Gauss quadrature for matrix inverse forms with applications

Chengtao Li ctli@mit.edu
Suvrit Sra suvrit@mit.edu
Stefanie Jegelka stefje@csail.mit.edu
Massachusetts Institute of Technology, Cambridge, MA 02139

Abstract

We present a framework for accelerating a spectrum of machine learning algorithms that require computation of *bilinear inverse forms* $u^\top A^{-1}u$, where A is a positive definite matrix and u a given vector. Our framework is built on Gauss-type quadrature and easily scales to large, sparse matrices. Further, it allows retrospective computation of lower and upper bounds on $u^\top A^{-1}u$, which in turn accelerates several algorithms. We prove that these bounds tighten iteratively and converge at a linear (geometric) rate. To our knowledge, ours is the first work to demonstrate these key properties of Gauss-type quadrature, which is a classical and deeply studied topic. We illustrate empirical consequences of our results by using quadrature to accelerate machine learning tasks involving determinantal point processes and submodular optimization, and observe tremendous speedups in several instances.

1 Introduction

Symmetric positive definite matrices arise in many areas in a variety of guises: covariances, kernels, graph Laplacians, or otherwise. A basic computation with such matrices is evaluation of the bilinear form $u^T f(A)v$, where f is a matrix function and u, v are given vectors. If $f(A) = A^{-1}$, we speak of computing a *bilinear inverse form (BIF)* $u^T A^{-1}v$. For example, with $u{=}v{=}e_i$ (i^{th} canonical vector) $u^T f(A)v = (A^{-1})_{ii}$ is the i^{th} diagonal entry of the inverse.

In this paper, we are interested in efficiently computing BIFs, primarily due to their importance in several machine learning contexts, e.g., evaluation of Gaussian density at a point, the Woodbury matrix inversion lemma, implementation of MCMC samplers for Determinantal Point Processes (Dpp), computation of graph centrality measures, and greedy submodular maximization (see Section 2).

When A is large, it is preferable to compute $u^T A^{-1}v$ iteratively rather than to first compute A^{-1} (using Cholesky) at a cost of $\mathcal{O}(N^3)$ operations. One could think of using conjugate gradients to solve $Ax = v$ approximately, and then obtain $u^T A^{-1}v = u^T x$. But several applications require precise bounds on numerical estimates to $u^\top A^{-1}v$ (e.g., in MCMC based Dpp samplers such bounds help decide whether to accept or reject a transition in each iteration–see Section 5.1), which necessitates a more finessed approach.

Gauss quadrature is one such approach. Originally proposed in [23] for approximating integrals, Gauss- and *Gauss-type quadrature* (i.e., Gauss-Lobatto [43] and Gauss-Radau [51] quadrature) have since found application to bilinear forms including computation of $u^T A^{-1}v$ [4]. Bai et al. also show that Gauss and (right) Gauss-Radau quadrature yield lower bounds, while Gauss-Lobatto and (left) Gauss-Radau yield upper bounds on the BIF $u^T A^{-1}v$.

However, despite its long history and voluminous existing work (see e.g., [29]), our understanding of Gauss-type quadrature for matrix problems is far from complete. For instance, it is not known whether the bounds on BIFs improve with more quadrature iterations; nor is it known how the bounds obtained from Gauss, Gauss-Radau and Gauss-Lobatto quadrature compare with each other. *We do not even know how fast the iterates of Gauss-Radau or Gauss-Lobatto quadrature converge.*

Contributions. We address all the aforementioned problems and make the following main contributions:

– We show that the lower and upper bounds generated by Gauss-type quadrature monotonically approach the target value (Theorems 4 and 6; Corr. 7). Furthermore, we show that for the same number of iterations, Gauss-Radau quadrature yields bounds superior to those given by Gauss or Gauss-Lobatto, but somewhat surprisingly all three share the same convergence rate.
– We prove linear convergence rates for Gauss-Radau and Gauss-Lobatto explicitly (Theorems 5 and 8; Corr. 9).
– We demonstrate implications of our results for two tasks: (i) scalable Markov chain sampling from a DPP; and (ii) running a greedy algorithm for submodular optimization. In these applications, quadrature accelerates computations, and the bounds aid early stopping.

Indeed, on large-scale sparse problems our methods lead to even several orders of magnitude in speedup.

Related Work. There exist a number of methods for efficiently approximating matrix bilinear forms. Brezinski [12] and Brezinski et al. [13] use extrapolation of matrix moments and interpolation to estimate the 2-norm error of linear systems and the trace of the matrix inverse. Fika et al. [20] extend the extrapolation method to BIFs and show that the derived one-term and two-term approximations coincide with Gauss quadrature, hence providing lower bounds. Further generalizations address $x^* f(A) y$ for a Hermitian matrix A [19]. In addition, other methods exist for estimating trace of a matrix function [3, 13, 18] or diagonal elements of matrix inverse [5, 60].

Many of these methods may be applied to computing BIFs. But they do not provide intervals bounding the target value, just approximations. Thus, a black-box use of these methods may change the execution of an algorithm whose decisions (e.g., whether to transit in a Markov Chain) rely on the BIF value to be within a specific interval. Such changes can break the correctness of the algorithm.

Our framework, in contrast, yields iteratively tighter lower and upper bounds (Section 4), so the algorithm is guaranteed to make correct decisions (Section 5).

2 Motivating Applications

BIFs are important to numerous problems. We recount below several notable examples: in all cases, efficient computation of bounds on BIFs is key to making the algorithms practical.

Determinantal Point Processes. A determinantal point process (DPP) is a distribution over subsets of a set $\mathcal{Y}$ ($|\mathcal{Y}| = N$). In its *L-ensemble* form, a DPP uses a positive semidefinite kernel $L \in \mathbb{R}^{N \times N}$, and to a set $Y \subseteq \mathcal{Y}$ assigns probability $P(Y) \propto \det(L_Y)$ where L_Y is the submatrix of L indexed by entries in Y. If we restrict to $|Y| = k$, we obtain a k-DPP. DPP's are widely used in machine learning, see e.g., the survey [36].

Exact sampling from a (k-)DPP requires eigendecomposition of L [33], which is prohibitive. For large N, Metropolis Hastings (MH) or Gibbs sampling are preferred and state-of-the-art. Therein the core task is to compute transition probabilities – an expression involving BIFs – which are compared with a random scalar threshold.

For MH [1, 7], the transition probabilities from a current subset (state) Y to Y' are $\min\{1, L_{u,u} - L_{u,Y} L_Y^{-1} L_{Y,u}\}$ for $Y' = Y \cup \{u\}$; and $\min\{1, L_{u,u} - L_{u,Y'} L_{Y'}^{-1} L_{Y',u}\}$ for $Y' = Y \backslash \{u\}$. In a k-DPP, the moves are swaps with transition probabilities $\min\left\{1, \frac{L_{u,u} - L_{u,Y'} L_{Y'}^{-1} L_{Y',u}}{L_{v,v} - L_{v,Y'} L_{Y'}^{-1} L_{Y',v}}\right\}$ for replacing $v \in Y$ by $u \notin Y$ (and $Y' = Y \backslash \{v\}$). We illustrate this application in greater detail in Section 5.1.

DPPs are also useful for (repulsive) priors in Bayesian models [37, 53]. Inference for such latent variable models uses Gibbs sampling, which again involves BIFs.

Submodular optimization, Sensing. Algorithms for maximizing submodular functions can equally benefit from efficient BIF bounds. Given a positive definite matrix $K \in \mathbb{R}^{N \times N}$, the set function $F(S) = \log\det(K_S)$ is *submodular*: for all $S \subseteq T \subseteq [N]$ and $i \in [N] \setminus T$, it holds that $F(S \cup \{i\}) - F(S) \geq F(T \cup \{i\}) - F(T)$.

Finding the set $S^* \subseteq [N]$ that maximizes $F(S)$ is a key task for MAP inference with DPPs [25], matrix approximations by column selection [11, 59] and sensing Krause et al. [35]. For the latter, we model spatial phenomena (temperature, pollution) via Gaussian Processes and select locations to maximize the joint entropy $F_1(S) = H(X_S) = \log\det(K_S) + \text{const}$ of the observed variables, or the mutual information $F_2(S) = I(X_S; X_{[N]\setminus S})$ between observed and unobserved variables.

Greedy algorithms for maximizing monotone [49] or non-monotone [14] submodular functions rely on marginal gains of the form

$$\begin{aligned} F_1(S \cup \{i\}) - F_1(S) &= \log(K_i - K_{iS}K_S^{-1}K_{Si}); \\ F_1(T \setminus \{i\}) - F_1(T) &= -\log(K_i - K_{iU}K_U^{-1}K_{Ui}); \\ F_2(S \cup \{i\}) - F_2(S) &= \log \tfrac{K_i - K_{iS}K_S^{-1}K_{Si}}{K_i - K_{i\bar{S}}K_{\bar{S}}^{-1}K_{\bar{S}i}} \end{aligned}$$

for $U = T\setminus\{i\}$ and $\bar{S} = [N]\setminus S$. The algorithms compare those gains to a random threshold, or find an item with the largest gain. In both cases, efficient BIF bounds offer speedups. They can be combined with lazy [47] and stochastic greedy algorithms [48].

Network Analysis, Centrality. When analyzing relationships and information flows between connected entities in a network, such as people, organizations, computers, smart hardwares, etc. [2, 9, 16, 17, 40, 54], an important question is to measure popularity, centrality, or importance of a node.

Several existing popularity measures can be expressed as the solution to a large-scale linear system. For example, *PageRank* [50] is the solution to $(I - (1-\alpha)A^\top)x = \alpha\mathbf{1}/N$, and *Bonacich centrality* [10] is the solution to $(I - \alpha A)x = \mathbf{1}$, where A is the adjacency matrix. When computing local estimates, i.e., only a few entries of x, we obtain exactly the task of computing BIFs [39, 61]. Moreover, we may only need local estimates to an accuracy sufficient for determining which entry is larger, a setting where our quadrature based bounds on BIFs will be useful.

Scientific Computing. In computational physics BIFs are used for estimating selected entries of the inverse of a large sparse matrix. More generally, BIFs can help in estimating the trace of the inverse, a computational substep in lattice Quantum Chromodynamics [15, 22], some signal processing tasks [31], and in Gaussian Process (GP) Regression [52], e.g., for estimating variances. In numerical linear algebra, BIFs are used in rational approximations [57], evaluation of Green's function [21], and selective inversion of sparse matrices [39, 41, 42]. A notable use is the design of preconditioners [8] and uncertainty quantification [6].

Benefiting from fast iterative bounds. Many of the above examples use BIFs to rank values, to identify the largest value or compare them to a scalar or to each other. In such cases, we first compute fast, crude lower and upper bounds on a BIF, refining iteratively, just as far as needed to determine the comparison. Figure 1 in Section 4.4 illustrates the evolution of these bounds, and Section 5 explains details.

3 Background on Gauss Quadrature

For convenience, we begin by recalling key aspects of Gauss quadrature,[1] as applied to computing $u^\top f(A)v$, for an $N \times N$ symmetric positive definite matrix A that has *simple* eigenvalues, arbitrary vectors u, v, and a matrix function f. For a more detailed account of the relevant background on Gauss-type quadratures please refer to Appendix A, or [29].

It suffices to consider $u^\top f(A)u$ thanks to the identity

$$u^\top f(A)v = \tfrac{1}{4}(u+v)^\top f(A)(u+v) - \tfrac{1}{4}(u-v)^\top f(A)(u-v).$$

[1]The summary in this section is derived from various sources: [4, 24, 29]. Experts can skim this section for collecting our notation before moving onto Section 4, which contains our new results.

Let $A = Q^\top \Lambda Q$ be the eigendecomposition of A where Q is orthonormal. Letting $\tilde{u} = Qu$, we then have

$$u^\top f(A)u = \tilde{u}^\top f(\Lambda)\tilde{u} = \sum_{i=1}^{N} f(\lambda_i)\tilde{u}_i^2.$$

Toward computing $u^T f(A)u$, a key conceptual step is to write the above sum as the Riemann-Stieltjes integral

$$I[f] := u^\top f(A)u = \int_{\lambda_{\min}}^{\lambda_{\max}} f(\lambda)d\alpha(\lambda), \tag{3.1}$$

where $\lambda_{\min} \in (0, \lambda_1)$, $\lambda_{\max} > \lambda_N$, and $\alpha(\lambda)$ is piecewise constant measure defined by

$$\alpha(\lambda) := \begin{cases} 0, & \lambda < \lambda_1, \\ \sum_{j=1}^{k} \tilde{u}_j^2, & \lambda_k \leq \lambda < \lambda_{k+1}, \quad k < N, \\ \sum_{j=1}^{N} \tilde{u}_j^2, & \lambda_N \leq \lambda. \end{cases}$$

Our task now reduces to approximating the integral (3.1), for which we invoke the powerful idea of Gauss-type quadratures [23, 24, 43, 51]. We rewrite the integral (3.1) as

$$I[f] := Q_n + R_n = \sum_{i=1}^{n} \omega_i f(\theta_i) + \sum_{i=1}^{m} \nu_i f(\tau_i) + R_n[f], \tag{3.2}$$

where Q_n denotes the nth degree approximation and R_n denotes the remainder term. In representation (3.2) the *weights* $\{\omega_i\}_{i=1}^n$, $\{\nu_i\}_{i=1}^m$, and quadrature *nodes* $\{\theta_i\}_{i=1}^n$ are unknown, while the values $\{\tau_i\}_{i=1}^m$ are prescribed and lie outside the interval of integration $(\lambda_{\min}, \lambda_{\max})$.

Different choices of these parameters yield different quadrature rules: $m = 0$ gives Gauss quadrature [23]; $m = 1$ with $\tau_1 = \lambda_{\min}$ ($\tau_1 = \lambda_{\max}$) gives left (right) Gauss-Radau quadrature [51]; $m = 2$ with $\tau_1 = \lambda_{\min}$ and $\tau_2 = \lambda_{\max}$ yields Gauss-Lobatto quadrature [43]; while for general m we obtain Gauss-Christoffel quadrature [24].

The weights $\{\omega_i\}_{i=1}^n$, $\{\nu_i\}_{i=1}^m$ and nodes $\{\theta_i\}_{i=1}^n$ are chosen such that if f is a polynomial of degree less than $2n + m - 1$, then the interpolation $I[f] = Q_n$ is *exact*. For Gauss quadrature, we can recursively build the *Jacobi matrix*

$$J_n = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \beta_2 & \ddots & \ddots & \\ & & \ddots & \alpha_{n-1} & \beta_{n-1} \\ & & & \beta_{n-1} & \alpha_n \end{bmatrix}, \tag{3.3}$$

and obtain from its spectrum the desired weights and nodes. Theorem 1 makes this more precise.

Theorem 1. *[30, 62] The eigenvalues of J_n form the nodes $\{\theta_i\}_{i=1}^n$ of Gauss quadrature; the weights $\{\omega_i\}_{i=1}^n$ are given by the squares of the first components of the eigenvectors of J_n.*

If J_n has the eigendecomposition $P_n^\top \Gamma P_n$, then for Gauss quadrature Thm. 1 yields

$$Q_n = \sum_{i=1}^{n} \omega_i f(\theta_i) = e_1^\top P_n^\top f(\Gamma) P_n e_1 = e_1^\top f(J_n) e_1. \tag{3.4}$$

Given A and u, our task is to compute Q_n and the Jacobi matrix J_n. For BIFs, we have that $f(J_n) = J_n^{-1}$, so (3.4) becomes $Q_n = e_1^T J_n^{-1} e_1$, which can be computed recursively using the Lanczos algorithm [38]. For Gauss-Radau and Gauss-Lobatto quadrature we can compute modified versions of Jacobi matrices J_n^{lr} (for left Gauss-Radau), J_n^{rr} (for right Gauss-Radau) and J_n^{lo} (for Gauss-Lobatto) based on J_n. The corresponding nodes and weights, and thus the approximation of Gauss-Radau and Gauss-Lobatto quadratures, are then obtained from these modified Jacobi matrices, similar to Gauss quadrature. Aggregating all these computations yields an algorithm that iteratively obtains bounds on $u^T A^{-1} u$. The combined procedure, *Gauss Quadrature Lanczos (GQL)* [28], is summarily presented as Algorithm 1. The complete algorithm may be found in Appendix A.

Algorithm 1 Gauss Quadrature Lanczos (GQL)

Input: Matrix A, vector u; lower and upper bounds $\lambda_{\min}$ and $\lambda_{\max}$ on the spectrum of A
Output: $(g_i, g_i^{\mathrm{rr}}, g_i^{\mathrm{lr}}, g_i^{\mathrm{lo}})$: Gauss, right Gauss-Radau, left Gauss-Radau, and Gauss-Lobatto quadrature estimates for each i
Initialize: $u_0 = u/\|u\|$, $g_1 = \|u\|/u_0^\top A u_0$, $i = 2$
for $i = 1$ to N **do**
  Update J_i using a Lanczos iteration
  Solve for the modified Jacobi matrices J_i^{lr}, J_i^{rr} and J_i^{lo}.
  Compute g_i, g_i^{rr}, g_i^{lr} and g_i^{lo} with Sherman-Morrison formula.
end for

Theorem 2. *[44] Let g_i, g_i^{lr}, g_i^{rr}, and g_i^{lo} be the i-th iterates of Gauss, left Gauss-Radau, right Gauss-Radau, and Gauss-Lobatto quadrature, respectively, as computed by Alg. 1. Then, g_i and g_i^{rr} provide lower bounds on $u^\top A^{-1} u$, while g_i^{lr} and g_i^{lo} provide upper bounds.*

It turns out that the bounds given by Gauss quadrature have a close relation to the approximation error of conjugate gradient (CG) applied to a suitable problem. Since we know the convergence rate of CG, we can obtain from it the following estimate on the *relative error* of Gauss quadrature.

Theorem 3 (Relative error Gauss quadrature)**.** *The i-th iterate of Gauss quadrature satisfies the relative error bound*

$$\frac{g_N - g_i}{g_N} \leq 2\Big(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\Big)^i, \tag{3.5}$$

where $\kappa := \lambda_1(A)/\lambda_N(A)$ is the condition number of A.

In other words, Thm. 3 shows that the iterates of Gauss quadrature have a linear (geometric) convergence rate.

4 Main Theoretical Results

In this section we summarize our main theoretical results. As before, detailed proofs may be found in Appendix B. The key questions that we answer are: (i) do the bounds on $u^\top A^{-1} u$ generated by GQL improve monotonically with each iteration; (ii) how tight are these bounds; and (iii) how fast do Gauss-Radau and Gauss-Lobatto iterations converge? Our answers not only fill gaps in the literature on quadrature, but provide a theoretical base for speeding up algorithms for some applications (see Sections 2 and 5).

4.1 Lower Bounds

Our first result shows that both Gauss and right Gauss-Radau quadratures give iteratively better lower bounds on $u^\top A^{-1} u$. Moreover, with the same number of iterations, right Gauss-Radau yields tighter bounds.

Theorem 4. *Let $i < N$. Then, g_i^{rr} yields better bounds than g_i but worse bounds than g_{i+1}; more precisely,*

$$g_i \leq g_i^{rr} \leq g_{i+1}, \quad i < N.$$

Combining Theorem 4 with the convergence rate of relative error for Gauss quadrature (Thm. 3) we obtain the following convergence rate estimate for right Gauss-Radau.

Theorem 5 (Relative error right Gauss-Radau)**.** *For each iteration i, the right Gauss-Radau iterate g_i^{rr} satisfies*

$$\frac{g_N - g_i^{rr}}{g_N} \leq 2\Big(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\Big)^i.$$

4.2 Upper Bounds

Our second result compares Gauss-Lobatto with left Gauss-Radau quadrature.

Theorem 6. *Let $i < N$. Then, g_i^{lr} gives better upper bounds than g_i^{lo} but worse than g_{i+1}^{lo}; more precisely,*

$$g_{i+1}^{lo} \leq g_i^{lr} \leq g_i^{lo}, \quad i < N.$$

This shows that bounds given by both Gauss-Lobatto and left Gauss-Radau become tighter with each iteration. For the same number of iterations, left Gauss-Radau provides a tighter bound than Gauss-Lobatto.

Combining the above two theorems, we obtain the following corollary for all four Gauss-type quadratures.

Corollary 7 (Monotonicity)**.** *With increasing i, g_i and g_i^{rr} give increasingly better lower bounds and g_i^{lr} and g_i^{lo} give increasingly better upper bounds, that is,*

$$g_i \leq g_{i+1}; \quad g_i^{rr} \leq g_{i+1}^{rr};$$
$$g_i^{lr} \geq g_{i+1}^{lr}; \quad g_i^{lo} \geq g_{i+1}^{lo}.$$

4.3 Convergence rates

Our next two results state linear convergence rates for left Gauss-Radau quadrature and Gauss-Lobatto quadrature applied to computing the BIF $u^T A^{-1} u$.

Theorem 8 (Relative error left Gauss-Radau)**.** *For each i, the left Gauss-Radau iterate g_i^{lr} satisfies*

$$\frac{g_i^{lr} - g_N}{g_N} \leq 2\kappa^+ \Big(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \Big)^i,$$

where $\kappa^+ := \lambda_N / \lambda_{\min}$, $i < N$.

Theorem 8 shows that the error again decreases linearly, and it also depends on teh accuracy of $\lambda_{\min}$, our estimate of the smallest eigenvalue that determines the range of integration. Using the relations between left Gauss-Radau and Gauss-Lobatto, we readily obtain the following corollary.

Corollary 9 (Relative error Gauss-Lobatto)**.** *For each i, the Gauss-Lobatto iterate g_i^{lo} satisfies*

$$\frac{g_i^{lo} - g_N}{g_N} \leq 2\kappa^+ \Big(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \Big)^{i-1},$$

where $\kappa^+ := \lambda_N / \lambda_{\min}$ and $i < N$.

Remarks All aforementioned results assumed that A is strictly positive definite with simple eigenvalues. In Appendix C, we show similar results for the more general case that A is only required to be symmetric, and u lies in the space spanned by eigenvectors of A corresponding to distinct positive eigenvalues.

4.4 Empirical Evidence

Next, we empirically verify our the theoretical results shown above. We generate a random symmetric matrix $A \in \mathbb{R}^{100 \times 100}$ with density 10%, where each entry is either zero or standard normal, and shift its diagonal entries to make its smallest eigenvalue $\lambda_1 = 10^{-2}$, thus making A positive definite. We set $\lambda_{\min} = \lambda_1^- = (\lambda_1 - 10^{-5})$ and $\lambda_{\max} = \lambda_N^+ = (\lambda_N + 10^{-5})$. We randomly sample $u \in \mathbb{R}^{100}$ from a

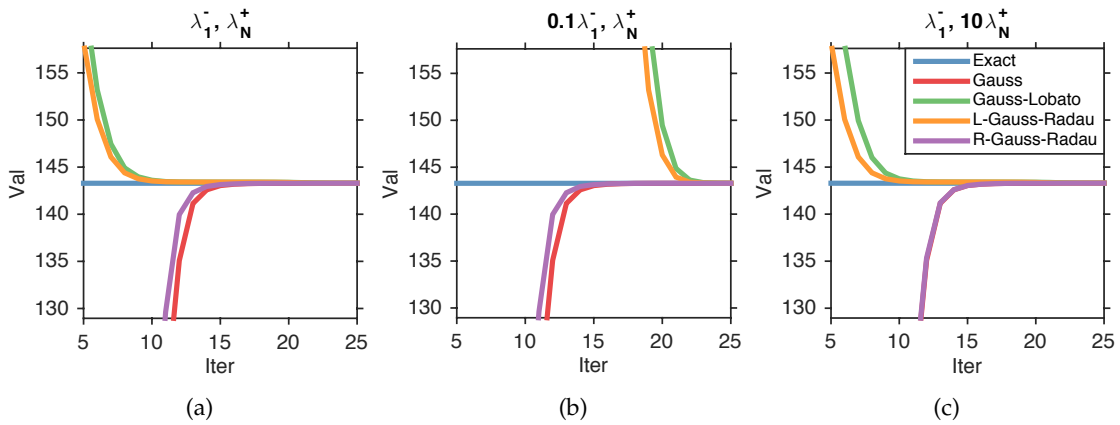

Figure 1: Lower and upper bounds computed by Gauss-type quadrature in each iteration on $u^\top A^{-1}u$ with $A \in \mathbb{R}^{100\times 100}$.

standard normal distribution. Figure 1 illustrates how the lower and upper bounds given by the four quadrature rules evolve with the number of iterations.

Figure 1 (b) and (c) show the sensitivity of the rules (except Gauss quadrature) to estimating the extremal eigenvalues. Specifically, we use $\lambda_{\min} = 0.1\lambda_1^-$ and $\lambda_{\max} = 10\lambda_N^+$.

The plots in Figure 1 agree with the theoretical results. First, all quadrature rules are seen to yield iteratively tighter bounds. The bounds obtained by the Gauss-Radau quadrature are superior to those given by Gauss and Gauss-Lobatto quadrature (also numerically verified). Notably, the bounds given by all quadrature rules converge very fast – within 25 iterations they yield reasonably tight bounds.

It is valuable to see how the bounds are affected if we do not have good approximations to the extremal eigenvalues λ_1 and λ_N. Since Gauss quadrature does not depend on the approximations $\lambda_{\min} < \lambda_1$ and $\lambda_{\max} > \lambda_N$, its bounds remain the same in (a),(b),(c). Left Gauss-Radau depends on the quality of $\lambda_{\min}$, and, with a poor approximation takes more iterations to converge (Figure 1(b)). Right Gauss-Radau depends on the quality of $\lambda_{\max}$; thus, if we use $\lambda_{\max} = 10\lambda_N^+$ as our approximation, its bounds become worse (Figure 1(c)). However, its bounds are never worse than those obtained by Gauss quadrature.

Finally, Gauss-Lobatto depends on both $\lambda_{\min}$ and $\lambda_{\max}$, so its bounds become worse whenever we lack good approximations to λ_1 or λ_N. Nevertheless, its quality is lower-bounded by left Gauss-Radau as stated in Thm. 6.

5 Algorithmic Results and Applications

Our theoretical results show that Gauss-Radau quadrature provides good lower and upper bounds to BIFs. More importantly, these bounds get iteratively tighter at a linear rate, finally becoming exact (see Appendix B). However, in many applications motivating our work (see Section 2), we do not need exact values of BIFs; bounds that are tight enough suffice for the algorithms to proceed. As a result, all these applications benefit from our theoretical results that provide iteratively tighter bounds. This idea translates into a *retrospective* framework for accelerating methods whose progress relies on knowing an interval containing the BIF. Whenever the algorithm takes a step (*transition*) that depends on a BIF (e.g., as in the next section, a state transition in a sampler if the BIF exceeds a certain threshold), we compute rough bounds on its value. If the bounds suffice to take the critical decision (e.g., decide the comparison), then we stop the quadrature. If they do not suffice, we take one or more additional iterations of quadrature to tighten the bound. Algorithm 2 makes this idea explicit.

We illustrate our framework by accelerating: (i) Markov chain sampling for (k-)DPPs; and (ii) maxi-

Algorithm 2 Efficient Retrospective Framework

Require: Algorithm with transitions that depend on BIFs
  while algorithm not yet done **do**
    while no transition request for values of a BIF **do**
      proceed with the original algorithm
    end while
    if exist transition request for values of a BIF **then**
      while bounds on the BIF not tight enough to make the transition **do**
        Retrospectively run one more iteration of left and(or) right Gauss-Radau to obtain tighter bounds.
      end while
      Make the correct transition with bounds
    end if
  end while

mization of a (specific) nonmonotone submodular function.

5.1 Retrospective Markov Chain (k-)Dpp

First, we use our framework to accelerate iterative samplers for Determinantal Point Processes. Specifically, we discuss MH sampling [34]; the variant for Gibbs sampling follows analogously.

The key insight is that all state transitions of the Markov chain rely on a comparison between a scalar p and a quantity involving the bilinear inverse form. Given the current set Y, assume we propose to add element y to Y. The probability of transitioning to state $Y \cup \{y\}$ is $q = \min\{1, L_{y,y} - L_{y,Y} L_Y^{-1} L_{Y,y}\}$. To decide whether to accept this transition, we sample $p \sim \text{Uniform}(0,1)$; if $p < q$ then we accept the transition, otherwise we remain at Y. Hence, we need to compute q just accurately enough to decide whether $p < q$. To do so, we can use the aforementioned lower and upper bounds on $L_{y,Y} L_Y^{-1} L_{Y,y}$.

Let s_i and t_i be lower and upper bounds for this BIF in the i-th iteration of Gauss quadrature. If $p \leq L_{y,y} - t_i$, then we can safely accept the transition, if $p \geq L_{y,y} - s_i$, then we can safely reject the transition. Only if $L_{y,y} - t_i < p < L_{y,y} - s_i$, we cannot make a decision yet, and therefore retrospectively perform one more iteration of Gauss quadrature to obtain tighter upper and lower bounds s_{i+1} and t_{i+1}. We continue until the bounds are sharp enough to safely decide whether to make the transition. Note that in each iteration we make the same decision as we would with the exact value of the BIF, and hence the resulting algorithm (Alg. 3) is an exact Markov chain for the Dpp. In each iteration, it calls Alg. 4, which uses step-wise lazy Gauss quadrature for deciding the comparison, while stopping as early as possible.

If we condition the Dpp on observing a set of a fixed cardinality k, we obtain a k-Dpp. The MH sampler for this process is similar, but a state transition corresponds to swapping two elements (adding y and removing v at the same time). Assume the current set is $Y = Y' \cup \{v\}$. If we propose to delete v and add y to Y', then the corresponding transition probability is

$$q = \min\left\{1, \frac{L_{y,y} - L_{y,Y'} L_{Y'}^{-1} L_{Y',y}}{L_{v,v} - L_{v,Y'} L_{Y'}^{-1} L_{Y',v}}\right\}. \tag{5.1}$$

Again, we sample $p \sim \text{Uniform}(0,1)$, but now we must compute two quantities, and hence two sets of lower and upper bounds: s_i^y, t_i^y for $L_{y,Y'} L_{Y'}^{-1} L_{Y',y}$ in the i-th Gauss quadrature iteration, and s_j^v, t_j^v for $L_{v,Y'} L_{Y'}^{-1} L_{Y',v}$ in the j-th Gauss quadrature iteration. Then if we have $p \leq \frac{L_{y,y} - t_i^y}{L_{v,v} - s_j^v}$, we can safely accept the transition; and if $p \geq \frac{L_{y,y} - s_i^y}{L_{v,v} - t_j^v}$ we can safely reject the transition; otherwise, we tighten the bounds via additional Gauss-Radau iterations.

Refinements. We could perform one iteration for both y and v, but it may be that one set of bounds is already sufficiently tight, while the other is loose. A straightforward idea would be to judge the

Algorithm 3 Gauss-DPP (L)

Require: DPP kernel L; ground set $\mathcal{Y}$
Ensure: Y sampled from exact DPP (L)
  Randomly Initialize $Y \subseteq \mathcal{Y}$
  while chain not mixed **do**
    Pick $y \in \mathcal{Y}$, $p \in (0,1)$ uniformly randomly
    if $y \in Y$ **then**
      $Y' = Y \backslash \{y\}$
      Compute bounds $\lambda_{\min}$, $\lambda_{\max}$ on the spectrum of $L_{Y'}$
      if DPPJUDGE($L_{yy}-p$, $L_{Y',y}$, $L_{Y'}$, $\lambda_{\min}$, $\lambda_{\max}$) **then**
        $Y = Y'$
      end if
    else
      $Y' = Y \cup \{y\}$
      Compute bounds $\lambda_{\min}$, $\lambda_{\max}$ on the spectrum of L_Y
      if not DPPJUDGE($L_{yy}-p$, $L_{Y,y}$, L_Y, $\lambda_{\min}$, $\lambda_{\max}$) **then**
        $Y = Y'$
      end if
    end if
  end while

Algorithm 4 DPPJUDGE($t, u, A, \lambda_{\min}, \lambda_{\max}$)

Require: target value t; vector u, matrix A; lower and upper bounds $\lambda_{\min}$ and $\lambda_{\max}$ on the spectrum of A
Ensure: Return **true** if $t < u^\top A^{-1} u$, **false** otherwise
  while true do
    Run one Gauss-Radau iteration to get g^{rr} and g^{lr} for $u^\top A^{-1} u$.
    if $t < g^{\text{rr}}$ **then**
      return true
    else if $t \geq g^{\text{lr}}$ **then**
      return false
    end if
    $i = i + 1$
  end while

tightness of the lower and upper bounds by their difference (gap) $t_i - s_i$, and decide accordingly which quadrature to iterate further.

But the bounds for y and v are not symmetric and contribute differently to the transition decision. In essence, we need to judge the relation between p and $\frac{L_{y,y} - L_{y,Y'} L_{Y'}^{-1} L_{Y',y}}{L_{v,v} - L_{v,Y'} L_{Y'}^{-1} L_{Y',v}}$, or, equivalently, the relation between $pL_{v,v} - L_{y,y}$ and $pL_{v,Y'} L_{Y'}^{-1} L_{Y',v} - L_{y,Y'} L_{Y'}^{-1} L_{Y',y}$. Since the left hand side is "easy", the essential part is the right hand side. Assuming that in practice the impact is larger when the gap is larger, we tighten the bounds for $L_{v,Y} L_Y^{-1} L_{Y,v}$ if $p(t_j^v - s_j^v) > (t_i^y - s_i^y)$, and otherwise tighen the bounds for $L_{y,Y} L_Y^{-1} L_{Y,y}$. Details of the final algorithm with this refinement are shown in Appendix D.

5.2 Retrospective Double Greedy Algorithm

As indicated in Section 2, a number of applications, including sensing and information maximization with Gaussian Processes, rely on maximizing a submodular function given as $F(S) = \log\det(L_S)$. In general, this function may be non-monotone. In this case, an algorithm of choice is the double greedy algorithm of Buchbinder et al. [14].

The double greedy algorithm starts with two sets $X_0 = \emptyset$ and $Y_0 = \mathcal{Y}$ and serially iterates through all elements to construct a near-optimal subset. At iteration i, it includes element i into X_{i-1} with

probability q_i, and with probability $1 - q_i$ it excludes i from Y_{i-1}. The decisive value q_i is determined by the marginal gains $\Delta_i^- = F(Y_{i-1} \backslash \{i\}) - F(Y_{i-1})$ and $\Delta_i^+ = F(X_{i-1} \cup \{i\}) - F(X_{i-1})$:

$$q_i = [\Delta_i^+]_+ / [\Delta_i^+]_+ + [\Delta_i^-]_+.$$

For the log-det function, we obtain

$$\begin{aligned} \Delta_i^+ &= -\log(L_{i,i} - L_{i,Y'_{i-1}} L_{Y'_{i-1}}^{-1} L_{Y'_{i-1},i}) \\ \Delta_i^- &= \log(L_{i,i} - L_{i,X_{i-1}} L_{X_{i-1}}^{-1} L_{X_{i-1},i}), \end{aligned}$$

where $Y'_{i-1} = Y_{i-1} \backslash \{i\}$. In other words, at iteration i the algorithm uniformly samples $p \in (0, 1)$, and then checks if

$$p[\Delta_i^-]_+ \leq (1-p)[\Delta_i^+]_+,$$

and if true, adds i to X_{i-1}, otherwise removes it from Y_{i-1}.

This essential decision, whether to retain or discard an element, again involves bounding BIFs, for which we can take advantage of our framework, and profit from the typical sparsity of the data. Concretely, we retrospectively compute the lower and upper bounds on these BIFs, i.e., lower and upper bounds l_i^+ and u_i^+ on Δ_i^+, and l_i^- and u_i^- on Δ_i^-. If $p[u_i^-]_+ \leq (1-p)[l_i^+]_+$ we safely add i to X_{i-1}; if $p[l_i^-]_+ > (1-p)[u_i^+]_+$ we safely remove i from Y_{i-1}; otherwise we compute a set of tighter bounds by further iterating the quadrature.

As before, the bounds for Δ_i^- and Δ_i^+ may not contribute equally to the transition decision. We can again apply the refinement mentioned in Section 5.1: if $p([u_i^-]_+ - [l_i^-]_+) \leq (1-p)([u_i^+]_+ - [l_i^+]_+)$ we tighten bounds for Δ_i^+, otherwise we tighten bounds for Δ_i^-. The resulting algorithm is shown in Appendix E.

Data	Dimension	nnz	Density(%)
Abalone	4,177	144,553	0.83
Wine	4,898	2,659,910	11.09
GR	5,242	34,209	0.12
HEP	9,877	61,821	0.0634
Epinions	75,879	518,231	0.009
Slashdot	82,168	959,454	0.014

Table 1: Data. For all datasets we add an 1E-3 times identity matrix to ensure positive definiteness.

5.3 Empirical Evidence

We perform experiments on both synthetic and real-world datasets to test the impact of our retrospective quadrature framework in applications. We focus on (k-)DPP sampling and the double greedy algorithm for the log-det objective.

5.3.1 Synthetic Datasets

We generate small sparse matrices using methods similar to Section 4.4. For (k-)DPP we generate 5000×5000 matrices while for double greedy we use 2000×2000. We vary the density of the matrices from 10^{-3} to 10^{-1}. The running time and speedup are shown in Figure 2.

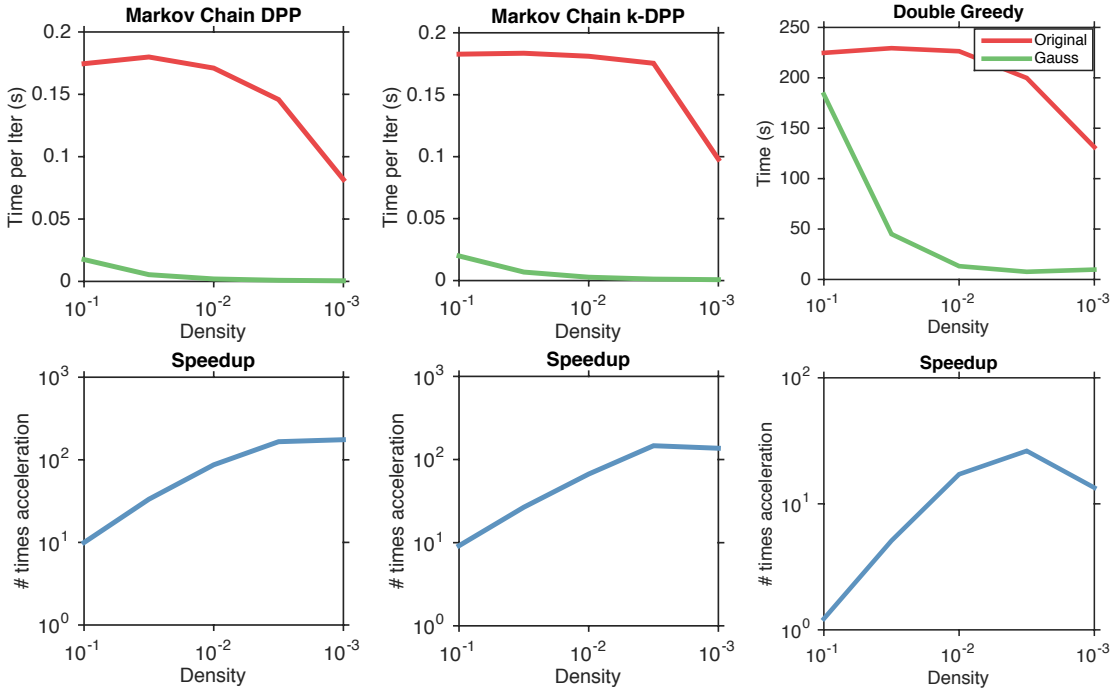

Figure 2: Running times (top) and corresponding speedup (bottom) on synthetic data. (k-)DPP is initialized with random subsets of size $N/3$ and corresponding running times are averaged over 1,000 iterations of the chain. All results are averaged over 3 runs.

	Abalone		Wine		GR		HEP		Epinions		Slashdot	
DPP	9.6E-3	1X	8.5E-2	1X	9.3E-3	1X	6.5E-2	1X	1.46	1X	5.85	1X
	5.4E-4	17.8X	5.9E-3	14.4X	4.3E-4	21.6X	5.9E-4	110.2X	3.7E-3	394.6X	7.1E-3	823.9X
k-DPP	1.4E-2	1X	0.15	1X	1.7E-2	1X	0.13	1X	2.40	1X	11.83	1X
	7.3E-4	19.2X	1.1E-2	13.6X	7.3E-4	23.3X	9.2E-4	141.3X	4.9E-3	489.8X	1E-2	1183X
DG	1025.6	1X	1951.3	1X	965.8	1X	6269.4	1X	*	*	*	*
	17.3	59.3X	423.2	4.6X	10	9.7X	25.3	247.8X	418	*	712.9	*

Table 2: Running time and speedup for (k-)DPP and double greedy. For results on each dataset (occupying two columns), the first column shows the running time (in seconds) and the second column shows the speedup. For each algorithm (occupying two rows), the first row shows results from the original algorithm and the second row shows results from algorithms using our framework. For Epinions and Slashdot, entries of "$*$" indicate that the experiments did not finish within 24 hours.

The results suggest that our framework greatly accelerates both DPP sampling and submodular maximization. The speedups are particularly pronounced for sparse matrices. As the matrices become very sparse, the original algorithms profit from sparsity too, and the difference shrinks a little. Overall, we see that our framework has the potential to lead to substantial speedups for algorithms involving bilinear inverse forms.

5.3.2 Real Datasets

We further test our framework on real-world datasets of varying sizes. We selected 6 datasets, four of them are of small/medium size and two are large. The four small/medium-sized datasets are used in [26]. The first two of small/medium-sized datasets, Abalone and Wine[2], are popular datasets for regression, and we construct sparse kernel matrices with an RBF kernel. We set the bandwidth parameter for Abalone as $\sigma = 0.15$ and that for Wine as $\sigma = 1$ and the cut-off parameter as 3σ for both datasets, as in [26]. The other two small/medium-sized datasets are GR (arXiv High Energy Physics collaboration graph) and HEP (arXiv General Relativity collaboration graph), where the kernel matrices are Laplacian matrices. The final two large datasets datasets are Epinions (Who-trusts-whom network of Epinions) and Slashdot (Slashdot social network from Feb. 2009) [3] with large Laplacian matrices. Dataset statistics are shown in Tab. 1.

The running times in Tab. 2 suggest that the iterative bounds from quadrature significantly accelerate (k-)DPP sampling and double greedy on real data. Our algorithms lead to speedups of up to a thousand times.

On the large sparse matrices, the "standard" double greedy algorithm did not finish within 24 hours, due to the expensive matrix operations involved. With our framework, the algorithm needs only 15 minutes.

To our knowledge, these results are the first time to run DPP and double greedy for information gain on such large datasets.

5.4 Numerical details

Instability. As seen in Alg. 1, the quadrature algorithm is built upon Lanczos iterations. Although in theory Lanczos iterations construct a set of orthogonal Lanczos vectors, in practice the constructed vectors usually lose orthogonality after some iterations due to rounding errors. One way to deal with this problem is to reorthogonalize the vectors, either completely at each iteration or selectively [**?**].

[2] Available at http://archive.ics.uci.edu/ml/.
[3] Available at https://snap.stanford.edu/data/.

Also, an equivalent Lanczos iteration proposed in [**?**] which uses a different expression to improve local orthogonality. Further discussion on numerical stability of the method lies beyond the scope of this paper.

Preconditioning. For Gauss quadrature on $u^\top A^{-1}u$, the convergence rate of bounds is dependent on the condition number of A. We can use preconditioning techniques to get a well-conditioned submatrix and proceed with that. Concretely, observe that for non-singular C,

$$\begin{aligned} u^\top A^{-1}u &= u^\top C^\top C^{-\top} A^{-1} C^{-1} C u \\ &= (Cu)(CAC^\top)^{-1}(Cu). \end{aligned}$$

Thus, if $CAC^\top$ is well-conditioned, we can use it with the vector Cu in Gauss quadrature.

There exists various ways to obtain good preconditioners for an SPD matrix. A simple choice is to use $C = [\text{diag}(A)]^{-1/2}$. There also exists methods for efficiently constructing sparse inverse matrix [**?**]. If L happens to be an SDD matrix, we can use techniques introduced in [**?**] to construct an approximate sparse inverse in near linear time.

6 Conclusion

In this paper we present a general and powerful computational framework for algorithms that rely on computations of bilinear inverse forms. The framework uses Gauss quadrature methods to lazily and iteratively tighten bounds, and is supported by our new theoretical results. We analyze properties of the various types of Gauss quadratures for approximating the bilinear inverse forms and show that all bounds are monotonically becoming tighter with the number of iterations; those given by Gauss-Radau are superior to those obtained from other Gauss-type quadratures; and both lower and upper bounds enjoy a linear convergence rate. We empirically verify the efficiency of our framework and are able to obtain speedups of up to a thousand times for two popular examples: maximizing information gain and sampling from determinantal point processes.

Acknowledgements This research was partially supported by NSF CAREER award 1553284 and a Google Research Award.

References

[1] Anari, Nima, Gharan, Shayan Oveis, and Rezaei, Alireza. Monte Carlo Markov chain algorithms for sampling strongly Rayleigh distributions and determinantal point processes. In *COLT*, 2016.

[2] Atzori, Luigi, Iera, Antonio, and Morabito, Giacomo. The Internet of Things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[3] Bai, Zhaojun and Golub, Gene H. Bounds for the trace of the inverse and the determinant of symmetric positive definite matrices. *Annals of Numerical Mathematics*, pp. 29–38, 1996.

[4] Bai, Zhaojun, Fahey, Gark, and Golub, Gene H. Some large-scale matrix computation problems. *Journal of Computational and Applied Mathematics*, pp. 71–89, 1996.

[5] Bekas, Constantine, Kokiopoulou, Effrosyni, and Saad, Yousef. An estimator for the diagonal of a matrix. *Applied numerical mathematics*, pp. 1214–1229, 2007.

[6] Bekas, Constantine, Curioni, Alessandro, and Fedulova, Irina. Low cost high performance uncertainty quantification. In *Proceedings of the 2nd Workshop on High Performance Computational Finance*, 2009.

[7] Belabbas, Mohamed-Ali and Wolfe, Patrick J. Spectral methods in machine learning and new strategies for very large datasets. *Proceedings of the National Academy of Sciences*, pp. 369–374, 2009.

[8] Benzi, Michele and Golub, Gene H. Bounds for the entries of matrix functions with applications to preconditioning. *BIT Numerical Mathematics*, pp. 417–438, 1999.

[9] Benzi, Michele and Klymko, Christine. Total communicability as a centrality measure. *J. Complex Networks*, pp. 124–149, 2013.

[10] Bonacich, Phillip. Power and centrality: A family of measures. *American Journal of Sociology*, pp. 1170–1182, 1987.

[11] Boutsidis, Christos, Mahoney, Michael W., and Drineas, Petros. An improved approximation algorithm for the column subset selection problem. In *SODA*, pp. 968–977, 2009.

[12] Brezinski, Claude. Error estimates for the solution of linear systems. *SIAM Journal on Scientific Computing*, pp. 764–781, 1999.

[13] Brezinski, Claude, Fika, Paraskevi, and Mitrouli, Marilena. Estimations of the trace of powers of positive self-adjoint operators by extrapolation of the moments. *Electronic Transactions on Numerical Analysis*, pp. 144–155, 2012.

[14] Buchbinder, Niv, Feldman, Moran, Naor, Joseph, and Schwartz, Roy. A tight linear time (1/2)-approximation for unconstrained submodular maximization. In *FOCS*, 2012.

[15] Dong, Shao-Jing and Liu, Keh-Fei. Stochastic estimation with z_2 noise. *Physics Letters B*, pp. 130–136, 1994.

[16] Estrada, Ernesto and Higham, Desmond J. Network properties revealed through matrix functions. *SIAM Review*, pp. 696–714, 2010.

[17] Fenu, Caterina, Martin, David R., Reichel, Lothar, and Rodriguez, Giuseppe. Network analysis via partial spectral factorization and Gauss quadrature. *SIAM Journal on Scientific Computing*, pp. A2046–A2068, 2013.

[18] Fika, Paraskevi and Koukouvinos, Christos. Stochastic estimates for the trace of functions of matrices via Hadamard matrices. *Communications in Statistics-Simulation and Computation*, 2015.

[19] Fika, Paraskevi and Mitrouli, Marilena. Estimation of the bilinear form $y^* f(A)x$ for Hermitian matrices. *Linear Algebra and its Applications*, 2015.

[20] Fika, Paraskevi, Mitrouli, Marilena, and Roupa, Paraskevi. Estimates for the bilinear form $x^T A^{-1} y$ with applications to linear algebra problems. *Electronic Transactions on Numerical Analysis*, pp. 70–89, 2014.

[21] Freericks, James K. Transport in multilayered nanostructures. *The Dynamical Mean-Field Theory Approach, Imperial College, London*, 2006.

[22] Frommer, Andreas, Lippert, Thomas, Medeke, Björn, and Schilling, Klaus. *Numerical Challenges in Lattice Quantum Chromodynamics: Joint Interdisciplinary Workshop of John Von Neumann Institute for Computing, Jülich, and Institute of Applied Computer Science, Wuppertal University, August 1999*, volume 15. Springer Science & Business Media, 2012.

[23] Gauss, Carl F. *Methodus nova integralium valores per approximationem inveniendi*. apvd Henricvm Dieterich, 1815.

[24] Gautschi, Walter. A survey of Gauss-Christoffel quadrature formulae. In *EB Christoffel*, pp. 72–147. Springer, 1981.

[25] Gillenwater, Jennifer, Kulesza, Alex, and Taskar, Ben. Near-optimal MAP inference for determinantal point processes. In *NIPS*, 2012.

[26] Gittens, Alex and Mahoney, Michael W. Revisiting the Nyström method for improved large-scale machine learning. *ICML*, 2013.

[27] Golub, Gene H. Some modified matrix eigenvalue problems. *SIAM Review*, pp. 318–334, 1973.

[28] Golub, Gene H. and Meurant, Gérard. Matrices, moments and quadrature II; how to compute the norm of the error in iterative methods. *BIT Numerical Mathematics*, pp. 687–705, 1997.

[29] Golub, Gene H. and Meurant, Gérard. *Matrices, moments and quadrature with applications*. Princeton University Press, 2009.

[30] Golub, Gene H. and Welsch, John H. Calculation of Gauss quadrature rules. *Mathematics of computation*, pp. 221–230, 1969.

[31] Golub, Gene H., Stoll, Martin, and Wathen, Andy. Approximation of the scattering amplitude and

linear systems. *Elec. Tran. on Numerical Analysis*, pp. 178–203, 2008.
[32] Hestenes, Magnus R. and Stiefel, Eduard. Methods of conjugate gradients for solving linear systems. *J. Research of the National Bureau of Standards*, pp. 409–436, 1952.
[33] Hough, J. Ben, Krishnapur, Manjunath, Peres, Yuval, and Virág, Bálint. Determinantal processes and independence. *Probability Surveys*, 2006.
[34] Kang, Byungkon. Fast determinantal point process sampling with application to clustering. In *NIPS*, pp. 2319–2327, 2013.
[35] Krause, Andreas, Singh, Ajit, and Guestrin, Carlos. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *JMLR*, pp. 235–284, 2008.
[36] Kulesza, Alex and Taskar, Ben. Determinantal point processes for machine learning. *arXiv:1207.6083*, 2012.
[37] Kwok, James T. and Adams, Ryan P. Priors for diversity in generative latent variable models. In *NIPS*, pp. 2996–3004, 2012.
[38] Lanczos, Cornelius. *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*. United States Governm. Press Office Los Angeles, CA, 1950.
[39] Lee, Christina E., Ozdaglar, Asuman E., and Shah, Devavrat. Solving systems of linear equations: Locally and asynchronously. *arXiv*, abs/1411.2647, 2014.
[40] Leskovec, Jure, Lang, Kevin J., Dasgupta, Anirban, and Mahoney, Michael W. Statistical properties of community structure in large social and information networks. In *WWW*, pp. 695–704, 2008.
[41] Lin, Lin, Yang, Chao, Lu, Jianfeng, and Ying, Lexing. A fast parallel algorithm for selected inversion of structured sparse matrices with application to 2D electronic structure calculations. *SIAM Journal on Scientific Computing*, pp. 1329–1351, 2011.
[42] Lin, Lin, Yang, Chao, Meza, Juan C., Lu, Jianfeng, Ying, Lexing, and E, Weinan. Selinv–an algorithm for selected inversion of a sparse symmetric matrix. *ACM Transactions on Mathematical Software*, 2011.
[43] Lobatto, Rehuel. *Lessen over de differentiaal-en integraal-rekening: Dl. 2 Integraal-rekening*, volume 1. Van Cleef, 1852.
[44] Meurant, Gérard. The computation of bounds for the norm of the error in the conjugate gradient algorithm. *Numerical Algorithms*, pp. 77–87, 1997.
[45] Meurant, Gérard. Numerical experiments in computing bounds for the norm of the error in the preconditioned conjugate gradient algorithm. *Numerical Algorithms*, pp. 353–365, 1999.
[46] Meurant, Gérard. *The Lanczos and conjugate gradient algorithms: from theory to finite precision computations*, volume 19. SIAM, 2006.
[47] Minoux, Michel. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization Techniques*, pp. 234–243. Springer, 1978.
[48] Mirzasoleiman, Baharan, Badanidiyuru, Ashwinkumar, Karbasi, Amin, Vondrák, Jan, and Krause, Andreas. Lazier than lazy greedy. In *AAAI*, 2015.
[49] Nemhauser, George L.., Wolsey, Laurence A., and Fisher, Marshall L. An analysis of approximations for maximizing submodular set functions–I. *Mathematical Programming*, pp. 265–294, 1978.
[50] Page, Lawrence, Brin, Sergey, Motwani, Rajeev, and Winograd, Terry. *The PageRank citation ranking: bringing order to the web.* Stanford InfoLab, 1999.
[51] Radau, Rodolphe. Étude sur les formules d'approximation qui servent à calculer la valeur numérique d'une intégrale définie. *J. de Mathématiques Pures et Appliquées*, pp. 283–336, 1880.
[52] Rasmussen, Carl E. and Williams, Christopher K. I. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006.
[53] Rocková, Veronika and George, Edward I. Determinantal priors for variable selection, 2015.
[54] Scott, John. *Social network analysis*. Sage, 2012.
[55] Sherman, Jack and Morrison, Winifred J. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, pp. 124–127, 1950.
[56] Shewchuk, Jonathan R. An introduction to the conjugate gradient method without the agonizing pain, 1994.

[57] Sidje, Roger B. and Saad, Yousef. Rational approximation to the fermi–dirac function with applications in density functional theory. *Numerical Algorithms*, pp. 455–479, 2011.
[58] Stoer, Josef and Bulirsch, Roland. *Introduction to numerical analysis*, volume 12. Springer Science & Business Media, 2013.
[59] Sviridenko, Maxim, Vondrák, Jan, and Ward, Justin. Optimal approximation for submodular and supermodular optimization with bounded curvature. In *SODA*, 2015.
[60] Tang, Jok M. and Saad, Yousef. A probing method for computing the diagonal of a matrix inverse. *Numerical Linear Algebra with Applications*, pp. 485–501, 2012.
[61] Wasow, Wolfgang R. A note on the inversion of matrices by random walks. *Mathematical Tables and Other Aids to Computation*, pp. 78–81, 1952.
[62] Wilf, Herbert S. *Mathematics for the Physical Sciences*. Wiley, New York, 1962.

A Further Background on Gauss Quadrature

We present below a more detailed summary of material on Gauss quadrature to make the paper self-contained.

A.1 Selecting weights and nodes

We've described that the Riemann-Stieltjes integral could be expressed as

$$I[f] := Q_n + R_n = \sum_{i=1}^{n} \omega_i f(\theta_i) + \sum_{i=1}^{m} \nu_i f(\tau_i) + R_n[f],$$

where Q_n denotes the nth degree approximation and R_n denotes a remainder term. The weights $\{\omega_i\}_{i=1}^n$, $\{\nu_i\}_{i=1}^m$ and nodes $\{\theta_i\}_{i=1}^n$ are chosen such that for all polynomials of degree less than $2n + m - 1$, denoted $f \in \mathbb{P}^{2n+m-1}$, we have *exact* interpolation $I[f] = Q_n$. One way to compute weights and nodes is to set $f(x) = x^i$ for $i \leq 2n + m - 1$ and then use this exact nonlinear system. But there is an easier way to obtain weights and nodes, namely by using polynomials orthogonal with respect to the measure α. Specifically, we construct a sequence of *orthogonal polynomials* $p_0(\lambda), p_1(\lambda), \ldots$ such that $p_i(\lambda)$ is a polynomial in λ of degree exactly k, and p_i, p_j are orthogonal, i.e., they satisfy

$$\int_{\lambda_{\min}}^{\lambda_{\max}} p_i(\lambda) p_j(\lambda) d\alpha(\lambda) = \begin{cases} 1, & i = j \\ 0, & \text{otherwise.} \end{cases}$$

The roots of p_n are distinct, real and lie in the interval of $[\lambda_{\min}, \lambda_{\max}]$, and form the nodes $\{\theta_i\}_{i=1}^n$ for Gauss quadrature (see, e.g., [29, Ch. 6]).

Consider the two *monic polynomials* whose roots serve as quadrature nodes:

$$\pi_n(\lambda) = \prod_{i=1}^{n} (\lambda - \theta_i), \quad \rho_m(\lambda) = \prod_{i=1}^{m} (\lambda - \tau_i),$$

where $\rho_0 = 1$ for consistency. We further denote $\rho_m^+ = \pm\rho_m$, where the sign is taken to ensure $\rho_m^+ \geq 0$ on $[\lambda_{\min}, \lambda_{\max}]$. Then, for $m > 0$, we calculate the quadrature weights as

$$\omega_i = I\left[\frac{\rho_m^+(\lambda)\pi_n(\lambda)}{\rho_m^+(\theta_i)\pi_n'(\theta_i)(\lambda - \theta_i)}\right], \quad \nu_j = I\left[\frac{\rho_m^+(\lambda)\pi_n(\lambda)}{(\rho_m^+)'(\tau_j)\pi_n(\tau_j)(\lambda - \tau_j)}\right],$$

where $f'(\lambda)$ denotes the derivative of f with respect to λ. When $m = 0$ the quadrature degenerates to Gauss quadrature and we have

$$\omega_i = I\left[\frac{\pi_n(\lambda)}{\pi_n'(\theta_i)(\lambda - \theta_i)}\right].$$

Although we have specified how to select nodes and weights for quadrature, these ideas cannot be applied to our problem because the measure α is unknown. Indeed, calculating the measure explicitly would require knowing the entire spectrum of A, which is as good as explicitly computing $f(A)$, hence untenable for us. The next section shows how to circumvent the difficulties due to unknown α.

A.2 Gauss Quadrature Lanczos (GQL)

The key idea to circumvent our lack of knowledge of α is to recursively construct polynomials called *Lanczos polynomials*. The construction ensures their orthogonality with respect to α. Concretely, we construct Lanczos polynomials via the following three-term recurrence:

$$\begin{aligned} &\beta_i p_i(\lambda) = (\lambda - \alpha_i) p_{i-1}(\lambda) - \beta_{i-1} p_{i-2}(\lambda), \quad i = 1, 2, \ldots, n \\ &p_{-1}(\lambda) \equiv 0; \quad p_0(\lambda) \equiv 1, \end{aligned} \tag{A.1}$$

while ensuring $\int_{\lambda_{\min}}^{\lambda_{\max}} d\alpha(\lambda) = 1$. We can express (A.1) in matrix form by writing

$$\lambda P_n(\lambda) = J_n P_n(\lambda) + \beta_n p_n(\lambda) e_n,$$

where $P_n(\lambda) := [p_0(\lambda), \ldots, p_{n-1}(\lambda)]^\top$, e_n is nth canonical unit vector, and J_n is the tridiagonal matrix

$$J_n = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \beta_2 & \ddots & \ddots & \\ & & \ddots & \alpha_{n-1} & \beta_{n-1} \\ & & & \beta_{n-1} & \alpha_n \end{bmatrix}. \tag{A.2}$$

This matrix is known as the *Jacobi matrix*, and is closed related to Gauss quadrature. The following well-known theorem makes this relation precise.

Theorem 10 ([30, 62]). *The eigenvalues of J_n form the nodes $\{\theta_i\}_{i=1}^n$ of Gauss-type quadratures. The weights $\{\omega_i\}_{i=1}^n$ are given by the squares of the first elements of the normalized eigenvectors of J_n.*

Thus, if J_n has the eigendecomposition $J_n = P_n^\top \Gamma P_n$, then for Gauss quadrature Thm. 10 yields

$$Q_n = \sum_{i=1}^n \omega_i f(\theta_i) = e_1^\top P_n^\top f(\Gamma) P_n e_1 = e_1^\top f(J_n) e_1. \tag{A.3}$$

Specialization. We now specialize to our main focus, $f(A) = A^{-1}$, for which we prove more precise results. In this case, (A.3) becomes $Q_n = [J_n^{-1}]_{1,1}$. The task now is to compute Q_n, and given A, u to obtain the Jacobi matrix J_n.

Fortunately, we can efficiently calculate J_n iteratively using the *Lanczos Algorithm* [38]. Suppose we have an estimate J_i, in iteration $(i+1)$ of Lanczos, we compute the tridiagonal coefficients α_{i+1} and β_{i+1} and add them to this estimate to form J_{i+1}. As to Q_n, assuming we have already computed $[J_i^{-1}]_{1,1}$, letting $j_i = J_i^{-1} e_i$ and invoking the Sherman-Morrison identity [55] we obtain the recursion:

$$[J_{i+1}^{-1}]_{1,1} = [J_i^{-1}]_{1,1} + \frac{\beta_i^2 ([j_i]_1)^2}{\alpha_{i+1} - \beta_i^2 [j_i]_i}, \tag{A.4}$$

where $[j_i]_1$ and $[j_i]_i$ can be recursively computed using a Cholesky-like factorization of J_i [29, p.31].

For Gauss-Radau quadrature, we need to modify J_i so that it has a prescribed eigenvalue. More precisely, we extend J_i to J_i^{lr} for left Gauss-Radau (J_i^{rr} for right Gauss-Radau) with β_i on the off-diagonal and α_i^{lr} (α_i^{rr}) on the diagonal, so that J_i^{lr} (J_i^{rr}) has a prescribed eigenvalue of $\lambda_{\min}$ ($\lambda_{\max}$).

For Gauss-Lobatto quadrature, we extend J_i to J_i^{lo} with values β_i^{lo} and α_i^{lo} chosen to ensure that J_i^{lo} has the prescribed eigenvalues $\lambda_{\min}$ and $\lambda_{\max}$. For more detailed on the construction, see [27].

For all methods, the approximated values are calculated as $[(J_i')^{-1}]_{1,1}$, where $J_i' \in \{J_i^{\mathrm{lr}}, J_i^{\mathrm{rr}}, J_i^{\mathrm{lo}}\}$ is the modified Jacobi matrix. Here J_i' is constructed at the i-th iteration of the algorithm.

The algorithm for computing Gauss, Gauss-Radau, and Gauss-Lobatto quadrature rules with the help of Lanczos iteration is called *Gauss Quadrature Lanczos* (GQL) and is shown in [28]. We recall its pseudocode in Alg. 1 to make our presentation self-contained (and for our proofs in Section 4).

The error of approximating $I[f]$ by Gauss-type quadratures can be expressed as

$$R_n[f] = \frac{f^{(2n+m)}(\xi)}{(2n+m)!} I[\rho_m \pi_n^2],$$

for some $\xi \in [\lambda_{\min}, \lambda_{\max}]$ (see, e.g., [58]). Note that ρ_m does not change sign in $[\lambda_{\min}, \lambda_{\max}]$; but with different values of m and τ_j we obtain different (but fixed) signs for $R_n[f]$ using $f(\lambda) = 1/\lambda$ and $\lambda_{\min} > 0$. Concretely, for Gauss quadrature $m = 0$ and $R_n[f] \geq 0$; for left Gauss-Radau $m = 1$ and

Algorithm 5 Gauss Quadrature Lanczos (GQL)

Require: u and A the corresponding vector and matrix, $\lambda_{\min}$ and $\lambda_{\max}$ lower and upper bounds for the spectrum of A

Ensure: g_i, g_i^{rr}, g_i^{lr} and g_i^{lo} the Gauss, right Gauss-Radau, left Gauss-Radau and Gauss-Lobatto quadrature computed at i-th iteration

Initialize: $u_{-1} = 0$, $u_0 = u/\|u\|$, $\alpha_1 = u_0^\top A u_0$, $\beta_1 = \|(A - \alpha_1 I)u_0\|$, $g_1 = \|u\|/\alpha_1$, $c_1 = 1$, $\delta_1 = \alpha_1$, $\delta_1^{\mathrm{lr}} = \alpha_1 - \lambda_{\min}$, $\delta_1^{\mathrm{rr}} = \alpha_1 - \lambda_{\max}$, $u_1 = (A - \alpha_1 I)u_0/\beta_1$, $i = 2$

while $i \leq N$ **do**

$\alpha_i = u_{i-1}^\top A u_{i-1}$ {Lanczos Iteration}

$\tilde{u}_i = A u_{i-1} - \alpha_i u_{i-1} - \beta_{i-1} u_{i-2}$

$\beta_i = \|\tilde{u}_i\|$

$u_i = \tilde{u}_i / \beta_i$

$g_i = g_{i-1} + \frac{\|u\| \beta_{i-1}^2 c_{i-1}^2}{\delta_{i-1}(\alpha_i \delta_{i-1} - \beta_{i-1}^2)}$ {Update g_i with Sherman-Morrison formula}

$c_i = c_{i-1} \beta_{i-1} / \delta_{i-1}$

$\delta_i = \alpha_i - \frac{\beta_{i-1}^2}{\delta_{i-1}}$, $\delta_i^{\mathrm{lr}} = \alpha_i - \lambda_{\min} - \frac{\beta_{i-1}^2}{\delta_{i-1}^{\mathrm{lr}}}$, $\delta_i^{\mathrm{rr}} = \alpha_i - \lambda_{\max} - \frac{\beta_{i-1}^2}{\delta_{i-1}^{\mathrm{rr}}}$

$\alpha_i^{\mathrm{lr}} = \lambda_{\min} + \frac{\beta_i^2}{\delta_i^{\mathrm{lr}}}$, $\alpha_i^{\mathrm{rr}} = \lambda_{\max} + \frac{\beta_i^2}{\delta_i^{\mathrm{rr}}}$ {Solve for J_i^{lr} and J_i^{rr}}

$\alpha_i^{\mathrm{lo}} = \frac{\delta_i^{\mathrm{lr}} \delta_i^{\mathrm{rr}}}{\delta_i^{\mathrm{rr}} - \delta_i^{\mathrm{lr}}}(\frac{\lambda_{\max}}{\delta_i^{\mathrm{lr}}} - \frac{\lambda_{\min}}{\delta_i^{\mathrm{rr}}})$, $(\beta_i^{\mathrm{lo}})^2 = \frac{\delta_i^{\mathrm{lr}} \delta_i^{\mathrm{rr}}}{\delta_i^{\mathrm{rr}} - \delta_i^{\mathrm{lr}}}(\lambda_{\max} - \lambda_{\min})$ {Solve for J_i^{lo}}

$g_i^{\mathrm{lr}} = g_i + \frac{\beta_i^2 c_i^2 \|u\|}{\delta_i(\alpha_i^{\mathrm{lr}} \delta_i - \beta_i^2)}$, $g_i^{\mathrm{rr}} = g_i + \frac{\beta_i^2 c_i^2 \|u\|}{\delta_i(\alpha_i^{\mathrm{rr}} \delta_i - \beta_i^2)}$, $g_i^{\mathrm{lo}} = g_i + \frac{(\beta_i^{\mathrm{lo}})^2 c_i^2 \|u\|}{\delta_i(\alpha_i^{\mathrm{lo}} \delta_i - (\beta_i^{\mathrm{lo}})^2)}$ {Update g_i^{rr}, g_i^{lr} and g_i^{lo} with Sherman-Morrison formula}

$i = i + 1$

end while

$\tau_1 = \lambda_{\min}$, so we have $R_n[f] \leq 0$; for right Gauss-Radau we have $m = 1$ and $\tau_1 = \lambda_{\max}$, thus $R_n[f] \geq 0$; while for Gauss-Lobatto we have $m = 2$, $\tau_1 = \lambda_{\min}$ and $\tau_2 = \lambda_{\max}$, so that $R_n[f] \leq 0$. This behavior of the errors clearly shows the ordering relations between the target values and the approximations made by the different quadrature rules. Lemma 2 (see e.g., [44]) makes this claim precise.

Lemma 11. *Let g_i, g_i^{lr}, g_i^{rr}, and g_i^{lo} be the approximations at the i-th iteration of Gauss, left Gauss-Radau, right Gauss-Radau, and Gauss-Lobatto quadrature, respectively. Then, g_i and g_i^{rr} provide lower bounds on $u^\top A^{-1} u$, while g_i^{lr} and g_i^{lo} provide upper bounds.*

The final connection we recall as background is the method of conjugate gradients. This helps us analyze the speed at which quadrature converges to the true value (assuming exact arithmetic).

A.3 Relation with Conjugate Gradient

While Gauss-type quadratures relate to the Lanczos algorithm, Lanczos itself is closely related to conjugate gradient (CG) [32], a well-known method for solving $Ax = b$ for positive definite A.

We recap this connection below. Let x_k be the estimated solution at the k-th CG iteration. If x^* denotes the true solution to $Ax = b$, then the *error* ε_k and *residual* r_k are defined as

$$\varepsilon_k := x^* - x_k, \qquad r_k = A\varepsilon_k = b - Ax_k, \tag{A.5}$$

At the k-th iteration, x_k is chosen such that r_k is orthogonal to the k-th *Krylov space*, i.e., the linear space $\mathcal{K}_k$ spanned by $\{r_0, Ar_0, \ldots, A^{k-1} r_0\}$. It can be shown [46] that r_k is a scaled Lanczos vector from the k-th iteration of Lanczos started with r_0. Noting the relation between Lanczos and Gauss quadrature applied to appoximate $r_0^\top A^{-1} r_0$, one obtains the following theorem that relates CG with GQL.

Theorem 12 (CG and GQL; [45]). *Let ε_k be the error as in (A.5), and let $\|\varepsilon_k\|_A^2 := \varepsilon_k^T A \varepsilon_k$. Then, it holds that*

$$\|\varepsilon_k\|_A^2 = \|r_0\|^2([J_N^{-1}]_{1,1} - [J_k^{-1}]_{1,1}),$$

where J_k is the Jacobi matrix at the k-th Lanczos iteration starting with r_0.

Finally, the rate at which $\|\varepsilon_k\|_A^2$ shrinks has also been well-studied, as noted below.

Theorem 13 (CG rate, see e.g. [56]). *Let ε_k be the error made by CG at iteration k when started with x_0. Let κ be the condition number of A, i.e., $\kappa = \lambda_1/\lambda_N$. Then, the error norm at iteration k satisfies*

$$\|\varepsilon_k\|_A \leq 2\Big(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\Big)^k \|\varepsilon_0\|_A.$$

Due to these explicit relations between CG and Lanczos, as well as between Lanczos and Gauss quadrature, we readily obtain the following convergence rate for relative error of Gauss quadrature.

Theorem 14 (Gauss quadrature rate). *The i-th iterate of Gauss quadrature satisfies the relative error bound*

$$\frac{g_N - g_i}{g_N} \leq 2\Big(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\Big)^i.$$

Proof. This is obtained by exploiting relations among CG, Lanczos and Gauss quadrature. Set $x_0 = 0$ and $b = u$. Then, $\varepsilon_0 = x^*$ and $r_0 = u$. An application of Thm. 12 and Thm. 13 thus yields the bound

$$\begin{aligned}\|\varepsilon_i\|_A^2 &= \|u\|^2([J_N^{-1}]_{1,1} - [J_i^{-1}]_{1,1}) = g_N - g_i \\ &\leq \quad 2\Big(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\Big)^i \|\varepsilon_0\|_A = 2\Big(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\Big)^i u^\top A^{-1} u = 2\Big(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\Big)^i g_N\end{aligned}$$

where the last equality draws from Lemma 15. □

In other words, Thm. 14 shows that the iterates of Gauss quadrature converge linearly.

B Proofs for Main Theoretical Results

We begin by proving an exactness property of Gauss and Gauss-Radau quadrature.

Lemma 15 (Exactness). *With A being symmetric positive definite with simple eigenvalues, the iterates g_N, g_N^{lr}, and g_N^{rr} are exact. Namely, after N iterations they satisfy*

$$g_N = g_N^{lr} = g_N^{rr} = u^\top A^{-1} u.$$

Proof. Observe that the Jacobi tridiagonal matrix can be computed via Lanczos iteration, and Lanczos is essentially essentially an iterative tridiagonalization of A. At the i-th iteration we have $J_i = V_i^\top A V_i$, where $V_i \in \mathbb{R}^{N\times i}$ are the first i Lanczos vectors (i.e., a basis for the i-th Krylov space). Thus, $J_N = V_N^\top A V_N$ where V_N is an $N \times N$ orthonormal matrix, showing that J_N has the same eigenvalues as A. As a result $\pi_N(\lambda) = \prod_{i=1}^N (\lambda - \lambda_i)$, and it follows that the remainder

$$R_N[f] = \frac{f^{(2N)}(\xi)}{(2N)!} I[\pi_N^2] = 0,$$

for some scalar $\xi \in [\lambda_{\min}, \lambda_{\max}]$, which shows that g_N is exact for $u^\top A^{-1} u$. For left and right Gauss-Radau quadrature, we have $\beta_N = 0$, $\alpha_N^{\text{lr}} = \lambda_{\min}$, and $\alpha_N^{\text{rr}} = \lambda_{\max}$, while all other elements of the

$(N+1)$-th row or column of J'_N are zeros. Thus, the eigenvalues of J'_N are $\lambda_1, \ldots, \lambda_N, \tau_1$, and $\pi_N(\lambda)$ again equals $\prod_{i=1}^{N}(\lambda - \lambda_i)$. As a result, the remainder satisfies

$$R_N[f] = \frac{f^{(2N)}(\xi)}{(2N)!} I[(\lambda - \tau_1)\pi_N^2] = 0,$$

from which it follows that both g_N^{rr} and g_N^{lr} are exact. □

The convergence rate in Thm. 13 and the final exactness of iterations in Lemma 15 does not necessarily indicate that we are making progress at each iterations. However, by exploiting the relations to CG we can indeed conclude that we are making progress in each iteration in Gauss quadrature.

Theorem 16. *The approximation g_i generated by Gauss quadrature is monotonically nondecreasing, i.e.,*

$$g_i \leq g_{i+1}, \quad \text{for } i < N.$$

Proof. At each iteration r_i is taken to be orthogonal to the i-th Krylov space: $\mathcal{K}_i = \mathrm{span}\{u, Au, \ldots, A^{i-1}u\}$. Let Π_i be the projection onto the complement space of $\mathcal{K}_i$. The residual then satisfies

$$\begin{aligned} \|\varepsilon_{i+1}\|_A^2 &= \varepsilon_{i+1}^T A \varepsilon_{i+1} = r_{i+1}^\top A^{-1} r_{i+1} \\ &= (\Pi_{i+1} r_i)^\top A^{-1} \Pi_{i+1} r_i \\ &= r_i^\top (\Pi_{i+1}^\top A^{-1} \Pi_{i+1}) r_i \leq r_i A^{-1} r_i, \end{aligned}$$

where the last inequality follows from $\Pi_{i+1}^\top A^{-1} \Pi_{i+1} \preceq A^{-1}$. Thus $\|\varepsilon_i\|_A^2$ is monotonically nonincreasing, whereby $g_N - g_i \geq 0$ is monotonically decreasing and thus g_i is monotonically nondecreasing. □

Before we proceed to Gauss-Radau, let us recall a useful theorem and its corollary.

Theorem 17 (Lanczos Polynomial [29])**.** *Let u_i be the vector generated by Alg. 1 at the i-th iteration; let p_i be the Lanczos polynomial of degree i. Then we have*

$$u_i = p_i(A)u_0, \quad \text{where } p_i(\lambda) = (-1)^i \frac{\det(J_i - \lambda I)}{\prod_{j=1}^{i} \beta_j}.$$

From the expression of Lanczos polynomial we have the following corollary specifying the sign of the polynomial at specific points.

Corollary 18. *Assume $i < N$. If i is odd, then $p_i(\lambda_{\min}) < 0$; for even i, $p_i(\lambda_{\min}) > 0$, while $p_i(\lambda_{\max}) > 0$ for any $i < N$.*

Proof. Since $J_i = V_i^\top A V_i$ is similar to A, its spectrum is bounded by $\lambda_{\min}$ and $\lambda_{\max}$ from left and right. Thus, $J_i - \lambda_{\min}$ is positive semi-definite, and $J_i - \lambda_{\max}$ is negative semi-definite. Taking $(-1)^i$ into consideration we will get the desired conclusions. □

We are ready to state our main result that compares (right) Gauss-Radau with Gauss quadrature.

Theorem 19 (Thm. 4 in the main text)**.** *Let $i < N$. Then, g_i^{rr} gives better bounds than g_i but worse bounds than g_{i+1}; more precisely,*

$$g_i \leq g_i^{rr} \leq g_{i+1}, \quad i < N. \tag{B.1}$$

Proof. We prove inequality (B.1) using the recurrences satisfied by g_i and g_i^{rr} (see Alg. 1)

Upper bound: $g_i^{rr} \leq g_{i+1}$. The iterative quadrature algorithm uses the recursive updates

$$g_i^{\mathrm{rr}} = g_i + \frac{\beta_i^2 c_i^2}{\delta_i(\alpha_i^{\mathrm{rr}}\delta_i - \beta_i^2)},$$
$$g_{i+1} = g_i + \frac{\beta_i^2 c_i^2}{\delta_i(\alpha_{i+1}\delta_i - \beta_i^2)}.$$

It suffices to thus compare α_i^{rr} and α_{i+1}. The three-term recursion for Lanczos polynomials shows that

$$\beta_{i+1}p_{i+1}(\lambda_{\max}) = (\lambda_{\max} - \alpha_{i+1})p_i(\lambda_{\max}) - \beta_i p_{i-1}(\lambda_{\max}) > 0,$$
$$\beta_{i+1}p^*_{i+1}(\lambda_{\max}) = (\lambda_{\max} - \alpha_i^{\mathrm{rr}})p_i(\lambda_{\max}) - \beta_i p_{i-1}(\lambda_{\max}) = 0,$$

where p_{i+1} is the original Lanczos polynomial, and p^*_{i+1} is the modified polynomial that has $\lambda_{\max}$ as a root. Noting that $p_i(\lambda_{\max}) > 0$, we see that $\alpha_{i+1} \leq \alpha_i^{\mathrm{rr}}$. Moreover, from Thm. 16 we know that the g_i's are monotonically increasing, whereby $\delta_i(\alpha_{i+1}\delta_i - \beta_i^2) > 0$. It follows that

$$0 < \delta_i(\alpha_{i+1}\delta_i - \beta_i^2) \leq \delta_i(\alpha_i^{\mathrm{rr}}\delta_i - \beta_i^2),$$

and from this inequality it is clear that $g_i^{\mathrm{rr}} \leq g_{i+1}$.
Lower-bound: $g_i \leq g_i^{rr}$. Since $\beta_i^2 c_i^2 \geq 0$ and $\delta_i(\alpha_i^{\mathrm{rr}}\delta_i - \beta_i^2) \geq \delta_i(\alpha_{i+1}\delta_i - \beta_i^2) > 0$, we readily obtain

$$g_i \leq g_i + \frac{\beta_i^2 c_i^2}{\delta_i(\alpha_i^{\mathrm{rr}}\delta_i - \beta_i^2)} = g_i^{\mathrm{rr}}. \qquad \square$$

Combining Thm. 19 with the convergence rate of relative error for Gauss quadrature (Thm. 14) immediately yields the following convergence rate for right Gauss-Radau quadrature:

Theorem 20 (Relative error of right Gauss-Radau, Thm. 5 in the main text)**.** *For each i, the right Gauss-Radau g_i^{rr} iterates satisfy*

$$\frac{g_N - g_i^{rr}}{g_N} \leq 2\Big(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\Big)^i.$$

This results shows that with the same number of iterations, right Gauss-Radau gives superior approximation over Gauss quadrature, though they share the same relative error convergence rate.

Our second main result compares Gauss-Lobatto with (left) Gauss-Radau quadrature.

Theorem 21 (Thm. 6 in the main text)**.** *Let $i < N$. Then, g_i^{lr} gives better upper bounds than g_i^{lo} but worse than g_{i+1}^{lo}; more precisely,*

$$g_{i+1}^{lo} \leq g_i^{lr} \leq g_i^{lo}, \quad i < N.$$

Proof. We prove these inequalities using the recurrences for g_i^{lr} and g_i^{lo} from Alg. 5.
$g_i^{lr} \leq g_i^{lo}$: From Alg. 5 we observe that $\alpha_i^{\mathrm{lo}} = \lambda_{\min} + \frac{(\beta_i^{\mathrm{lo}})^2}{\delta_i^{\mathrm{lr}}}$. Thus we can write g_i^{lr} and g_i^{lo} as

$$g_i^{\mathrm{lr}} = g_i + \frac{\beta_i^2 c_i^2}{\delta_i(\alpha_i^{\mathrm{lr}}\delta_i - \beta_i^2)} = g_i + \frac{\beta_i^2 c_i^2}{\lambda_{\min}\delta_i^2 + \beta_i^2(\delta_i^2/\delta_i^{\mathrm{lr}} - \delta_i)}$$
$$g_i^{\mathrm{lo}} = g_i + \frac{(\beta_i^{\mathrm{lo}})^2 c_i^2}{\delta_i(\alpha_i^{\mathrm{lo}}\delta_i - (\beta_i^{\mathrm{lo}})^2)} = g_i + \frac{(\beta_i^{\mathrm{lo}})^2 c_i^2}{\lambda_{\min}\delta_i^2 + (\beta_i^{\mathrm{lo}})^2(\delta_i^2/\delta_i^{\mathrm{lr}} - \delta_i)}$$

To compare these quantities, as before it is helpful to begin with the original three-term recursion for the Lanczos polynomial, namely

$$\beta_{i+1}p_{i+1}(\lambda) = (\lambda - \alpha_{i+1})p_i(\lambda) - \beta_i p_{i-1}(\lambda).$$

In the construction of Gauss-Lobatto, to make a new polynomial of order $i+1$ that has roots $\lambda_{\min}$ and $\lambda_{\max}$, we add $\sigma_1 p_i(\lambda)$ and $\sigma_2 p_{i-1}(\lambda)$ to the original polynomial to ensure

$$\begin{cases} \beta_{i+1}p_{i+1}(\lambda_{\min}) + \sigma_1 p_i(\lambda_{\min}) + \sigma_2 p_{i-1}(\lambda_{\min}) &= 0, \\ \beta_{i+1}p_{i+1}(\lambda_{\max}) + \sigma_1 p_i(\lambda_{\max}) + \sigma_2 p_{i-1}(\lambda_{\max}) &= 0. \end{cases}$$

Since β_{i+1}, $p_{i+1}(\lambda_{\max})$, $p_i(\lambda_{\max})$ and $p_{i-1}(\lambda_{\max})$ are all greater than 0, $\sigma_1 p_i(\lambda_{\max}) + \sigma_2 p_{i-1}(\lambda_{\max}) < 0$. To determine the sign of polynomials at $\lambda_{\min}$, consider the two cases:

1. Odd i. In this case $p_{i+1}(\lambda_{\min}) > 0$, $p_i(\lambda_{\min}) < 0$, and $p_{i-1}(\lambda_{\min}) > 0$;
2. Even i. In this case $p_{i+1}(\lambda_{\min}) < 0$, $p_i(\lambda_{\min}) > 0$, and $p_{i-1}(\lambda_{\min}) < 0$.

Thus, if $S = (\mathrm{sgn}(\sigma_1), \mathrm{sgn}(\sigma_2))$, where the signs take values in $\{0, \pm 1\}$, then $S \neq (1,1)$, $S \neq (-1,1)$ and $S \neq (0,1)$. Hence, $\sigma_2 \leq 0$ must hold, and thus $(\beta_i^{\mathrm{lo}})^2 = (\beta_i - \sigma_2)^2 \geq \beta_i^2$ given that $\beta_i^2 > 0$ for $i < N$.

Using $(\beta_i^{\mathrm{lo}})^2 \geq \beta_i^2$ with $\lambda_{\min} c_i^2 (\delta_i)^2 \geq 0$, an application of monotonicity of the univariate function $g(x) = \frac{ax}{b+cx}$ for $ab \geq 0$ to the recurrences defining g_i^{lr} and g_i^{lo} yields the desired inequality $g_i^{\mathrm{lr}} \leq g_i^{\mathrm{lo}}$.
$g_{i+1}^{lo} \leq g_i^{lr}$: From recursion formulas we have

$$g_i^{\mathrm{lr}} = g_i + \frac{\beta_i^2 c_i^2}{\delta_i(\alpha_i^{\mathrm{lr}}\delta_i - \beta_i^2)},$$
$$g_{i+1}^{\mathrm{lo}} = g_{i+1} + \frac{(\beta_{i+1}^{\mathrm{lo}})^2 c_{i+1}^2}{\delta_{i+1}(\alpha_{i+1}^{\mathrm{lo}}\delta_{i+1} - (\beta_{i+1}^{\mathrm{lo}})^2)}.$$

Establishing $g_i^{\mathrm{lr}} \geq g_{i+1}^{\mathrm{lo}}$ thus amounts to showing that (noting the relations among g_i, g_i^{lr} and g_i^{lo}):

$$\begin{aligned}
& \frac{\beta_i^2 c_i^2}{\delta_i(\alpha_i^{\mathrm{lr}}\delta_i - \beta_i^2)} - \frac{\beta_i^2 c_i^2}{\delta_i(\alpha_{i+1}\delta_i - \beta_i^2)} \quad \geq \quad \frac{(\beta_{i+1}^{\mathrm{lo}})^2 c_{i+1}^2}{\delta_{i+1}(\alpha_{i+1}^{\mathrm{lo}}\delta_{i+1} - (\beta_{i+1}^{\mathrm{lo}})^2)} \\
\iff \quad & \frac{\beta_i^2 c_i^2}{\delta_i(\alpha_i^{\mathrm{lr}}\delta_i - \beta_i^2)} - \frac{\beta_i^2 c_i^2}{\delta_i(\alpha_{i+1}\delta_i - \beta_i^2)} \quad \geq \quad \frac{(\beta_{i+1}^{\mathrm{lo}})^2 c_i^2 \beta_i^2}{(\delta_i)^2\delta_{i+1}(\alpha_{i+1}^{\mathrm{lo}}\delta_{i+1} - (\beta_{i+1}^{\mathrm{lo}})^2)} \\
\iff \quad & \frac{1}{\alpha_i^{\mathrm{lr}}\delta_i - \beta_i^2} - \frac{1}{\alpha_{i+1}\delta_i - \beta_i^2} \quad \geq \quad \frac{(\beta_{i+1}^{\mathrm{lo}})^2}{\delta_i\delta_{i+1}(\alpha_{i+1}^{\mathrm{lo}}\delta_{i+1} - (\beta_{i+1}^{\mathrm{lo}})^2)} \\
\iff \quad & \frac{1}{(\alpha_{i+1} - \delta_{i+1}^{\mathrm{lr}}) - \beta_i^2/\delta_i} - \frac{1}{\alpha_{i+1} - \beta_i^2/\delta_i} \quad \geq \quad \frac{1}{\delta_{i+1}(\alpha_{i+1}^{\mathrm{lo}}\delta_{i+1}/(\beta_{i+1}^{\mathrm{lo}})^2 - 1)} \qquad \text{(Lemma 23)} \\
\iff \quad & \frac{1}{\delta_{i+1} - \delta_{i+1}^{\mathrm{lr}}} - \frac{1}{\delta_{i+1}} \quad \geq \quad \frac{1}{\delta_{i+1}(\frac{\lambda_{\min}\delta_{i+1}}{(\beta_{i+1}^{\mathrm{lo}})^2} + \frac{\delta_{i+1}}{\delta_{i+1}^{\mathrm{lr}}} - 1)} \\
\iff \quad & \frac{\lambda_{\min}\delta_{i+1}}{(\beta_{i+1}^{\mathrm{lo}})^2} + \frac{\delta_{i+1}}{\delta_{i+1}^{\mathrm{lr}}} - 1 \geq \frac{\delta_{i+1}}{\delta_{i+1}^{\mathrm{lr}}} - 1 \\
\iff \quad & \frac{\lambda_{\min}\delta_{i+1}}{(\beta_{i+1}^{\mathrm{lo}})^2} \geq 0,
\end{aligned}$$

where the last inequality is obviously true; hence the proof is complete. □

In summary, we have the following corollary for all the four quadrature rules:

Corollary 22 (Monotonicity of Lower and Upper Bounds, Corr. 7 in the main text)**.** *As the iteration proceeds, g_i and g_i^{rr} gives increasingly better asymptotic lower bounds and g_i^{lr} and g_i^{lo} gives increasingly better upper bounds, namely*

$$g_i \leq g_{i+1}; \quad g_i^{rr} \leq g_{i+1}^{rr}$$
$$g_i^{lr} \geq g_{i+1}^{lr}; \quad g_i^{lo} \geq g_{i+1}^{lo}.$$

Proof. Directly drawn from Thm. 16, Thm. 19 and Thm. 21. □

Before proceeding further to our analysis of convergence rates of left Gauss-Radau and Gauss-Lobatto, we note two technical results that we will need.

Lemma 23. *Let α_{i+1} and α_i^{lr} be as in Alg. 1. The difference $\Delta_{i+1} = \alpha_{i+1} - \alpha_i^{lr}$ satisfies $\Delta_{i+1} = \delta_{i+1}^{lr}$.*

Proof. From the Lanczos polynomials in the definition of left Gauss-Radau quadrature we have

$$\begin{aligned}\beta_{i+1}p_{i+1}^*(\lambda_{\min}) &= (\lambda_{\min} - \alpha_i^{\mathrm{lr}})p_i(\lambda_{\min}) - \beta_i p_{i-1}(\lambda_{\min}) \\ &= (\lambda_{\min} - (\alpha_{i+1} - \Delta_{i+1}))p_i(\lambda_{\min}) - \beta_i p_{i-1}(\lambda_{\min}) \\ &= \beta_{i+1}p_{i+1}(\lambda_{\min}) + \Delta_{i+1}p_i(\lambda_{\min}) = 0.\end{aligned}$$

Rearrange this equation to write $\Delta_{i+1} = -\beta_{i+1}\frac{p_{i+1}(\lambda_{\min})}{p_i(\lambda_{\min})}$, which can be further rewritten as

$$\Delta_{i+1} \overset{\text{Thm. 17}}{=} -\beta_{i+1}\frac{(-1)^{i+1}\det(J_{i+1} - \lambda_{\min}I)/\prod_{j=1}^{i+1}\beta_j}{(-1)^i\det(J_i - \lambda_{\min}I)/\prod_{j=1}^{i}\beta_j} = \frac{\det(J_{i+1} - \lambda_{\min}I)}{\det(J_i - \lambda_{\min}I)} = \delta_{i+1}^{\mathrm{lr}}.$$ □

Remark 24. Lemma 23 has an implication beyond its utility for the subsequent proofs: it provides a new way of calculating α_{i+1} given the quantities $\delta_{i+1}^{\mathrm{lr}}$ and α_i^{lr}; this saves calculation in Alg. 5.

The following lemma relates δ_i to δ_i^{lr}, which will prove useful in subsequent analysis.

Lemma 25. *Let δ_i^{lr} and δ_i be computed in the i-th iteration of Alg. 1. Then, we have the following:*

$$\delta_i^{lr} < \delta_i, \tag{B.2}$$

$$\frac{\delta_i^{lr}}{\delta_i} \leq 1 - \frac{\lambda_{\min}}{\lambda_N}. \tag{B.3}$$

Proof. We prove (B.2) by induction. Since $\lambda_{\min} > 0$, $\delta_1 = \alpha_1 > \lambda_{\min}$ and $\delta_1^{\mathrm{lr}} = \alpha - \lambda_{\min}$ we know that $\delta_1^{\mathrm{lr}} < \delta_1$. Assume that $\delta_i^{\mathrm{lr}} < \delta_i$ is true for all $i \leq k$ and considering the $(k+1)$-th iteration:

$$\delta_{k+1}^{\mathrm{lr}} = \alpha_{k+1} - \lambda_{\min} - \frac{\beta_k^2}{\delta_k^{\mathrm{lr}}} < \alpha_{k+1} - \frac{\beta_k^2}{\delta_k} = \delta_{k+1}.$$

To prove (B.3), simply observe the following

$$\frac{\delta_i^{\mathrm{lr}}}{\delta_i} = \frac{\alpha_i - \lambda_{\min} - \beta_{i-1}^2/\delta_{i-1}^{\mathrm{lr}}}{\alpha_i - \beta_{i-1}^2/\delta^{i-1}} \overset{\text{(B.2)}}{\leq} \frac{\alpha_i - \lambda_{\min}}{\alpha_i} \leq 1 - \frac{\lambda_{\min}}{\lambda_N}.$$ □

With aforementioned lemmas we will be able to show how fast the difference between g_i^{lr} and g_i decays. Note that g_i^{lr} gives an upper bound on the objective while g_i gives a lower bound.

Lemma 26. *The difference between g_i^{lr} and g_i decreases linearly. More specifically we have*

$$g_i^{lr} - g_i \leq 2\kappa^+\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^i g_N$$

where $\kappa^+ = \lambda_N/\lambda_{\min}$ and κ is the condition number of A, i.e., $\kappa = \lambda_N/\lambda_1$.

Proof. We rewrite the difference $g_i^{\mathrm{lr}} - g_i$ as follows

$$\begin{aligned} g_i^{\mathrm{lr}} - g_i &= \frac{\beta_i^2 c_i^2}{\delta_i(\alpha_i^{\mathrm{lr}}\delta_i - \beta_i^2)} \\ &= \frac{\beta_i^2 c_i^2}{\delta_i(\alpha_{i+1}\delta_i - \beta_i^2)} \frac{\delta_i(\alpha_{i+1}\delta_i - \beta_i^2)}{\delta_i(\alpha_i^{\mathrm{lr}}\delta_i - \beta_i^2)} \\ &= \frac{\beta_i^2 c_i^2}{\delta_i(\alpha_{i+1}\delta_i - \beta_i^2)} \frac{1}{(\alpha_i^{\mathrm{lr}} - \beta_i^2/\delta_i)/(\alpha_{i+1} - \beta_i^2/\delta_i)} \\ &= \frac{\beta_i^2 c_i^2}{\delta_i(\alpha_i\delta_i - \beta_i^2)} \frac{1}{1 - \Delta_{i+1}/\delta_{i+1}}, \end{aligned}$$

where $\Delta_{i+1} = \alpha_{i+1} - \alpha_i^{\mathrm{lr}}$. Next, recall that $\frac{g_N - g_i}{g_N} \leq 2\Big(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\Big)^i$. Since g_i lower bounds g_N, we have

$$\Big(1 - 2\Big(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\Big)^i\Big) g_N \leq g_i \leq g_N,$$
$$\Big(1 - 2\Big(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\Big)^{i+1}\Big) g_N \leq g_{i+1} \leq g_N.$$

Thus, we can conclude that

$$\frac{\beta_i^2 c_i^2}{\delta_i(\alpha_i\delta_i - \beta_i^2)} = g_{i+1} - g_i \leq 2\Big(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\Big)^i g_N.$$

Now we focus on the term $\big(1 - \Delta_{i+1}/\delta_{i+1}\big)^{-1}$. Using Lemma 23 we know that $\Delta_{i+1} = \delta_{i+1}^{\mathrm{lr}}$. Hence,

$$\begin{aligned} 1 - \Delta_{i+1}/\delta_{i+1} &= 1 - \delta_{i+1}^{\mathrm{lr}}/\delta_{i+1} \\ &\geq 1 - (1 - \lambda_{\min}/\lambda_N) = \lambda_{\min}/\lambda_N \triangleq \frac{1}{\kappa^+}. \end{aligned}$$

Finally we have

$$g_i^{\mathrm{lr}} - g_i = \frac{\beta_i^2 c_i^2}{\delta_i(\alpha_i\delta_i - \beta_i^2)} \frac{1}{1 - \Delta_{i+1}/\delta_{i+1}} \leq 2\kappa^+ \Big(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\Big)^i g_N. \qquad \square$$

Theorem 27 (Relative error of left Gauss-Radau, Thm. 8 in the main text)**.** *For left Gauss-Radau quadrature where the preassigned node is $\lambda_{\min}$, we have the following bound on relative error:*

$$\frac{g_i^{lr} - g_N}{g_N} \leq 2\kappa^+ \Big(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\Big)^i,$$

where $\kappa^+ := \lambda_N/\lambda_{\min}$, $i < N$.

Proof. Write $g_i^{\mathrm{lr}} = g_i + (g_i^{\mathrm{lr}} - g_i)$. Since $g_i \leq g_N$, using Lemma 26 to bound the second term we obtain

$$g_i^{\mathrm{lr}} \leq g_N + 2\kappa^+ \Big(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\Big)^i g_N,$$

from which the claim follows upon rearrangement.

$\square$

Due to the relations between left Gauss-Radau and Gauss-Lobatto, we have the following corollary:

Corollary 28 (Relative error of Gauss-Lobatto, Corr. 9 in the main text). *For Gauss-Lobatto quadrature, we have the following bound on relative error:*

$$\frac{g_i^{lo} - g_N}{g_N} \leq 2\kappa^+ \Big(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\Big)^{i-1}, \tag{B.4}$$

where $\kappa^+ := \lambda_N/\lambda_{\min}$ *and* $i < N$.

C Generalization: Symmetric Matrices

In this section we consider the case where u lies in the column space of several top eigenvectors of A, and discuss how the aforementioned theorems vary. In particular, note that the previous analysis assumes that A is positive definite. With our analysis in this section we relax this assumption to the more general case where A is symmetric with simple eigenvalues, though we require u to lie in the space spanned by eigenvectors of A corresponding to positive eigenvalues.

We consider the case where A is symmetric and has the eigendecomposition of $A = Q\Lambda Q^\top = \sum_{i=1}^N \lambda_i q_i q_i^\top$ where λ_i's are eigenvalues of A increasing with i and q_i's are corresponding eigenvectors. Assume that u lies in the column space spanned by top k eigenvectors of A where all these k eigenvectors correspond to positive eigenvalues. Namely we have $u \in \text{Span}\{\{q_i\}_{i=N-k+1}^N\}$ and $0 < \lambda_{N-k+1}$.

Since we only assume that A is symmetric, it is possible that A is singular and thus we consider the value of $u^\top A^\dagger u$, where $A^\dagger$ is the pseudo-inverse of A. Due to the constraints on u we have

$$u^\top A^\dagger u = u^\top Q\Lambda^\dagger Q^\top u = u^\top Q_k \Lambda_k^\dagger Q_k^\top u = u^\top B^\dagger u,$$

where $B = \sum_{i=N-k+1}^N \lambda_i q_i q_i^\top$. Namely, if u lies in the column space spanned by the top k eigenvectors of A then it is equivalent to substitute A with B, which is the truncated version of A at top k eigenvalues and corresponding eigenvectors.

Another key observation is that, given that u lies only in the space spanned by $\{q_i\}_{i=N-k+1}^N$, the Krylov space starting at u becomes

$$\text{Span}\{u, Au, A^2u, \ldots\} = \text{Span}\{u, Bu, B^2u, \ldots, B^{k-1}u\} \tag{C.1}$$

This indicates that Lanczos iteration starting at matrix A and vector u will finish constructing the corresponding Krylov space after the k-th iteration. Thus under this condition, Alg. 1 will run at most k iterations and then stop. At that time, the eigenvalues of J_k are exactly the eigenvalues of B, thus they are exactly $\{\lambda_i\}_{i=N-k+1}^N$ of A. Using similar proof as in Lemma 15, we can obtain the following generalized exactness result.

Corollary 29 (Generalized Exactness). g_k, g_k^{rr} *and* g_k^{lr} *are exact for* $u^\top A^\dagger u = u^\top B^\dagger u$, *namely*

$$g_k = g_k^{rr} = g_k^{lr} = u^\top A^\dagger u = u^\top B^\dagger u.$$

The monotonicity and the relations between bounds given by various Gauss-type quadratures will still be the same as in the original case in Section 4, but the original convergence rate cannot apply in this case because now we probably have $\lambda_{\min}(B) = 0$, making κ undefined. This crash of convergence rate results from the crash of the convergence of the corresponding conjugate gradient algorithm for solving $Ax = u$. However, by looking at the proof of, e.g., [56], and by noting that $\lambda_1(B) = \ldots = \lambda_{N-k}(B) = 0$, with a slight modification of the proof we actually obtain the bound

$$\|\varepsilon^i\|_A^2 \leq \min_{P_i} \max_{\lambda \in \{\lambda_i\}_{i=N-k+1}^N} [P_i(\lambda)]^2 \|\varepsilon^0\|_A^2,$$

where P_i is a polynomial of order i. By using properties of Chebyshev polynomials and following the original proof (e.g., [29] or [56]) we obtain the following lemma for conjugate gradient.

Lemma 30. *Let ε^k be as before (for conjugate gradient). Then,*

$$\|\varepsilon^k\|_A \leq 2\Big(\frac{\sqrt{\kappa'}-1}{\sqrt{\kappa'}+1}\Big)^k \|\varepsilon_0\|_A, \quad \text{where } \kappa' := \lambda_N/\lambda_{N-k+1}.$$

Following this new convergence rate and connections between conjugate gradient, Lanczos iterations and Gauss quadrature mentioned in Section 4, we have the following convergence bounds.

Corollary 31 (Convergence Rate for Special Case)**.** *Under the above assumptions on A and u, due to the connection Between Gauss quadrature, Lanczos algorithm and Conjugate Gradient, the relative convergence rates of g_i, g_i^{rr}, g_i^{lr} and g_i^{lo} are given by*

$$\begin{aligned}
\frac{g_k - g_i}{g_k} &\leq 2\Big(\frac{\sqrt{\kappa'}-1}{\sqrt{\kappa'}+1}\Big)^i \\
\frac{g_k - g_i^{rr}}{g_k} &\leq 2\Big(\frac{\sqrt{\kappa'}-1}{\sqrt{\kappa'}+1}\Big)^i \\
\frac{g_i^{lr} - g_k}{g_k} &\leq 2\kappa'_m\Big(\frac{\sqrt{\kappa'}-1}{\sqrt{\kappa'}+1}\Big)^i \\
\frac{g_i^{lo} - g_k}{g_k} &\leq 2\kappa'_m\Big(\frac{\sqrt{\kappa'}-1}{\sqrt{\kappa'}+1}\Big)^i,
\end{aligned}$$

where $\kappa'_m = \lambda_N/\lambda'_{\min}$ and $0 < \lambda'_{\min} < \lambda_{N-k+1}$ is a lowerbound for nonzero eigenvalues of B.

D Accelerating MCMC for k-Dpp

We present details of a *Retrospective Markov Chain Monte Carlo (MCMC)* in Alg. 6 and Alg. 7 that samples for efficiently drawing samples from a k-Dpp, by accelerating it using our results on Gauss-type quadratures.

Algorithm 6 Gauss-kDpp (L, k)

Require: L the kernel matrix we want to sample Dpp from, k the size of subset and $\mathcal{Y} = [N]$ the ground set
Ensure: Y sampled from exact kDpp (L) where $|Y| = k$
  Randomly Initialize $Y \subseteq \mathcal{Y}$ where $|Y| = k$
  while not mixed **do**
    Pick $v \in Y$ and $u \in \mathcal{Y} \backslash Y$ uniformly randomly
    Pick $p \in (0, 1)$ uniformly randomly
    $Y' = Y \backslash \{v\}$
    Get lower and upper bounds $\lambda_{\min}$, $\lambda_{\max}$ of the spectrum of $L_{Y'}$
    if k-Dpp-JudgeGauss$(pL_{v,v} - L_{u,u}, p, L_{Y',u}, L_{Y',v}, \lambda_{\min}, \lambda_{\max})$ = *True* **then**
      $Y = Y' \cup \{u\}$
    end if
  end while

E Accelerating Stochastic Double Greedy

We present details of *Retrospective Stochastic Double Greedy* in Alg. 8 and Alg. 9 that efficiently select a subset $Y \in \mathcal{Y}$ that approximately maximize $\log\det(L_Y)$.

Algorithm 7 kDPP-JudgeGauss$(t, p, u, v, A, \lambda_{\min}, \lambda_{\max})$

Require: t the target value, p the scaling factor, u, v and A the corresponding vectors and matrix, $\lambda_{\min}$ and $\lambda_{\max}$ lower and upper bounds for the spectrum of A
Ensure: Return *True* if $t < p(v^\top A^{-1} v) - u^\top A^{-1} u$, *False* if otherwise
  $u_{-1} = 0$, $u_0 = u/\|u\|$, $i^u = 1$, $\beta_0^u = 0$, $d^u = \infty$
  $v_{-1} = 0$, $v_0 = v/\|v\|$, $i^v = 1$, $\beta_0^v = 0$, $d^v = \infty$
  while *True* **do**
    if $d^u > p d^v$ **then**
      Run one more iteration of Gauss-Radau on $u^\top A^{-1} u$ to get tighter $(g^{\text{lr}})^u$ and $(g^{\text{rr}})^u$
      $d^u = (g^{\text{lr}})^u - (g^{\text{rr}})^u$
    else
      Run one more iteration of Gauss-Radau on $v^\top A^{-1} v$ to get tighter $(g^{\text{lr}})^v$ and $(g^{\text{rr}})^v$
      $d^v = (g^{\text{lr}})^v - (g^{\text{rr}})^v$
    end if
    if $t < p\|v\|^2 (g^{\text{rr}})^v - \|u\|^2 (g^{\text{lr}})^u$ **then**
      Return *True*
    else if $t \geq p\|v\|^2 (g^{\text{lr}})^v - \|u\|^2 (g^{\text{rr}})^u$ **then**
      Return *False*
    end if
  end while

Algorithm 8 Gauss-DG (L)

Require: L the kernel matrix and $\mathcal{Y} = [N]$ the ground set
Ensure: $X \in \mathcal{Y}$ that approximately maximize $\log\det(L_Y)$
  $X_0 = \emptyset$, $Y_0 = \mathcal{Y}$
  for $i = 1, 2, \ldots, N$ **do**
    $Y_i' = Y_{i-1} \backslash \{i\}$
    Sample $p \in (0, 1)$ uniformly randomly
    Get lower and upper bounds $\lambda_{\min}^-, \lambda_{\max}^-, \lambda_{\min}^+, \lambda_{\max}^+$ of the spectrum of $L_{X_{i-1}}$ and $L_{Y_i'}$ respectively
    if DG-JudgeGauss$(L_{X_{i-1}}, L_{Y_i'}, L_{X_{i-1},i}, L_{Y_i',i}, L_{i,i}, p, \lambda_{\min}^-, \lambda_{\max}^-, \lambda_{\min}^+, \lambda_{\max}^+)$ = *True* **then**
      $X_i = X_{i-1} \cup \{i\}$
    else
      $Y_i = Y_i'$
    end if
  end for

Algorithm 9 DG-JudgeGauss($A, B, u, v, t, p, \lambda^A_{\min}, \lambda^A_{\max}, \lambda^B_{\min}, \lambda^B_{\max}$)

Require: t the target value, p the scaling factor, u, v, A and B the corresponding vectors and matrix, $\lambda^A_{\min}$, $\lambda^A_{\max}$, $\lambda^B_{\min}$, $\lambda^B_{\max}$ lower and upper bounds for the spectrum of A and B

Ensure: Return *True* if $p|\log(t - u^\top A^{-1}u)|_+ \leq (1-p)|-\log(t - v^\top B^{-1}v)|_+$, *False* if otherwise

$d^u = \infty$, $d^v = \infty$

while *True* **do**

  if $pd^u > (1-p)d^v$ **then**

    Run one more iteration of Gauss-Radau on $u^\top A^{-1}u$ to get tighter lower and upper bounds l^u, u^u for $|\log(t - u^\top A^{-1}u)|_+$

    $d^u = u^u - l^u$

  else

    Run one more iteration of Gauss-Radau on $v^\top B^{-1}v$ to get tighter lower and upper bounds l^v, u^v for $|\log(t - v^\top B^{-1}v)|_+$

    $d^v = u^v - l^v$

  end if

  if $pu^u \leq (1-p)l^v$ **then**

    Return *True*

  else if $pl^u > (1-p)u^v$ **then**

    Return *False*

  end if

end while