Gaussian quadrature for matrix inverse forms with applications

Chengtao Li CTLI@MIT.EDU
Suvrit Sra SUVRIT@MIT.EDU
Stefanie Jegelka STEFJE@MIT.EDU
Massachusetts Institute of Technology

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 (Gauss, 1815) for approximating integrals, Gauss- and *Gauss-type quadrature* (i.e., Gauss-Lobatto (Lobatto, 1852) and Gauss-Radau (Radau, 1880) quadrature) have since found application to bilinear forms including computation of $u^T A^{-1}v$ (Bai et al., 1996). 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., (Golub & Meurant, 2009)), 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

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

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 (1999) and Brezinski et al. (2012) 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. (2014) 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 (Fika & Mitrouli, 2015). In addition, other methods exist for estimating trace of a matrix function (Bai & Golub, 1996; Brezinski et al., 2012; Fika & Koukouvinos, 2015) or diagonal elements of matrix inverse (Bekas et al., 2007; Tang & Saad, 2012).

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 (Kulesza & Taskar, 2012).

Exact sampling from a (k-)DPP requires eigendecomposition of L (Hough et al., 2006), 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 (Belabbas & Wolfe, 2009; Anari et al., 2016), 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 (Rocková & George, 2015; Kwok & Adams, 2012). 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 (Gillenwater et al., 2012), matrix approximations by column selection (Boutsidis et al., 2009; Sviridenko et al., 2015) and sensing Krause et al. (2008). 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 (Nemhauser et al., 1978) or non-monotone (Buchbinder et al., 2012) submodular functions rely on marginal gains of the form

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

for $U = T \backslash \{i\}$ and $\bar{S} = [N] \backslash S$. 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 (Minoux, 1978) and stochastic greedy algorithms (Mirzasoleiman et al., 2015).

Network Analysis, Centrality. When analyzing relationships and information flows between connected entities in a network, such as people, organizations, computers, smart hardwares, etc. (Scott, 2012; Leskovec et al., 2008; Atzori et al., 2010; Fenu et al., 2013; Estrada & Higham, 2010; Benzi & Klymko, 2013), an important question is to mea-

sure 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* (Page et al., 1999) is the solution to $(I-(1-\alpha)A^\top)x = \alpha\mathbf{1}/N$, and *Bonacich centrality* (Bonacich, 1987) 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 (Wasow, 1952; Lee et al., 2014). 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 (Dong & Liu, 1994; Frommer et al., 2012), some signal processing tasks (Golub et al., 2008), and in Gaussian Process (GP) Regression (Rasmussen & Williams, 2006), e.g., for estimating variances. In numerical linear algebra, BIFs are used in rational approximations (Sidje & Saad, 2011), evaluation of Green's function (Freericks, 2006), and selective inversion of sparse matrices (Lin et al., 2011a;b; Lee et al., 2014). A notable use is the design of preconditioners (Benzi & Golub, 1999) and uncertainty quantification (Bekas et al., 2009).

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 (Golub & Meurant, 2009).

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

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

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

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

Our task now reduces to approximating the integral (3.1), for which we invoke the powerful idea of Gauss-type quadratures (Gauss, 1815; Radau, 1880; Lobatto, 1852; Gautschi, 1981). 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 (Gauss, 1815); $m = 1$ with $\tau_1 = \lambda_{\min}$ ($\tau_1 = \lambda_{\max}$) gives left (right) Gauss-Radau quadrature (Radau, 1880); $m = 2$ with $\tau_1 = \lambda_{\min}$ and $\tau_2 = \lambda_{\max}$ yields Gauss-Lobatto quadrature (Lobatto, 1852); while for general m we obtain Gauss-Christoffel quadrature (Gautschi, 1981).

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. *(Wilf, 1962; Golub & Welsch, 1969) 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.*

[1] The summary in this section is derived from various sources: (Gautschi, 1981; Bai et al., 1996; Golub & Meurant, 2009). Experts can skim this section for collecting our notation before moving onto Section 4, which contains our new results.

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: $(g_i, g_i^{\text{rr}}, g_i^{\text{lr}}, g_i^{\text{lo}})$: 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

If J_n has the eigendecomposition $P_n^\top \Gamma P_n$, then for Gauss quadrature Theorem 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 (Lanczos, 1950). 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)* (Golub & Meurant, 1997), is summarily presented as Algorithm 1. The complete algorithm may be found in Appendix A.

Theorem 2. *(Meurant, 1997) 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, Theorem 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*

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

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*

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

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

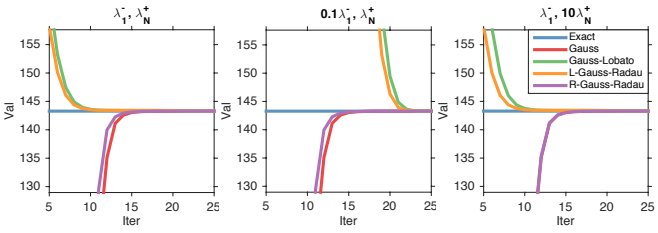
Remarks All aforementioned results assumed that A is strictly positive definite with simple eigenvalues. In Appendix C, we show similar results for the more general case that A is only required to be symmetric, and u lies in the space spanned by eigenvectors of A corresponding to distinct positive eigenvalues.

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 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.



(a) (b) (c)

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

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 Theorem 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) maximization of a

Algorithm 2 Efficient Retrospective Framework

Input: 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

(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 (Kang, 2013); 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 $p \le L_{y,y} - t_i$, then we can safely accept the transition, if $p \ge L_{y,y} - s_i$, 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 (Algorithm 3) is an exact Markov chain for the DPP. In each iteration, it calls Algorithm 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 $p \le \frac{L_{y,y} - t_i^y}{L_{v,v} - s_j^v}$, we can safely accept the transition; and if $p \ge \frac{L_{y,y} - s_i^y}{L_{v,v} - t_j^v}$ 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 tightness of the lower and upper bounds by their difference (gap) $t_i - s_i$, and decide accordingly which quadrature to iterate further.

Algorithm 3 Gauss-DPP (L)

Input: DPP kernel L; ground set $\mathcal{Y}$
Output: 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($t, u, A, \lambda_{\min}, \lambda_{\max}$)

Input: target value t; vector u, matrix A; lower and upper bounds $\lambda_{\min}$ and $\lambda_{\max}$ on the spectrum of A
Output: 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 $t \ge g^{\text{lr}}$ **then**
    return false
  end if
  $i = i + 1$
end while

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. (2012).

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

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

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.

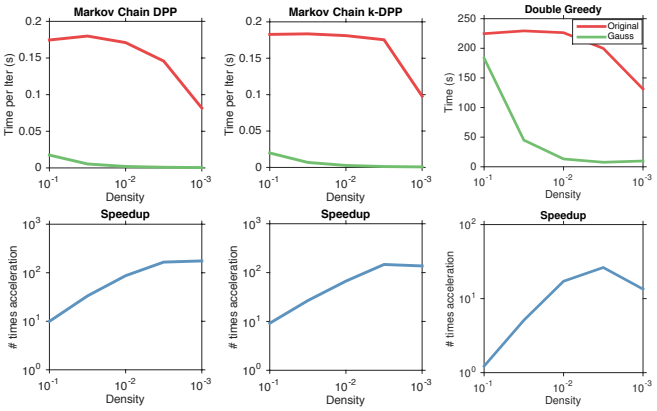


Figure 2: Running times (top) and corresponding speedup (bottom) on synthetic data. (k-)DPP is initialized with random subsets of size $N/3$ and corresponding running times are averaged over 1,000 iterations of the chain. All results are averaged over 3 runs.

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.

The results suggest that our framework greatly accelerates

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

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 (Gittens & Mahoney, 2013). 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 (Gittens & Mahoney, 2013). 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 Table 1.

The running times in Table 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.

[2]Available at http://archive.ics.uci.edu/ml/.

[3]Available at https://snap.stanford.edu/data/.

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 an NSF CAREER award 1553284 and a Google Research Award.

References

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.

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

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.

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

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

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.

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.

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

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

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

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

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

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.

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

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

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

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.

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

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

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.

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

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.

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

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

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

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

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

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.

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

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

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.

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.

Hough, J. Ben, Krishnapur, Manjunath, Peres, Yuval, and Virág, Bálint. Determinantal processes and independence. *Probability Surveys*, 2006.

Kang, Byungkon. Fast determinantal point process sampling with application to clustering. In *NIPS*, pp. 2319–2327, 2013.

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.

Kulesza, Alex and Taskar, Ben. Determinantal point processes for machine learning. *arXiv:1207.6083*, 2012.

Kwok, James T. and Adams, Ryan P. Priors for diversity in generative latent variable models. In *NIPS*, pp. 2996–3004, 2012.

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.

Lee, Christina E., Ozdaglar, Asuman E., and Shah, Devavrat. Solving systems of linear equations: Locally and asynchronously. *arXiv*, abs/1411.2647, 2014.

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.

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, 2011a.

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*, 2011b.

Lobatto, Rehuel. *Lessen over de differentiaal-en integraalrekening: Dl. 2 Integraal-rekening*, volume 1. Van Cleef, 1852.

Meurant, Gérard. The computation of bounds for the norm of the error in the conjugate gradient algorithm. *Numerical Algorithms*, pp. 77–87, 1997.

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.

Meurant, Gérard. *The Lanczos and conjugate gradient algorithms: from theory to finite precision computations*, volume 19. SIAM, 2006.

Minoux, Michel. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization Techniques*, pp. 234–243. Springer, 1978.

Mirzasoleiman, Baharan, Badanidiyuru, Ashwinkumar, Karbasi, Amin, Vondrák, Jan, and Krause, Andreas. Lazier than lazy greedy. In *AAAI*, 2015.

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.

Page, Lawrence, Brin, Sergey, Motwani, Rajeev, and Winograd, Terry. *The PageRank citation ranking: bringing order to the web.* Stanford InfoLab, 1999.

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.

Rasmussen, Carl E. and Williams, Christopher K. I. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006.

Rocková, Veronika and George, Edward I. Determinantal priors for variable selection, 2015.

Scott, John. *Social network analysis*. Sage, 2012.

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.

Shewchuk, Jonathan R. An introduction to the conjugate gradient method without the agonizing pain, 1994.

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.

Stoer, Josef and Bulirsch, Roland. *Introduction to numerical analysis*, volume 12. Springer Science & Business Media, 2013.

Sviridenko, Maxim, Vondrák, Jan, and Ward, Justin. Optimal approximation for submodular and supermodular optimization with bounded curvature. In *SODA*, 2015.

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.

Wasow, Wolfgang R. A note on the inversion of matrices by random walks. *Mathematical Tables and Other Aids to Computation*, pp. 78–81, 1952.

Wilf, Herbert S. *Mathematics for the Physical Sciences*. Wiley, New York, 1962.