceq \mathbf{L}_{G_r} \preceq \frac{r}{2}\mathbf{L}_{G_2}$. It is proved in Lemma 26 by diagonalizing $\mathbf{X}$ to reduce it to the scalar case. Therefore, we can bound the effective resistance on G_r, which allows us to use the resistance of any length-r path on G to upper bound the effective resistance between its two endpoints on G_r, as shown in Lemma 10.

Sample a path $\boldsymbol{p}$ from G with probability proportional to $\tau_{\boldsymbol{p}} = w(\boldsymbol{p})Z(\boldsymbol{p})$:

a. Pick an integer $k \in [1 : r]$ and an edge $e \in G$, both uniformly at random.

b. Perform $(k-1)$-step random walk from one end of e.

c. Perform $(r-k)$-step random walk from the other end of e.

d. Keep track of $w(\boldsymbol{p})$ during the process, and finally add a fraction of this edge to our sparsifier.

Figure 2: Pseudocode for Sparsification of Random Walk Matrices

For each length-r path $\boldsymbol{p} = (u_0 \dots u_r)$ in G, we have a corresponding edge in G_r, with weight proportional to the chance of this particular path showing up in the random walk. We can view G_r as the union of these edges, i.e., $\mathbf{L}_{G_r}(u_0, u_r) = -\sum_{\boldsymbol{p}=(u_0 \dots u_r)} w(\boldsymbol{p})$.

We bound the effective resistance on G_r in two different ways. If r is odd, G is a good approximation of G_r, so we can obtain an upper bound using the resistance of a length-r (not necessarily simple) path on G. If r is even, G_2 is a good approximation of G_r, in this case, we get an upper bound by composing the effective resistance of 2-hop paths in different subgraphs of G_2. We summarize this in the following Lemma, which we prove in Appendix C.

Lemma 10 (Upper Bounds on Effective Resistance) *For a graph G with Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$, let $\mathbf{L}_{G_r} = \mathbf{D} - \mathbf{D}(\mathbf{D}^{-1}\mathbf{A})^r$ be its r-step random-walk matrix. Then, the effective resistance between two vertices u_0 and u_r on $\mathbf{L}_{G_r}$ is upper bounded by $R_{G_r}(u_0, u_r) \leq \sum_{i=1}^{r} \frac{2}{\mathbf{A}(u_{i-1}, u_i)}$, where $(u_0 \dots u_r)$ is a path in G.*

The simple form of the upper bounds yields the following mathematical identity that will be crucial in our analysis: $\sum_{\boldsymbol{p}} w(\boldsymbol{p}) \cdot Z(\boldsymbol{p}) = 2rm$, as stated in Lemma 29. Here $w(\boldsymbol{p})$ and $Z(\boldsymbol{p})$ are respectively the weight and the upper bound on effective resistance of the edge associated with a length-r path $\boldsymbol{p} = (u_0 \dots u_r)$ on G. Formally, $w(\boldsymbol{p}) = \prod_{i=1}^{r} \mathbf{A}(u_{i-1}, u_i) / \prod_{i=1}^{r-1} \mathbf{D}(u_i, u_i)$, and $Z(\boldsymbol{p}) = \sum_{i=1}^{r} \frac{2}{\mathbf{A}(u_{i-1}, u_i)}$.

Now we show that we can perform Step (2) in GRAPHSAMPLING efficiently. Lemma 29 shows that sampling $\tilde{O}(rm)$ paths suffices. Recall that sampling an edge from G_r corresponds to sampling a path of length r in G. We take samples in the same way we cancel the terms in the previous proof.

Lemma 11 *There exists an algorithm for the Step (2) in GRAPHSAMPLING, such that after preprocessing with work $O(n)$, it can draw an edge e as in Step (2) with work $O(r \cdot \log n)$.*

Proof The task is to draw a sample $\boldsymbol{p} = (u_0 \dots u_r)$ from the multivariate distribution $\mathcal{D}$

$$\Pr(u_0 \dots u_r) = \frac{1}{2rm} \cdot \left(\sum_{i=1}^{r} \frac{2}{\mathbf{A}(u_{i-1}, u_i)} \right) \cdot \left(\frac{\prod_{i=1}^{r} \mathbf{A}(u_{i-1}, u_i)}{\prod_{j=1}^{r-1} \mathbf{D}(u_i, u_i)} \right). \tag{12}$$

For any fixed $k \in [1 : r]$ and $e \in G$, we can rewrite the distribution as

$$\Pr(u_0 \dots u_r) = \Pr((u_{k-1}, u_k) = e) \cdot \prod_{1}^{i=k-1} \Pr(u_{i-1} | u_i) \cdot \prod_{i=k+1}^{r} \Pr(u_i | u_{i-1}) \tag{13}$$

Note that $\Pr((u_{k-1}, u_k) = e) = \frac{1}{m}$, and $\Pr(u_{i-1}|u_i) = \mathbf{A}(u_i, u_{i-1})/\mathbf{D}(u_i, u_i)$. The three terms in Equation (13) corresponds to Step (a)-(c) in the sampling algorithm stated above. With linear preprocessing time, we can draw an uniform random edge in time $O(\log n)$, and we can also simulate two random walks with total length r in time $O(r \log n)$, so Step (2) in GRAPHSAMPLING can be done within $O(r \log n)$ time. ■

Combining this with spectral sparsifiers gives our algorithm for efficiently sparsifying low-degree polynomials.

Proof [Proof of Theorem 2] First we show on how to sparsify a degree-t monomial $\mathbf{L}_{G_d} = \mathbf{D} - \mathbf{D} \cdot \left(\mathbf{D}^{-1}\mathbf{A}\right)^t$. We use the sampling algorithm described in Theorem 5, together with upper bounds on effective resistance of G_t obtained from $\mathbf{D} - \mathbf{A}$ and $\mathbf{D} - \mathbf{A}\mathbf{D}^{-1}\mathbf{A}$. The total number of samples requires is $O(tm \log n/\epsilon^2)$. We use Lemma 11 to draw a single edge from G_t, where we sample t-step random walks on $\mathbf{D} - \mathbf{A}$, so the total running time is $O(t^2 m \log^2 n/\epsilon^2)$. Now that we have a spectral sparsifier with $O(tm \log n/\epsilon^2)$ edges, we can sparsify one more time to reduce the number of edges to $O(n \log n/\epsilon^2)$ by Spielman and Teng (2011) in time $O(tm \log^2 n/\epsilon^2)$.

To sparsify a random-walk matrix polynomial $\mathbf{L}_{\boldsymbol{\alpha}}(G)$, we sparsify all the even/odd terms together, so the upper bound of effective resistance in Lemma 10 still holds. To sample an edge, we first decide the length r of the path, according to the probability distribution $\Pr(\text{length} = r | r \text{ is odd/even}) \propto \alpha_r$. ■

The factorization in Equation (10) leads to a family of sparse inverse chain:

Definition 12 (q-Sparse Inverse Factor Chain) *For a sequence of approximation parameters $\boldsymbol{\epsilon} = (\epsilon_0, \ldots, \epsilon_d)$, a $(q, \boldsymbol{\epsilon})$-sparse factor chain $(\mathbf{X}_1, \ldots, \mathbf{X}_d)$ for $\mathbf{M} = \mathbf{I} - \mathbf{X}_0$ satisfies the following conditions:*

1. $i = 0, 1, 2, \ldots, d-1$, $\mathbf{I} - \mathbf{X}_{i+1} \approx_{\epsilon_i} \left(\mathbf{I} + \frac{\mathbf{X}_i}{2q}\right)^{2q} (\mathbf{I} - \mathbf{X}_i)$;
2. $\mathbf{I} \approx_{\epsilon_d} \mathbf{I} - \mathbf{X}_d$.

To obtain a random sample $\boldsymbol{y}$, we multiply a random vector $\boldsymbol{x}$ containing i.i.d. standard Gaussian random variable through the chain: $\boldsymbol{y} = (\mathbf{I} + \mathbf{X}_0/2q) \cdot (\mathbf{I} + \mathbf{X}_1/2q) \cdots (\mathbf{I} + \mathbf{X}_{d-1}/2q)\, \boldsymbol{x}$.

Now we state our main theorem for the Newton's method based factorization for matrix inverse factorization. The result for inverse q^{th} root is deferred to Appendix C (See Theorem 30). Note that the chain construction cost is on the same order as in the *Maclaurin series based parallel inverse factorization*. But without the involvement of the Maclaurin series, it requires less work when multiplied with a vector.

Theorem 13 (Newton's Method Based Parallel Inverse Factorization) *Given an SDDM matrix $\mathbf{M} = \mathbf{I} - \mathbf{X}_0$ and a precision parameter ϵ, we can construct a representation of an $n \times n$ matrix $\tilde{\mathbf{C}}$, such that $\tilde{\mathbf{C}}\tilde{\mathbf{C}}^\top \approx_\epsilon (\mathbf{I} - \mathbf{X}_0)^{-1}$ in work $O(m \cdot \log^{c_1} n \cdot \log^{c_2} \kappa)$ and parallel depth $O(\log^{c_3} n \cdot \log \kappa)$ for some constants c_1, c_2, and c_3.*

Moreover, for any $\boldsymbol{x} \in \mathbb{R}^n$, we can compute $\tilde{\mathbf{C}}\boldsymbol{x}$ in work $O((m + n \cdot \log^c n \cdot \log^3 \kappa) \cdot \log(1/\epsilon))$ and depth $O(\log n \cdot \log \kappa \cdot \log(1/\epsilon))$ for some other constant c.

Acknowledgments

D. Cheng and Y. Liu are supported in part by NSF research grants IIS-1134990 and NSF research grants IIS- 1254206. Y. Cheng and S.-H. Teng are supported in part by a Simons Investigator Award from the Simons Foundation and the NSF grant CCF-1111270.

References

Sungjin Ahn, Babak Shahbaba, and Max Welling. Distributed stochastic gradient MCMC. In *ICML*, 2014.

Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to MCMC for machine learning. *Machine learning*, 50, 2003.

Rajendra Bhatia. *Positive definite matrices*. Princeton University Press, 2007.

Christopher M. Bishop. *Pattern recognition and machine learning*. springer New York, 2006.

Edmond Chow and Yousef Saad. Preconditioned krylov subspace methods for sampling multivariate gaussian distributions. *SIAM Journal on Scientific Computing*, 2014.

Paul Christiano, Jonathan A Kelner, Aleksander Madry, Daniel A Spielman, and Shang-Hua Teng. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 273–282. ACM, 2011.

Ronald R Coifman, Stephane Lafon, Ann B Lee, Mauro Maggioni, Boaz Nadler, Frederick Warner, and Steven W Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Multiscale methods. *Proceedings of the National Academy of Sciences of the United States of America*, 102(21):7432–7437, 2005.

Samuel I. Daitch and Daniel A. Spielman. Faster approximate lossy generalized flow via interior point algorithms. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, STOC '08. ACM, 2008.

Joseph Gonzalez, Yucheng Low, Arthur Gretton, and Carlos Guestrin. Parallel gibbs sampling: From colored fields to thin junction trees. In *Proc. of AISTATS'11*, 2011.

Keith D Gremban. *Combinatorial preconditioners for sparse, symmetric, diagonally dominant linear systems*. PhD thesis, Carnegie Mellon University, 1996.

Nicholas J Higham and Lijing Lin. On pth roots of stochastic matrices. *Linear Algebra and its Applications*, 435(3):448–463, 2011.

Alexander Ihler, Er T. Ihler, Erik Sudderth, William Freeman, and Alan Willsky. Efficient multiscale sampling from products of gaussian mixtures. In *In NIPS*. MIT Press, 2003.

Matthew Johnson, James Saunderson, and Alan Willsky. Analyzing hogwild parallel gaussian gibbs sampling. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2715–2723. Curran Associates, Inc., 2013.

M. I. Jordan. *Learning in Graphical Models*. The MIT press, 1998.

Jonathan A Kelner and Alex Levin. Spectral sparsification in the semi-streaming setting. *Theory of Computing Systems*, 53(2):243–262, 2013.

Jonathan A. Kelner, Lorenzo Orecchia, Aaron Sidford, and Zeyuan Allen Zhu. A simple, combinatorial algorithm for solving sdd systems in nearly-linear time. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, 2013.

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009. ISBN 0262013193, 9780262013192.

Ioannis Koutis. A simple parallel algorithm for spectral sparsification. *CoRR*, 2014.

Ioannis Koutis, Gary L. Miller, and Richard 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, 2010.

S. Leang Shieh, Jason Tsai, and Sui Lian. Determining continuous-time state equations from discrete-time state equations via the principal q th root method. *Automatic Control, IEEE Transactions on*, 1986.

Ying Liu, Oliver Kosut, and Alan S. Willsky. Sampling from gaussian graphical models using subgraph perturbations. In *Proceedings of the 2013 IEEE International Symposium on Information Theory, Istanbul, Turkey, July 7-12, 2013*, 2013.

Gary L. Miller, Richard Peng, and Shen Chen Xu. Improved parallel algorithms for spanners and hopsets. *CoRR*, 2013.

Geng Niu, Benjamin Recht, Christopher Re, and Stephen J. Wright. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS*, 2011.

Richard Peng. *Algorithm Design Using Spectral Graph Theory*. PhD thesis, Carnegie Mellon University, Pittsburgh, August 2013. CMU CS Tech Report CMU-CS-13-121.

Richard Peng and Daniel A. Spielman. An efficient parallel solver for sdd linear systems. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC '14. ACM, 2014.

Daniel A. Spielman and Nikil Srivastava. Graph sparsification by effective resistances. *SIAM J. Comput.*, 40(6):1913–1926, 2011.

Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing*, STOC '04, 2004.

Daniel A. Spielman and Shang-Hua Teng. Spectral sparsification of graphs. *SIAM J. Comput.*, 2011.

Daniel A. Spielman and Shang-Hua Teng. Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *SIAM J. Matrix Analysis and Applications*, 35(3):835–885, 2014.

Appendix A. Sparse Factor Chains: Construction and Analysis

Using the Maclaurin chain, our basic algorithm constructs an approximate factor of $(\mathbf{I}-\mathbf{X})^p$, for any $p \in [-1,1]$. Using the factorization with Maclaurin matrix polynomials given by Lemma 24, we extend the framework of Peng and Spielman (2014) for parallel approximation of matrix inverse to SDDM factorization: Our algorithm uses the following identity

$$(\mathbf{I}-\mathbf{X})^p = \left(\mathbf{I}+\frac{1}{2}\mathbf{X}\right)^{-p/2}\left(\mathbf{I}-\frac{1}{2}\mathbf{X}-\frac{1}{2}\mathbf{X}^2\right)^p\left(\mathbf{I}+\frac{1}{2}\mathbf{X}\right)^{-p/2}, \tag{14}$$

and the fact that we can approximate $\left(\mathbf{I}+\frac{1}{2}\mathbf{X}\right)^{-p/2}$ with its Maclaurin series expansion (Lemma 24). It thus numerically reduces the factorization problem of $(\mathbf{I}-\mathbf{X})^p$ to that of $\left(\mathbf{I}-\frac{1}{2}\mathbf{X}-\frac{1}{2}\mathbf{X}^2\right)^p$. Similarly, using the Newton chain, we numerically reduce the factorization problem of $(\mathbf{I}-\mathbf{X})^{-1/q}$ where $q \in \mathbb{Z}^{++}$ to that of $\left[\left(\mathbf{I}+\frac{\mathbf{X}}{2q}\right)^{2q}(\mathbf{I}-\mathbf{X})\right]^{-1/q}$.

In summary, the chain is constructed by the reduction from $\mathbf{I}-\mathbf{X}$ to $\mathbf{I}-\sum_{r=1}^t \alpha_r \mathbf{X}^r$, where $\alpha_r \geq 0$ and $\sum_{r=1}^t \alpha_r = 1$. The key to efficient applications of this iterative reduction is the spectral sparsification of $\mathbf{I}-\sum_{r=1}^t \alpha_r \mathbf{X}^r$, so that our matrix polynomials only use sparse matrices. In particular, we start with $\mathbf{I}-\mathbf{X}_0$, and at step i we sparsify $\mathbf{I}-\sum_{r=1}^t \alpha_r \mathbf{X}_i^r$ to obtain its spectral approximation $\mathbf{I}-\mathbf{X}_{i+1}$, and then proceed to the next iteration until $\rho(\mathbf{X}_d)$ is small enough, at which point we approximate $\mathbf{I}-\mathbf{X}_d$ with $\mathbf{I}$.

The following basic lemma then guarantees spectral approximation is preserved under the p^{th} power, which enables the continuation of the reduction.

Lemma 14 *Let $\mathbf{A}$, $\mathbf{B}$ be positive definite matrices with $\mathbf{A} \approx_\epsilon \mathbf{B}$, then for all $p \in [-1,1]$, we have that*

$$\mathbf{A}^p \approx_{|p|\epsilon} \mathbf{B}^p. \tag{15}$$

Proof By Theorem 4.2.1 of Bhatia (2007), if $\mathbf{A} \succcurlyeq \mathbf{B}$ then $\mathbf{A}^p \succcurlyeq \mathbf{B}^p$ for all $p \in [0,1]$. Thus, if $\mathbf{A} \approx_\epsilon \mathbf{B}$, then for all $p \in [0,1]$,

$$\exp(p\epsilon)\,\mathbf{A}^p \succcurlyeq \mathbf{B}^p \succcurlyeq \exp(-p\epsilon)\,\mathbf{A}^p, \tag{16}$$

which implies $\mathbf{A}^p \approx_{|p|\epsilon} \mathbf{B}^p$. By Fact 4.d, the claim holds for all $p \in [-1,0]$ as well. ■

In the rest of the section, we will bound the length of our sparse factor chain (Section A.1), and show that the chain indeed provides a good approximation to $(\mathbf{I}-\mathbf{X}_0)^p$ (Section A.2). In Section A.3, we present a refinement technique to further reduce the dependency of ϵ to $\log(1/\epsilon)$ for the case when $p = \pm 1$.

A.1. Convergence of the Matrix Chain

To bound the length of the sparse factor chain, we need to study the convergence behavior of $\rho(\mathbf{X}_i)$. We show that when $\alpha_1 \leq \frac{1}{2}$, a properly constructed matrix chain of length $O(\log(\kappa/\epsilon))$ is sufficient to achieve ϵ precision. The requirement $\alpha_1 \leq \frac{1}{2}$ is for the simplicity of the proof. The chain will

converge for any α_1 such that $1-\alpha_1 = \Omega(1)$. Our analysis is analogous to the one in Peng and Spielman (2014). However, we have to deal with the general form, i.e., $\mathbf{I} - \sum_{r=1}^{t} \alpha_r \mathbf{X}^r$.

We define the following two polynomials for simplicity: $f_{\boldsymbol{\alpha}}(x) = 1 - \sum_{r=1}^{t} \alpha_r (1-x)^r$ and $g_{\boldsymbol{\alpha}}(x) = 1 - \sum_{r=0}^{\lfloor t/2 \rfloor - 1} \alpha_r (1-x)^{2r+1}$. For instance, the Maclaurin chain admits $f_{(0.5,0.5)}(x) = \left(3x - x^2\right)/2$ and $g_{(0.5,0.5)}(x) = (3-x)/2$.

We start with the following lemma, which analyzes one iteration of the chain construction. We also take into account the approximation error introduced by sparsification.

Lemma 15 *Let $\mathbf{X}$ and $\bar{\mathbf{X}}$ be nonnegative symmetric matrices such that $\mathbf{I} \succcurlyeq \mathbf{X}$, $\mathbf{I} \succcurlyeq \bar{\mathbf{X}}$, and*

$$\mathbf{I} - \bar{\mathbf{X}} \approx_{\epsilon} \mathbf{I} - \sum_{r=1}^{t} \alpha_r \mathbf{X}^r. \tag{17}$$

If $\rho(\mathbf{X}) \leq 1-\lambda$, then the eigenvalues of $\bar{\mathbf{X}}$ all lie between $1 - g_{\boldsymbol{\alpha}}(\lambda)\exp(\epsilon)$ and $1 - f_{\boldsymbol{\alpha}}(\lambda)\exp(-\epsilon)$.

Proof If the eigenvalues of $\mathbf{I} - \mathbf{X}$ are between λ and $2-\lambda$, the eigenvalues of $\mathbf{I} - \sum_{r=1}^{t} \alpha_r \mathbf{X}^r$ will belong to $[f_{\boldsymbol{\alpha}}(\lambda), g_{\boldsymbol{\alpha}}(\lambda)]$. The bound stated in this lemma then follows from the fact that the spectral sparsification step preserves the eigenvalues of $\mathbf{I} - \sum_{r=1}^{t} \alpha_r \mathbf{X}^r$ to within a factor of $\exp(\pm\epsilon)$. ■

We next prove that our Maclaurin chain achieves overall precision ϵ with length $O(\log(\kappa/\epsilon))$.

Lemma 16 (Short Maclaurin Chain) *Let $\mathbf{M} = \mathbf{I} - \mathbf{X}_0$ be an SDDM matrix with condition number κ. For any approximation parameter ϵ, there exists an $(\epsilon_0, \ldots, \epsilon_d)$-sparse factor chain of $\mathbf{M}$ with $d = \log_{9/8}(\kappa/\epsilon)$ and $\sum_{j=0}^{d} \epsilon_j \leq \epsilon$, satisfying*

$$\begin{aligned} \mathbf{I} - \mathbf{X}_{i+1} &\approx_{\epsilon_i} \mathbf{I} - \frac{1}{2}\mathbf{X}_i - \frac{1}{2}\mathbf{X}_i^2 \qquad \forall\, 0 \leq i \leq d-1, \\ \mathbf{I} - \mathbf{X}_d &\approx_{\epsilon_d} \mathbf{I}. \end{aligned} \tag{18}$$

Proof Without loss of generality, we assume $\epsilon \leq \frac{1}{10}$. For $0 \leq i \leq d-1$, we set $\epsilon_i = \frac{\epsilon}{8d}$. We will show that our chain satisfies the condition given by Equation (18) with $\epsilon_d \leq (7\epsilon)/8$, and thus $\sum_{i=0}^{d} \epsilon_i \leq \epsilon$.

Let $\lambda_i = 1 - \rho(\mathbf{X}_i)$. We split the error analysis to two parts, one for $\lambda_i \leq \frac{1}{2}$, and one for $\lambda_i \geq \frac{1}{2}$.

In the first part, because $\epsilon_i \leq 1/10$,

$$\begin{aligned} 1 - \frac{1}{2}\left(3\lambda_i - \lambda_i^2\right)\exp(-\epsilon_i) &< 1 - \frac{9}{8}\lambda_i \quad \text{and} \\ 1 - \frac{1}{2}\left(3 - \lambda_i\right)\exp(\epsilon_i) &> -1 + \frac{9}{8}\lambda_i \end{aligned} \tag{19}$$

which, by Lemma 15, implies that $\lambda_{i+1} \geq \frac{9}{8}\lambda_i$. Because initially, $\lambda_0 \geq 1/\kappa$, after $d_1 = \log_{9/8}\kappa$ iterations, it must be the case that $\lambda_{d_1} \geq \frac{1}{2}$. Now we enter the second part. Note that $\epsilon_i = \frac{\epsilon}{8d} \leq \frac{\epsilon}{6}$. Thus, when $\frac{1}{2} \leq \lambda_i \leq 1 - \frac{5}{6}\epsilon$, we have $\epsilon_i \leq \frac{1-\lambda_i}{5}$ and therefore

$$\begin{aligned} 1 - \frac{1}{2}\left(3\lambda_i - \lambda_i^2\right)\exp(-\epsilon_i) &< \frac{8}{9}(1-\lambda_i) \quad \text{and} \\ 1 - \frac{1}{2}\left(3 - \lambda_i\right)\exp(\epsilon_i) &> \frac{8}{9}(-1+\lambda_i) \end{aligned} \tag{20}$$

which, by Lemma 15, implies that $1-\lambda_{i+1} \leq \frac{8}{9}(1-\lambda_i)$. Because $1-\lambda_{d_1} \leq \frac{1}{2}$, after another $d_2 = \log_{9/8}(1/\epsilon)$ iterations, we get $\rho\left(\mathbf{X}_{d_1+d_2}\right) \leq \frac{5}{6}\epsilon$. Because $\exp(-\frac{7}{8}\epsilon) < 1-\frac{5}{6}\epsilon$ when $\epsilon < \frac{1}{10}$, we conclude that $(\mathbf{I}-\mathbf{X}_{d_1+d_2}) \approx_{\epsilon_d} \mathbf{I}$ with $\epsilon_d \leq \frac{7}{8}\epsilon$. ■

Similarly, we can easily prove our Newton chain achieves overall precision ϵ with length $O(\log(\kappa/\epsilon))$. Indeed, any chain with $\alpha_1 < \frac{1}{2}$ converges strictly faster than the Maclaurin chain.

Corollary 17 (Short Newton Chain) *Let $\mathbf{M} = \mathbf{I}-\mathbf{X}_0$ be an SDDM matrix with condition number κ and the vector $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_t)$ with $\alpha_1 \leq \frac{1}{2}$. For any approximation parameter ϵ, there exists an $[(\epsilon_0, \ldots, \epsilon_d), \boldsymbol{\alpha}]$-sparse factor chain of $\mathbf{M}$ with $d = \log_{9/8}(\kappa/\epsilon)$ and $\sum_{j=0}^{d}\epsilon_j \leq \epsilon$, satisfying*

$$\begin{aligned} \mathbf{I}-\mathbf{X}_{i+1} &\approx_{\epsilon_i} \mathbf{I} - \sum_{r=1}^{t}\alpha_r\mathbf{X}^r \qquad \forall\, 0 \leq i \leq d-1, \\ \mathbf{I}-\mathbf{X}_d &\approx_{\epsilon_d} \mathbf{I}. \end{aligned} \tag{21}$$

A.2. Precision of the Factor Chain

We now bound the total error of the matrix factor that our algorithm constructs.

A.2.1. MACLAURIN CHAIN

We start with an error-analysis of a single reduction step in the algorithm.

Lemma 18 *Fix $p \in [-1,1]$, given an $\boldsymbol{\epsilon}$-sparse factor chain, for all $0 \leq i \leq d-1$, there exists degree t_i polynomials, with $t_i = O(\log(1/\epsilon_i))$, such that*

$$(\mathbf{I}-\mathbf{X}_i)^p \approx_{2\epsilon_i} T_{-\frac{1}{2}p,t_i}\left(\mathbf{I}+\frac{1}{2}\mathbf{X}_i\right)(\mathbf{I}-\mathbf{X}_{i+1})^p\, T_{-\frac{1}{2}p,t_i}\left(\mathbf{I}+\frac{1}{2}\mathbf{X}_i\right). \tag{22}$$

Proof

$$\begin{aligned} (\mathbf{I}-\mathbf{X}_i)^p &\approx_{\epsilon_i} T_{-\frac{1}{2}p,t_i}\left(\mathbf{I}+\frac{1}{2}\mathbf{X}_i\right)\left(\mathbf{I}-\frac{1}{2}\mathbf{X}_i-\frac{1}{2}\mathbf{X}_i^2\right)^p T_{-\frac{1}{2}p,t_i}\left(\mathbf{I}+\frac{1}{2}\mathbf{X}_i\right) \\ &\approx_{\epsilon_i} T_{-\frac{1}{2}p,t_i}\left(\mathbf{I}+\frac{1}{2}\mathbf{X}_i\right)(\mathbf{I}-\mathbf{X}_{i+1})^p\, T_{-\frac{1}{2}p,t_i}\left(\mathbf{I}+\frac{1}{2}\mathbf{X}_i\right). \end{aligned} \tag{23}$$

The first approximation follows from Lemma 24. The second approximation follows from Lemma 14, Fact 4.e, and $\mathbf{I}-\mathbf{X}_{i+1} \approx_{\epsilon_i} \mathbf{I}-\frac{1}{2}\mathbf{X}_i-\frac{1}{2}\mathbf{X}_i^2$. ■

The next lemma bounds the total error of our construction.

Lemma 19 *Define $\mathbf{Z}_{p,i}$ as*

$$\mathbf{Z}_{p,i} = \begin{cases} \mathbf{I} & \text{if } i = d, \text{ and} \\ \left(T_{-\frac{1}{2}p,t_i}\left(\mathbf{I}+\frac{1}{2}\mathbf{X}_i\right)\right)\mathbf{Z}_{p,i+1} & \text{if } 0 \leq i \leq d-1. \end{cases}$$

Then the following statement is true for all $0 \le i \le d$

$$\mathbf{Z}_{p,i}\mathbf{Z}_{p,i}^\top \approx_{2\sum_{j=i}^{d}\epsilon_j} (\mathbf{I}-\mathbf{X}_i)^p\,. \tag{24}$$

Proof We prove this lemma by reverse induction. Combine $\mathbf{I} \approx_{\epsilon_d} \mathbf{I}-\mathbf{X}_d$ and Lemma 14, we know that $\mathbf{Z}_{p,d}\mathbf{Z}_{p,d}^\top = \mathbf{I}^p \approx_{\epsilon_d} (\mathbf{I}-\mathbf{X}_d)^p$, so the claim is true for $i=d$. Now assume the claim is true for $i=k+1$, then

$$\begin{aligned}\mathbf{Z}_{p,k}\mathbf{Z}_{p,k}^\top =& T_{-\frac{1}{2}p,t_k}\left(\mathbf{I}+\frac{1}{2}\mathbf{X}_k\right)\mathbf{Z}_{p,k+1}\mathbf{Z}_{p,k+1}^\top T_{-\frac{1}{2}p,t_k}\left(\mathbf{I}+\frac{1}{2}\mathbf{X}_k\right)\\ \approx_{2\sum_{j=k+1}^{d}\epsilon_j}& T_{-\frac{1}{2}p,t_k}\left(\mathbf{I}+\frac{1}{2}\mathbf{X}_k\right)(\mathbf{I}-\mathbf{X}_{k+1})^p\, T_{-\frac{1}{2}p,t_k}\left(\mathbf{I}+\frac{1}{2}\mathbf{X}_k\right)\\ \approx_{2\epsilon_k}& (\mathbf{I}-\mathbf{X}_k)^p\,.\end{aligned} \tag{25}$$

The last approximation follows from Lemma 18. ■

If we define $\tilde{\mathbf{C}}$ to be $\mathbf{Z}_{-\frac{1}{2},0}$, this lemma implies $\tilde{\mathbf{C}}\tilde{\mathbf{C}}^\top \approx_{2\sum_{j=0}^{d}\epsilon_j} (\mathbf{I}-\mathbf{X}_0)^{-1}$.

A.2.2. NEWTON CHAIN

We start with an error-analysis of a single reduction step in the algorithm. We assume that the $2q+1$-dimensional vector $\boldsymbol{\alpha}$, where $q \in \mathbb{Z}^{++}$, is defined by the identity: $1-\sum_{r=1}^{2q+1}\alpha_r = \left(1+\frac{x}{2q}\right)^{2q}(1-x)$.

Lemma 20 *Fix* $p \in [-1,1]$*, given an* $\boldsymbol{\epsilon}$*-sparse factor chain, for all* $0 \le i \le d-1$*, there exists degree* t_i *polynomials, with* $t_i = O(\log(1/\epsilon_i))$*, such that*

$$(\mathbf{I}-\mathbf{X}_i)^{-1/q} \approx_{\epsilon_i} \left(\mathbf{I}+\frac{\mathbf{X}_i}{2q}\right)(\mathbf{I}-\mathbf{X}_{i+1})^{-1/q}\left(\mathbf{I}+\frac{\mathbf{X}_i}{2q}\right). \tag{26}$$

Proof

$$\begin{aligned}(\mathbf{I}-\mathbf{X}_i)^{-1/q} &= \left(\mathbf{I}+\frac{\mathbf{X}_i}{2q}\right)\left(\mathbf{I}-\sum_{r=1}^{2q+1}\alpha_r\mathbf{X}_i^r\right)^{-1/q}\left(\mathbf{I}+\frac{\mathbf{X}_i}{2q}\right)\\ &\approx_{\epsilon_i} \left(\mathbf{I}+\frac{\mathbf{X}_i}{2q}\right)(\mathbf{I}-\mathbf{X}_{i+1})^{-1/q}\left(\mathbf{I}+\frac{\mathbf{X}_i}{2q}\right).\end{aligned} \tag{27}$$

The first identity follows from Equation 10. The second approximation follows from Lemma 14, Fact 4.e, and $\mathbf{I}-\mathbf{X}_{i+1} \approx_{\epsilon_i} \mathbf{I}-\sum_{r=1}^{2q+1}\alpha_r\mathbf{X}_i^r$. ■

The next lemma bounds the total error of our construction.

Lemma 21 *Define* $\mathbf{W}_{q,i}$ *as*

$$\mathbf{W}_{q,i} = \begin{cases}\mathbf{I} & \text{if } i=d, \text{ and}\\ \left(\mathbf{I}+\frac{\mathbf{X}_i}{2q}\right)\mathbf{W}_{q,i+1} & \text{if } 0 \le i \le d-1.\end{cases}$$

Then the following statement is true for all $0 \leq i \leq d$

$$\mathbf{W}_{q,i}\mathbf{W}_{q,i}^{\top} \approx_{\sum_{j=i}^{d}\epsilon_j} (\mathbf{I} - \mathbf{X}_i)^{-1/q}. \tag{28}$$

Proof The proof is similar to Lemma 19. ■

If we define $\tilde{\mathbf{C}}$ to be $\mathbf{W}_{1,0}$, this lemma implies $\tilde{\mathbf{C}}\tilde{\mathbf{C}}^{\top} \approx_{\sum_{j=0}^{d}\epsilon_j} (\mathbf{I} - \mathbf{X}_0)^{-1}$.

A.3. Iterative Refinement for Inverse Square-Root Factor

In this section, we provide a highly accurate factor for the case when $p = -1$. We use a refinement approach inspired by the precondition technique for square factors described in Section 2.2 of Chow and Saad (2014).

Theorem 22 *For any symmetric positive definite matrix* $\mathbf{M}$*, any matrix* $\mathbf{C}$ *such that* $\mathbf{M}^{-1} \approx_{O(1)} \mathbf{Z}\mathbf{Z}^{\top}$*, and any error tolerance* $\epsilon > 0$*, there exists a linear operator* $\tilde{\mathbf{C}}$ *which is an* $O(\log(1/\epsilon))$*-degree polynomial of* $\mathbf{M}$ *and* $\mathbf{Z}$*, such that* $\tilde{\mathbf{C}}\tilde{\mathbf{C}}^{\top} \approx_{\epsilon} \mathbf{M}^{-1}$.

Proof

Starting from $\mathbf{M}^{-1} \approx_{O(1)} \mathbf{Z}\mathbf{Z}^{\top}$, we know that $\mathbf{Z}^{\top}\mathbf{M}\mathbf{Z} \approx_{O(1)} \mathbf{I}$. This implies that $\kappa(\mathbf{Z}^{\top}\mathbf{M}\mathbf{Z}) = O(1)$, which we can scale $\mathbf{Z}^{\top}\mathbf{M}\mathbf{Z}$ so that its eigenvalues lie in $[1-\delta, 1+\delta]$ for some constant $0 < \delta < 1$.

By applying Lemma 23 on its eigenvalues, there is an $O(\log(1/\epsilon))$ degree polynomial $T_{-\frac{1}{2},t}(\cdot)$ that approximates the inverse square root of $\mathbf{Z}^{\top}\mathbf{M}\mathbf{Z}$, i.e.

$$\left(T_{-\frac{1}{2},O(\log(1/\epsilon))}\left(\mathbf{Z}^{\top}\mathbf{M}\mathbf{Z}\right)\right)^2 \approx_{\epsilon} \left(\mathbf{Z}^{\top}\mathbf{M}\mathbf{Z}\right)^{-1}. \tag{29}$$

Here, we can rescale $\mathbf{Z}^{\top}\mathbf{M}\mathbf{Z}$ back inside $T_{-\frac{1}{2},t}(\cdot)$, which does not affect the multiplicative error. Then by Fact 4.e, we have

$$\mathbf{Z}\left(T_{-\frac{1}{2},O(\log(1/\epsilon))}\left(\mathbf{Z}^{\top}\mathbf{M}\mathbf{Z}\right)\right)^2\mathbf{Z}^{\top} \approx_{\epsilon} \mathbf{Z}\left(\mathbf{Z}^{\top}\mathbf{M}\mathbf{Z}\right)^{-1}\mathbf{Z}^{\top} = \mathbf{M}^{-1}. \tag{30}$$

So if we define the linear operator $\tilde{\mathbf{C}} = \mathbf{Z}\left(T_{-\frac{1}{2},O(\log(1/\epsilon))}\left(\mathbf{Z}^{\top}\mathbf{M}\mathbf{Z}\right)\right)$, $\tilde{\mathbf{C}}$ satisfies the claimed properties. ■

Note that this refinement procedure is specific to $p = \pm 1$. It remains open if it can be extended to general $p \in [-1, 1]$.

Appendix B. Maclaurin Series Expansion

We show how to approximate $\left(\mathbf{I} + \frac{1}{2}\mathbf{X}\right)^p$, when $\rho(\mathbf{X}) < 1$. Because $\mathbf{I} + \frac{1}{2}\mathbf{X}$ is well-conditioned, we can approximate its p^{th} power to any approximation parameter $\epsilon > 0$ using an $O(\log(1/\epsilon))$-order Maclaurin expansion, i.e., a low degree polynomial of $\mathbf{X}$. Moreover, since the approximation is a polynomial of $\mathbf{X}$, the eigenbasis is preserved. We start with the following lemma on the residue of Maclaurin expansion.

Lemma 23 (Maclaurin Series) *Fix $p \in [-1,1]$ and $\delta \in (0,1)$, for any error tolerance $\epsilon > 0$, there exists a t-degree polynomial $T_{p,t}(\cdot)$ with $t \leq \frac{\log(1/(\epsilon(1-\delta)^2)}{1-\delta}$, such that for all $\lambda \in [1-\delta, 1+\delta]$,*

$$\exp(-\epsilon)\lambda^p \leq T_{p,t}(\lambda) \leq \exp(\epsilon)\lambda^p. \tag{31}$$

Proof Let $g_{p,t}(x) = \sum_{i=0}^{\infty} a_i x^i$ be the t^{th} order Maclaurin series expansion of $(1-x)^p$. Because $|a_1| = |p| \leq 1$, and $|a_{k+1}| = \left|\frac{p-k}{k+1}\right| |a_k| \leq |a_k|$, we can prove that $|a_i| \leq 1$ for all i by induction. For $|x| \leq \delta < 1$, we have

$$|(1-x)^p - g_{p,t}(x)| = \left|\sum_{i=t+1}^{\infty} a_i x^i\right| \leq \sum_{i=t+1}^{\infty} \delta^i = \frac{\delta^{t+1}}{1-\delta}. \tag{32}$$

We define $T_{p,t}(x) = g_{p,t}(1-x)$. Because $|1-\lambda| \leq \delta$ and $\lambda^p \geq 1-\delta$,

$$|T_{p,t}(\lambda) - \lambda^p| \leq \frac{\delta^{t+1}}{1-\delta} \leq \frac{\delta^{t+1}}{(1-\delta)^2}\lambda^p. \tag{33}$$

Therefore, when $t = (1-\delta)^{-1}\log(1/(\epsilon(1-\delta)^2)))$, we get $\frac{\delta^{t+1}}{(1-\delta)^2} < \epsilon$ and

$$|T_{p,t}(\lambda) - \lambda^p| \leq \epsilon\lambda^p, \tag{34}$$

which implies

$$\exp(-\epsilon)\lambda^p \leq T_{p,t}(\lambda) \leq \exp(\epsilon)\lambda^p. \tag{35}$$

■

Now we present one of our key lemmas, which provides the backbone for bounding overall error of our sparse factor chain. We claim that substituting $\left(\mathbf{I} + \frac{1}{2}\mathbf{X}\right)^p$ with its Maclaurin expansion into Equation 7 preserves the multiplicative approximation. We prove the lemma for $p \in [-1,1]$ which later we use for computing the p^{th} power of SDDM matrices. For computing the inverse square-root factor, we only need the special case when $p = -1$.

Lemma 24 (Factorization with Matrix Polynomials) *Let $\mathbf{X}$ be a symmetric matrix with $\rho(\mathbf{X}) < 1$ and fix $p \in [-1,1]$. For any approximation parameter $\epsilon > 0$, there exists a t-degree polynomial $T_{p,t}(\cdot)$ with $t = O(\log(1/\epsilon))$, such that*

$$(\mathbf{I}-\mathbf{X})^p \approx_\epsilon T_{-\frac{p}{2},t}\left(\mathbf{I}+\frac{1}{2}\mathbf{X}\right)\left(\mathbf{I}-\frac{1}{2}\mathbf{X}-\frac{1}{2}\mathbf{X}^2\right)^p T_{-\frac{p}{2},t}\left(\mathbf{I}+\frac{1}{2}\mathbf{X}\right). \tag{36}$$

Proof Take the spectral decomposition of $\mathbf{X}$

$$\mathbf{X} = \sum_i \lambda_i \boldsymbol{u}_i \boldsymbol{u}_i^\top, \tag{37}$$

where λ_i are the eigenvalues of $\mathbf{X}$. Then we can represent the left hand side of Equation 36 as

$$\begin{aligned}(\mathbf{I}-\mathbf{X})^p &= \left(\mathbf{I}+\frac{1}{2}\mathbf{X}\right)^{-p/2}\left(\mathbf{I}-\frac{1}{2}\mathbf{X}-\frac{1}{2}\mathbf{X}^2\right)^p\left(\mathbf{I}+\frac{1}{2}\mathbf{X}\right)^{-p/2}\\ &= \sum_i \left(1-\frac{1}{2}\lambda_i-\frac{1}{2}\lambda_i^2\right)\left(1+\frac{1}{2}\lambda_i\right)^{-p}\boldsymbol{u}_i\boldsymbol{u}_i^\top,\end{aligned} \tag{38}$$

and the right hand side of Equation 36 as

$$\begin{aligned}&T_{-\frac{p}{2},t}\left(\mathbf{I}+\frac{1}{2}\mathbf{X}\right)\left(\mathbf{I}-\frac{1}{2}\mathbf{X}-\frac{1}{2}\mathbf{X}^2\right)^p T_{-\frac{p}{2},t}\left(\mathbf{I}+\frac{1}{2}\mathbf{X}\right)\\ &\qquad = \sum_i \left(1-\frac{1}{2}\lambda_i-\frac{1}{2}\lambda_i^2\right)\left(T_{-\frac{p}{2},t}\left(1+\frac{1}{2}\lambda_i\right)\right)^2\boldsymbol{u}_i\boldsymbol{u}_i^\top.\end{aligned} \tag{39}$$

Because $\rho(\mathbf{X})<1$, for all i we have $\left(1-\frac{1}{2}\lambda_i-\frac{1}{2}\lambda_i^2\right)>0$. By invoking Lemma 23 with $\delta=1/2$, error tolerance $\epsilon/2$ and power $-p/2$, we can obtain the polynomial $T_{-\frac{1}{2}p,t}(\cdot)$ with the following property,

$$\exp(-\epsilon)\left(1+\frac{1}{2}\lambda_i\right)^{-p}\le\left(T_{-\frac{1}{2}p,t}\left(1+\frac{1}{2}\lambda_i\right)\right)^2\le\exp(\epsilon)\left(1+\frac{1}{2}\lambda_i\right)^{-p}. \tag{40}$$

To conclude the proof, we use this inequality inside Equation 39, and compare Equation 38 and 39. ■

Appendix C. Spectral Sparsification of Random-Walk Matrix Polynomial

Proposition 25 (Laplacian Preservation) *For any weighted, undirected graph G with adjacency matrix $\mathbf{A}$ and diagonal matrix $\mathbf{D}$, for every non-negative vector $\boldsymbol{\alpha}=(\alpha_1,...,\alpha_d)$ such that with $\sum_{r=1}^d\alpha_r=1$, the random-walk matrix polynomial $\mathbf{L}_{\boldsymbol{\alpha}}(G)$ remains a Laplacian matrix.*

Proof First note that $\mathbf{L}_{\boldsymbol{\alpha}}(G)=\sum_{r=1}^d\alpha_r\mathbf{D}(\mathbf{D}^{-1}\mathbf{A})^r$ is symmetric and has non-positive off-diagonals, so to prove that $\mathbf{L}_{\boldsymbol{\alpha}}(G)$ is also a Laplacian matrix, we only need to show the off-diagonals sum to the diagonal. Fix an integer r and a row index i, we study the i-th row sum S_r of $\mathbf{D}(\mathbf{D}^{-1}\mathbf{A})^r$.

For $r=1$, we have that the row sum S_1 of i-th row of $\mathbf{A}$ gives $S_1=\sum_j\mathbf{A}_{i,j}=\mathbf{D}_{i,i}$. We show that the row sum S_{r+1} can be reduce to S_r as follows,

$$S_{r+1}=\sum_k\left(\left(\mathbf{D}(\mathbf{D}^{-1}\mathbf{A})^r\right)_{i,k}\cdot\mathbf{D}_{k,k}^{-1}\cdot\sum_j\mathbf{A}_{k,j}\right)=\sum_k\left(\mathbf{D}(\mathbf{D}^{-1}\mathbf{A})^r\right)_{i,k}=S_r \tag{41}$$

By induction, we have that $S_n=\cdots=S_1=\mathbf{D}_{i,i}$. Thus, the i-th row sum of $\mathbf{L}_{\boldsymbol{\alpha}}(G)$

$$\sum_j\left(\mathbf{L}_{\boldsymbol{\alpha}}(G)\right)_{i,j}=\sum_{r=1}^t\alpha_rS_r=\mathbf{D}_{i,i}. \tag{42}$$

Therefore, $\mathbf{L}_{\boldsymbol{\alpha}}(G)$ is a Laplacian matrix. ■

Lemma 26 (Two-Step Supports) *For a graph Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$, with diagonal matrix $\mathbf{D}$ and nonnegative off-diagonal $\mathbf{A}$, for all positive odd integer r, we have*

$$\frac{1}{2}\mathbf{L}_G \preceq \mathbf{L}_{G_r} \preceq r\mathbf{L}_G. \tag{43}$$

and for all positive even integers r we have

$$\mathbf{L}_{G_2} \preceq \mathbf{L}_{G_r} \preceq \frac{r}{2}\mathbf{L}_{G_2}. \tag{44}$$

Proof Let $\mathbf{X} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$, for any integer r, the statements are equivalent to

$$\frac{1}{2}\left(\mathbf{I} - \mathbf{X}\right) \preceq \mathbf{I} - \mathbf{X}^{2r+1} \preceq (2r+1)\left(\mathbf{I} - \mathbf{X}\right) \tag{45}$$

$$\mathbf{I} - \mathbf{X}^2 \preceq \mathbf{I} - \mathbf{X}^{2r} \preceq r\left(\mathbf{I} - \mathbf{X}^2\right). \tag{46}$$

Because $\mathbf{X}$ can be diagonalized by unitary matrix $\mathbf{U}$ as $\mathbf{\Lambda} = \mathbf{U}\mathbf{X}\mathbf{U}^\top$, where $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ and $\lambda_i \in [-1, 1]$ for all i. Therefore we can reduce the inequalities to the scalar case, and we conclude the proof with the following inequalities:

$$\begin{aligned} \frac{1}{2}(1-\lambda) \leq 1 - \lambda^{2r+1} \leq (2r+1)(1-\lambda), &\quad \forall \lambda \in (-1,1) \text{ and odd integer } r; \\ (1-\lambda^2) \leq 1 - \lambda^{2r} \leq r(1-\lambda^2), &\quad \forall \lambda \in (-1,1) \text{ and even integer } r. \end{aligned} \tag{47}$$

■

Lemma 27 (Upper Bounds on Effective Resistance) *For a graph G with Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$, let $\mathbf{L}_{G_r} = \mathbf{D} - \mathbf{D}(\mathbf{D}^{-1}\mathbf{A})^r$ be its r-step random-walk matrix. Then, the effective resistance between two vertices u_0 and u_r on $\mathbf{L}_{G_r}$ is upper bounded by $R_{G_r}(u_0, u_r) \leq \sum_{i=1}^{r} \frac{2}{\mathbf{A}(u_{i-1}, u_i)}$, where $(u_0 \ldots u_r)$ is a path in G.*

Proof When r is a positive odd integer, by Lemma 26, we have that $\frac{1}{2}\mathbf{L}_G \preceq \mathbf{L}_{G_r}$, which implies that for any edge (u, v) in G,

$$R_{G_r}(u, v) \leq 2r_G(u, v) = \frac{2}{\mathbf{A}(u, v)}. \tag{48}$$

Because effective resistance satisfies triangular inequality, this concludes the proof for odd r.

When r is even, by Lemma 26, we have that $\mathbf{L}_{G_2} \preceq \mathbf{L}_{G_r}$. The effective resistance of $\mathbf{D} - \mathbf{A}\mathbf{D}^{-1}\mathbf{A}$ is studied in Peng and Spielman (2014) and restated in Proposition 28. For the subgraph $G_2(u)$ anchored at the vertex u (the subgraph formed by length-2 paths where the middle node is u), for any two of its neighbors v_1 and v_2, we have

$$R_{G_r}(v_1, v_2) \leq R_{G_2(u)}(v_1, v_2) = \frac{1}{\mathbf{A}(v_1, u)} + \frac{1}{\mathbf{A}(u, v_2)}. \tag{49}$$

Because we have the above upper bound for any 2-hop path (v_1, u, v_2), by the triangular inequality of effective resistance, the lemma holds for even r as well. ■

Proposition 28 (Claim 6.3. from Peng and Spielman (2014)) *Given a graph of size n with the Laplacian matrix $\mathbf{L} = \mathbf{D} - \frac{1}{d}\boldsymbol{a}\boldsymbol{a}^\top$, where $\mathbf{D}_{i,i} = (a_i s)/d$ with $s = \sum_{i=1}^n a_i$. The effective resistance for edge (i,j) is*

$$\frac{d}{s}(\frac{1}{a_i} + \frac{1}{a_j}). \tag{50}$$

Proof Let $\boldsymbol{e}_i$ denote the vector where the i-th entry is 1, and 0 everywhere else. We have

$$d\mathbf{L}\left(\frac{\boldsymbol{e}_i}{a_i} - \frac{\boldsymbol{e}_j}{a_j}\right) = (s - a_i)\boldsymbol{e}_i - \sum_{k\neq i} a_k \boldsymbol{e}_k - (s - a_j)\boldsymbol{e}_j + \sum_{k\neq j} a_k \boldsymbol{e}_k = s(\boldsymbol{e}_i - \boldsymbol{e}_j). \tag{51}$$

Therefore

$$(\boldsymbol{e}_i - \boldsymbol{e}_j)^\top \mathbf{L}^\dagger (\boldsymbol{e}_i - \boldsymbol{e}_j) = \frac{d}{s}(\boldsymbol{e}_i - \boldsymbol{e}_j)^\top (\frac{\boldsymbol{e}_i}{a_i} - \frac{\boldsymbol{e}_j}{a_j}) = \frac{d}{s}(\frac{1}{a_i} + \frac{1}{a_j}). \tag{52}$$

■

Lemma 29 (A Random-Walk Identity) *Given a graph G with m edges, consider the r-step random-walk graph G_r and the corresponding Laplacian matrix $\mathbf{L}_{G_r} = \mathbf{D} - \mathbf{D}(\mathbf{D}^{-1}\mathbf{A})^r$. For a length-$r$ path $\boldsymbol{p} = (u_0 \dots u_r)$ on G, we have*

$$w(\boldsymbol{p}) = \frac{\prod_{i=1}^r \mathbf{A}(u_{i-1}, u_i)}{\prod_{i=1}^{r-1} \mathbf{D}(u_i, u_i)}, \quad Z(\boldsymbol{p}) = \sum_{i=1}^r \frac{2}{A(u_{i-1}, u_i)}.$$

The summation of $w(\boldsymbol{p})Z(\boldsymbol{p})$ over all length-r paths satisfies

$$\sum_{\boldsymbol{p}} w(\boldsymbol{p}) \cdot Z(\boldsymbol{p}) = 2rm. \tag{53}$$

Proof We substitute the expression for $w(\boldsymbol{p})$ and $Z(\boldsymbol{p})$ in to the summation.

$$\sum_{\boldsymbol{p}=(u_0 \dots u_r)} w(\boldsymbol{p})Z(\boldsymbol{p}) = \sum_{\boldsymbol{p}} \left(\sum_{i=1}^r \frac{2}{\mathbf{A}(u_{i-1}, u_i)}\right)\left(\frac{\prod_{j=1}^r \mathbf{A}(u_{j-1}, u_j)}{\prod_{j=1}^{r-1} \mathbf{D}(u_j, u_j)}\right) \tag{54}$$

$$= 2\sum_{\boldsymbol{p}} \sum_{i=1}^r \left(\frac{\prod_{j=1}^{i-1} \mathbf{A}(u_{j-1}, u_j) \prod_{j=i}^{r-1} \mathbf{A}(u_j, u_{j+1})}{\prod_{j=1}^{r-1} \mathbf{D}(u_j, u_j)}\right) \tag{55}$$

$$= 2\sum_{e\in G} \sum_{i=1}^r \left(\sum_{\boldsymbol{p} \text{ with } (u_{i-1},u_i)=e} \frac{\prod_{j=1}^{i-1} \mathbf{A}(u_{j-1}, u_j) \prod_{j=i}^{r-1} \mathbf{A}(u_j, u_{j+1})}{\prod_{j=1}^{r-1} \mathbf{D}(u_j, u_j)}\right) \tag{56}$$

$$= 2\sum_{e\in G} \sum_{i=1}^r \left(\sum_{\boldsymbol{p} \text{ with } (u_{i-1},u_i)=e} \frac{\prod_{j=1}^{i-1} \mathbf{A}(u_{j-1}, u_j)}{\prod_{j=1}^{i-1} \mathbf{D}(u_j, u_j)} \cdot \frac{\prod_{j=i}^{r-1} \mathbf{A}(u_j, u_{j+1})}{\prod_{j=i}^{r-1} \mathbf{D}(u_j, u_j)}\right) \tag{57}$$

$$= 2\sum_{e\in G} \sum_{i=1}^r 1 \tag{58}$$

$$= 2mr. \tag{59}$$

From Equation 55 to 56, instead of enumerating all paths, we first fix an edge e to be the i-th edge on the path, and then extend from both ends of e. From Equation 57 to 58, we sum over indices iteratively from u_0 to u_{i-1}, and from u_r to u_i. Because $\mathbf{D}(u,u) = \sum_v \mathbf{A}(u,v)$, this summation over all possible paths anchored at (u_{i-1}, u_i) equals to 1. ■

When computing the q^{th} root, the work of chain construction depends quadratically on q, while the multiplication with vector cost the same for all q. Thus it should be only used for $q = O(1)$.

Theorem 30 (Factoring $\mathbf{M}^{-1/q}$) *For any $q \in \mathbb{Z}^+$, and an SDDM matrix $\mathbf{M} = \mathbf{I} - \mathbf{X}_0$, we can construct a representation of an $n \times n$ matrix $\tilde{\mathbf{C}}$ such that $\tilde{\mathbf{C}}\tilde{\mathbf{C}}^\top \approx_\epsilon \mathbf{M}^{-1/q}$ in work $O(m \cdot q^2 \cdot \log^{c_1} n \cdot \log^{c_2}(\kappa/\epsilon)/\epsilon^4)$ and depth $O(q \cdot \log^{c_3} n \cdot \log(\kappa/\epsilon))$ for some constants c_1, c_2, and c_3. Moreover, for any $\boldsymbol{x} \in \mathbb{R}^n$, we can compute $\tilde{\mathbf{C}}\boldsymbol{x}$ in work $O(m + n \cdot \log^c n \cdot \log^3(\kappa/\epsilon)/\epsilon^2)$ and depth $O(\log n \cdot \log(\kappa/\epsilon))$ for some other constant c.*

Appendix D. Extension to SDDM Matrices

Our path sampling algorithm from Section 5 can be generalized to an SDDM matrix with splitting $\mathbf{D} - \mathbf{A}$. The idea is to split out the extra diagonal entries to reduce it back to the Laplacian case. Of course, the $\mathbf{D}^{-1}$ in the middle of the monomial is changed, however, it only decreases the true effective resistance so the upper bound in Lemma 10 still holds without change. The main difference is that we need to put back the extra diagonal entries, which is done by multiplying an all 1 vector through $\mathbf{D} - \mathbf{D}(\mathbf{D}^{-1}\mathbf{A})^r$.

The follow Lemma can be proved similar to Lemma 25.

Lemma 31 (SDDM Preservation) *If $\mathbf{M} = \mathbf{D} - \mathbf{A}$ is an SDDM matrix with diagonal matrix $\mathbf{D}$ and nonnegative off-diagonal $\mathbf{A}$, for any nonnegative $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d)$ with $\sum_{r=1}^d \alpha_r = 1$, $\mathbf{M}_{\boldsymbol{\alpha}} = \mathbf{D} - \sum_{r=1}^d \alpha_r \mathbf{D}(\mathbf{D}^{-1}\mathbf{A})^r$ is also an SDDM matrix.*

Our algorithm is a direction modification of the algorithm from Section 5. To analyze it, we need a variant of Lemma 10 that bounds errors w.r.t. the matrix by which we measure effective resistances. We use the following statement from Peng (2013).

Lemma 32 (Lemma B.0.1. from Peng (2013)) *Let $\mathbf{A} = \sum_i \boldsymbol{y}_e^T \boldsymbol{y}_e$ and $\mathbf{B}$ be $n \times n$ positive semi-definite matrices such that the image space of $\mathbf{A}$ is contained in the image space of $\mathbf{B}$, and τ be a set of estimates such that*

$$\tau_e \geq \boldsymbol{y}_e^T \mathbf{B}^\dagger \boldsymbol{y}_e \qquad \forall e.$$

Then for any error ϵ and any failure probability $\delta = n^{-d}$, there exists a constant c_s such that if we construct $\mathbf{A}$ using the sampling process from Figure 1, with probability at least $1 - \delta = 1 - n^{-d}$, $\tilde{\mathbf{A}}$ satisfies:

$$\mathbf{A} - \epsilon\,(\mathbf{A} + \mathbf{B}) \preceq \tilde{\mathbf{A}} \preceq \mathbf{A} + \epsilon\,(\mathbf{A} + \mathbf{B})\,.$$

Theorem 33 (SDDM Polynomials Sparsification) *Let $\mathbf{M} = \mathbf{D} - \mathbf{A}$ be an SDDM matrix with diagonal $\mathbf{D}$, nonnegative off-diagonal $\mathbf{A}$ with m nonzero entries, for any nonnegative $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d)$ with $\sum_{r=1}^d \alpha_r = 1$, we can define $\mathbf{M}_{\boldsymbol{\alpha}} = \mathbf{D} - \sum_{r=1}^d \alpha_r \mathbf{D}(\mathbf{D}^{-1}\mathbf{A})^r$. For any approximation parameter $\epsilon > 0$, we can construct an SDDM matrix $\tilde{\mathbf{M}}$ with $O(n \log n/\epsilon^2)$ nonzero entries, in time $O(m \cdot \log^2 n \cdot d^2/\epsilon^2)$, such that $\tilde{\mathbf{M}} \approx_\epsilon \mathbf{M}_{\boldsymbol{\alpha}}$.*

Proof

We look at each monomial separately. First, by Lemma 31, $\mathbf{M}_r$ is an SDDM matrix. It can be decomposed as the sum of two matrices, a Laplacian matrix $\mathbf{L}_r = \mathbf{D}_r - \mathbf{D}(\mathbf{D}^{-1}\mathbf{A})^r$, and the remaining diagonal $\mathbf{D}_{\text{extra}}$. As in the Laplacian case, a length-r paths in $\mathbf{D} - \mathbf{A}$ corresponds to an edge in $\mathbf{L}_r$. We apply Lemma 32 to $\mathbf{M}_r$ and $\mathbf{L}_r = \sum_{e \in P} \boldsymbol{y}_e \boldsymbol{y}_e^\top$, where P is the set of all length-r paths in $\mathbf{D} - \mathbf{A}$, and $\boldsymbol{y}_e$ is the column of the incidence matrix associated with e.

When r is an odd integer, we have

$$\boldsymbol{y}_e^\top \left(\mathbf{M}_r\right)^{-1} \boldsymbol{y}_e \leq 2\boldsymbol{y}_e^\top \left(\mathbf{D} - \mathbf{A}\right)^{-1} \boldsymbol{y}_e, \tag{60}$$

and when r is an even integer, we have

$$\boldsymbol{y}_e^\top \left(\mathbf{M}_r\right)^{-1} \boldsymbol{y}_e \leq 2\boldsymbol{y}_e^\top \left(\mathbf{D} - \mathbf{A}\mathbf{D}^{-1}\mathbf{A}\right)^{-1} \boldsymbol{y}_e. \tag{61}$$

Let e denote the edge corresponds to the length-r path $(u_0, \ldots, u_r)$, the weight of e is

$$w(e) = w(u_0 \ldots u_r) = \boldsymbol{y}_e^\top \boldsymbol{y}_e = \frac{\prod_{i=1}^{r} \mathbf{A}(u_{i-1}, u_i)}{\prod_{i=1}^{r-1} \mathbf{D}(u_i, u_i)} \leq \frac{\prod_{i=1}^{r} \mathbf{A}(u_{i-1}, u_i)}{\prod_{i=1}^{r-1} \mathbf{D}_g(u_i, u_i)}, \tag{62}$$

where $\mathbf{D}_g(u, u) = \sum_{v \neq u} \mathbf{A}(u, v)$, so we have the same upper bound as the Laplacian case, and we can sample random walks in the exact same distribution.

By Lemma 32 there exists $M = O(r \cdot m \cdot \log n/\epsilon^2)$ such that with probability at least $1 - \frac{1}{n}$, the sampled graph $\tilde{G} = \textsc{GraphSampling}(G_r, \tau_e, M)$ satisfies

$$\mathbf{M}_r - \frac{1}{2}\epsilon(\mathbf{L}_r + \mathbf{M}_r) \preccurlyeq \mathbf{L}_{\tilde{G}} + \mathbf{D}_{\text{extra}} \preccurlyeq \mathbf{M}_r + \frac{1}{2}\epsilon(\mathbf{L}_r + \mathbf{M}_r). \tag{63}$$

Now if we set $\tilde{\mathbf{M}} = \mathbf{L}_{\tilde{G}} + \mathbf{D}_{\text{extra}}$, we will have

$$(1 - \epsilon)\mathbf{M}_r \preccurlyeq \tilde{\mathbf{M}} \preccurlyeq (1 + \epsilon)\mathbf{M}_r. \tag{64}$$

Note that $\mathbf{D}_{\text{extra}}$ can be computed efficiently by computing $\text{diag}(\mathbf{M}_r\mathbf{1})$ via matrix-vector multiplications. ■

Appendix E. Gremban's Reduction for Inverse Square Root Factor

We first note that Gremban's reduction (Gremban, 1996) can be extended from solving linear systems to computing the inverse square-root factor. The problem of finding a inverse factor of an SDD matrix, can be reduced to finding a inverse factor of an SDDM matrix that is twice as large.

Lemma 34 *Let $\mathbf{\Lambda} = \mathbf{D} + \mathbf{A}_n + \mathbf{A}_p$ be an $n \times n$ SDD matrix, where $\mathbf{D}$ is the diagonal of $\mathbf{\Lambda}$, $\mathbf{A}_p$ contains all the positive off-diagonal entries of $\mathbf{\Lambda}$, and $\mathbf{A}_n$ contains all the negative off-diagonal entries.*

Consider the following SDDM matrix $\mathbf{S}$, and an inverse factor $\tilde{\mathbf{C}}$ of $\mathbf{S}$, such that

$$\mathbf{S} = \begin{bmatrix} \mathbf{D} + \mathbf{A}_n & -\mathbf{A}_p \\ -\mathbf{A}_p & \mathbf{D} + \mathbf{A}_n \end{bmatrix} \text{ and } \tilde{\mathbf{C}}\tilde{\mathbf{C}}^\top = \mathbf{S}. \tag{65}$$

Let $\mathbf{I}_n$ be the $n \times n$ identity matrix. The matrix

$$\mathbf{C} = \frac{1}{\sqrt{2}} \begin{bmatrix} \mathbf{I}_n & -\mathbf{I}_n \end{bmatrix} \tilde{\mathbf{C}}$$

is an inverse square-root factor of $\mathbf{\Lambda}$.

Proof We can prove that

$$\mathbf{\Lambda}^{-1} = \frac{1}{2} \begin{bmatrix} \mathbf{I}_n & -\mathbf{I}_n \end{bmatrix} \mathbf{S}^{-1} \begin{bmatrix} \mathbf{I}_n \\ -\mathbf{I}_n \end{bmatrix} \tag{66}$$

thus we have $\mathbf{C}\mathbf{C}^\top = \mathbf{\Lambda}^{-1}$. ■

Note that if $\tilde{\mathbf{C}}$ is $2n \times 2n$, $\mathbf{C}$ will be an $n \times 2n$ matrix. Given the non-square inverse factor $\mathbf{C}$, we can draw a sample from multivariate Gaussian distribution with covariance $\mathbf{\Lambda}^{-1}$ as follows. First draw $2n$ i.i.d. standard Gaussian random variables $\boldsymbol{x} = (x_1, x_2, \ldots, x_{2n})^\top$, then $\boldsymbol{y} = \mathbf{C}\boldsymbol{x}$ is a multivariate Gaussian random vector with covariance $\mathbf{\Lambda}^{-1}$.

Appendix F. Commutative Splitting of SDDM Matrices

For an SDDM matrix $\mathbf{D} - \mathbf{A}$, one way to normalize it is to multiply $\mathbf{D}^{-1/2}$ on both sides,

$$\mathbf{D} - \mathbf{A} = \mathbf{D}^{1/2} \left(\mathbf{I} - \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2} \right) \mathbf{D}^{1/2}.$$

Because $\rho\left(\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right) < 1$, this enables us to approximate $\left(\mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right)^p$. It is not clear how to undo the normalization and obtain $(\mathbf{D} - \mathbf{A})^p$ from it, due to the fact that $\mathbf{D}$ and $\mathbf{A}$ do not commute. Instead, we consider another normalization as in Lemma 3 and provide its proof.

Lemma 35 (Commutative Splitting) *For an SDDM matrix $\mathbf{M} = \mathbf{D} - \mathbf{A}$ with diagonal $\mathbf{D} = \mathit{diag}\,(d_1, d_2, \ldots, d_n)$, and let $\kappa = \max\{2, \kappa(\mathbf{D} - \mathbf{A})\}$. We can pick $c = (1 - 1/\kappa)/(\max_i d_i)$, so that all eigenvalues of $c(\mathbf{D} - \mathbf{A})$ lie between $\frac{1}{2\kappa}$ and $2 - \frac{1}{2\kappa}$. Moreover, we can rewrite $c(\mathbf{D} - \mathbf{A})$ as an SDDM matrix $\mathbf{I} - \mathbf{X}$, where all entries of $\mathbf{X} = \mathbf{I} - c(\mathbf{D} - \mathbf{A})$ are nonnegative, and $\rho(\mathbf{X}) \leq 1 - \frac{1}{2\kappa}$.*

Proof Let $j = \arg\max_i d_i$, by definition $c = (1 - 1/\kappa)/d_j$.

Let $[\lambda_{\min}, \lambda_{\max}]$ denote the range of the eigenvalues of $\mathbf{D} - \mathbf{A}$. By Gershgorin circle theorem and the fact that $\mathbf{D} - \mathbf{A}$ is diagonally dominant, we have that $\lambda_{\max} \leq 2d_j$. Also $d_j = \boldsymbol{e}_j^\top (\mathbf{D} - \mathbf{A}) \boldsymbol{e}_j$, so $\lambda_{\max} \geq d_j$, and therefore $\lambda_{\min} \geq \lambda_{\max}/\kappa \geq d_j/\kappa$.

The eigenvalues of $c(\mathbf{D} - \mathbf{A})$ lie in the interval $[c\lambda_{\min}, c\lambda_{\max}]$, and can be bounded as

$$\frac{1}{2\kappa} \leq \left(1 - \frac{1}{\kappa}\right) \frac{1}{d_j} \left(\frac{d_j}{\kappa}\right) \leq c\lambda_{\min} \leq c\lambda_{\max} \leq \left(1 - \frac{1}{\kappa}\right) \frac{1}{d_j} (2d_j) \leq 2 - \frac{1}{2\kappa}. \tag{67}$$

To see that $\mathbf{X} = (\mathbf{I} - c\mathbf{D}) + c\mathbf{A}$ is a nonnegative matrix, note that $\mathbf{A}$ is nonnegative and $\mathbf{I}$ is entry-wise larger than $c\mathbf{D}$. ■

Appendix G. Inverse Factor by Combinatorial SDDM Factorization

A simple combinatorial construction of $\tilde{\mathbf{C}}$ follows from the fact that SDDM matrices can be factorized.

Lemma 36 (Combinatorial Factorization) *Let $\mathbf{M}$ be an SDDM matrix with m non-zero entries and condition number κ, then*

(1) $\mathbf{M}$ can be factored into

$$\mathbf{M} = \mathbf{B}\mathbf{B}^\top,$$

where $\mathbf{B}$ is an $n \times m$ matrix with at most 2 non-zeros per column;

(2) for any approximation parameter $\epsilon > 0$, if $\mathbf{Z}$ is linear operator satisfying $\mathbf{Z} \approx_\epsilon \mathbf{M}^{-1}$, then the matrix

$$\tilde{\mathbf{C}} = \mathbf{Z}\mathbf{B}$$

satisfies

$$\tilde{\mathbf{C}}\tilde{\mathbf{C}}^\top \approx_{2\epsilon} \mathbf{M}^{-1}.$$

Moreover, a representation of $\tilde{\mathbf{C}}$ can be computed in work $O(m \cdot \log^{c_1} n \cdot \log^{c_2} \kappa)$ and depth $O(\log^{c_3} n \cdot \log \kappa)$ for some modest constants c_1, c_2, and c_3 so that for any m-dimensional vector $\boldsymbol{x}$, the product $\tilde{\mathbf{C}}\boldsymbol{x}$ can be computed in work $O((m + n \cdot \log^c n \cdot \log^3 \kappa) \cdot \log(1/\epsilon))$ and depth $O(\log n \cdot \log \kappa \cdot \log(1/\epsilon))$ for a modest constant c.

Proof Let $\boldsymbol{e}_i$ be the i^{th} standard basis, then $\mathbf{M}$ can be represented by

$$\mathbf{M} = \frac{1}{2} \sum_{1 \le i < j \le n} |\mathbf{M}_{i,j}| \left(\boldsymbol{e}_i - \boldsymbol{e}_j\right)\left(\boldsymbol{e}_i - \boldsymbol{e}_j\right)^\top + \sum_{i=1}^{n} a_i \boldsymbol{e}_i \boldsymbol{e}_i^\top,$$

where $a_i = \mathbf{M}_{i,j} - \sum_{j \neq i} |\mathbf{M}_{i,j}| \geq 0$. Indeed, we can rewrite $\mathbf{M}$ as $\sum_{i=1}^{m} \boldsymbol{y}_i \boldsymbol{y}_i^\top$ this way, and by setting $\mathbf{B} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_m)$ concludes the proof for statement (1).

For statement (2),

$$\begin{aligned} \tilde{\mathbf{C}}\tilde{\mathbf{C}}^\top &= \mathbf{Z}\mathbf{B}\mathbf{B}^\top\mathbf{Z}^\top \\ &= \mathbf{Z}\mathbf{M}\mathbf{Z} \\ &\approx_\epsilon \mathbf{Z}\mathbf{Z}^{-1}\mathbf{Z} = \mathbf{Z}. \end{aligned} \tag{68}$$

The third line is true because of Fact 4.d and Fact 4.e. From $\tilde{\mathbf{C}}\tilde{\mathbf{C}}^\top \approx_\epsilon \mathbf{Z}$, by Fact 4.c, we have $\tilde{\mathbf{C}}\tilde{\mathbf{C}}^\top \approx_{2\epsilon} \mathbf{M}^{-1}$.

The total work and depth of constructing the operator $\mathbf{Z}\mathbf{B}$ is dominated by the work and depth of constructing the approximate inverse $\mathbf{Z}$, which we refer to the analysis in Peng and Spielman (2014). The same statement also holds for the work and depth of matrix-vector multiplication of $\mathbf{Z}\mathbf{B}$. ■

In particular, the parallel solver algorithm from Peng and Spielman (2014) computes $\mathbf{Z} \approx_\epsilon \mathbf{M}^{-1}$ in total work nearly-linear in m, and depth polylogarithmic in n, κ and ϵ^{-1}. However, it requires m i.i.d. standard Gaussian samples to generate an n-dimensional Gaussian random vector.