PHILOSOPHICAL TRANSACTIONS A

rsta.royalsocietypublishing.org

Research

Article submitted to journal

Subject Areas:
Determinantal Point Processes

Keywords:
Determinantal Point Processes, High-Performance Computing, sparse-direct, factorization, parallel, Kasteleyn, non-Hermitian

Author for correspondence:
Jack Poulson <jack@hodgestar.com>

High-performance sampling of generic Determinantal Point Processes

Jack Poulson

Hodge Star Scientific Computing, hodgestar.com

Determinantal Point Processes (DPPs) were introduced by Macchi [1] as a model for repulsive (fermionic) particle distributions. But their recent popularization is largely due to their usefulness for encouraging diversity in the final stage of a recommender system [2].

The standard sampling scheme for finite DPPs is a spectral decomposition followed by an equivalent of a randomly diagonally-pivoted Cholesky factorization of an orthogonal projection, which is only applicable to Hermitian kernels and has an expensive setup cost. Researchers have begun to connect DPP sampling to LDL^H factorizations as a means of avoiding the initial spectral decomposition, but existing approaches have only outperformed the spectral decomposition approach in special circumstances, where the number of kept modes is a small percentage of the ground set size.

This article proves that trivial modifications of LU and LDL^H factorizations yield efficient direct sampling schemes for non-Hermitian and Hermitian DPP kernels, respectively. Further, it is experimentally shown that even dynamically-scheduled, shared-memory parallelizations of high-performance dense and sparse-direct factorizations can be trivially modified to yield DPP sampling schemes with essentially identical performance.

The software developed as part of this research, Catamari [hodgestar.com/catamari] is released under the Mozilla Public License v2.0. It contains header-only, C++14 plus OpenMP 4.0 implementations of dense and sparse-direct, Hermitian and non-Hermitian DPP samplers.

THE ROYAL SOCIETY PUBLISHING

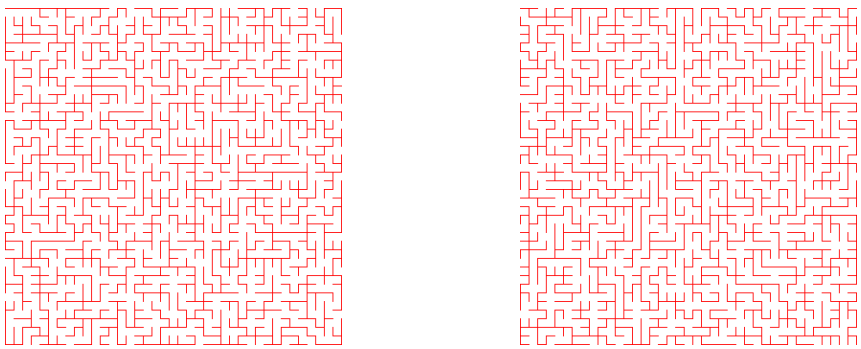


Figure 1: Two samples from the uniform distribution over spanning trees of a 40 x 40 box of $\mathbb{Z}^2$. Each has likelihood $\exp(-1794.24)$.

Figure 2: Two samples from the uniform distribution over spanning trees of a 10 x 10 section of hexagonal tiles. Each has likelihood $\exp(-299.101)$.

1. Introduction

Determinantal Point Processes (DPPs) were first studied as a distinct class by Macchi in the mid 1970's [1,3] as a probability distribution for the locations of repulsive (fermionic) particles, in direct contrast with Permanental – or, bosonic – Point Processes. Particular instances of DPPs, representing the eigenvalue distributions of classical random matrix ensembles, appeared in a series of papers in the beginning of the 1960's [4–8] (see [9] for a review, and [10] for a computational perspective on sampling classical eigenvalue distributions). Some of the early investigations of (finite) DPPs involved their usage for uniformly sampling spanning trees of graphs [11,12] (see Figs. 1 and 2), non-intersecting random walks [13], and domino tilings of the Aztec diamond [14–17] (see Figs. 3 and 4). Generalizations from determinantal and permanental point processes to immanants are also known [18].

Definition 1
A finite **Determinantal Point Process** *is a random variable* $\mathbf{Y} \sim DPP(K)$ *over the power set of a ground set* $\{0, 1, ..., n-1\} = [n]$ *such that*

$$\mathbb{P}[Y \subseteq \mathbf{Y}] = \det(K_Y),$$

where $K \in \mathbb{C}^{n \times n}$ *is called the* **marginal kernel matrix** *and* K_Y *denotes the restriction of* K *to the row and column indices of* Y.

We can immediately observe that the j-th diagonal entry of a marginal kernel K is the probability of index j being in the sample, $\mathbf{Y}$, so the diagonal of every marginal kernel must lie in $[0, 1] \subset \mathbb{R}$. A characteristic requirement for a complex matrix to be admissible as a marginal kernel is given in the following proposition, due to Brunel [19], which we will provide an alternative proof of after introducing our factorization-based sampling algorithm.

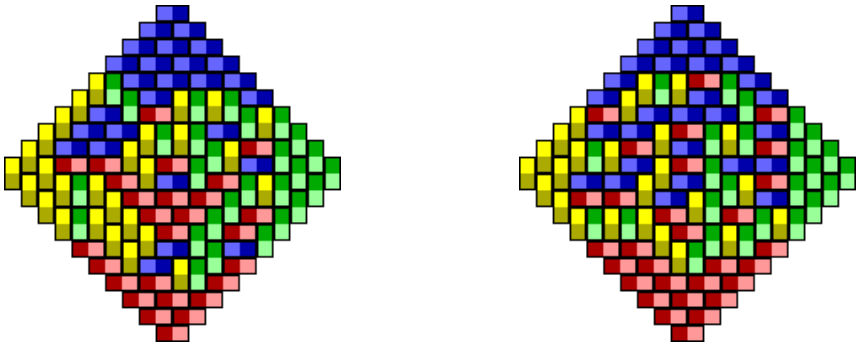


Figure 3: Two samples from the Aztec diamond DPP of order 10, defined via a complex, non-Hermitian kernel based upon Kenyon's formula over the Kasteleyn matrix. Their likelihoods are both $\exp(-38.1231)$.

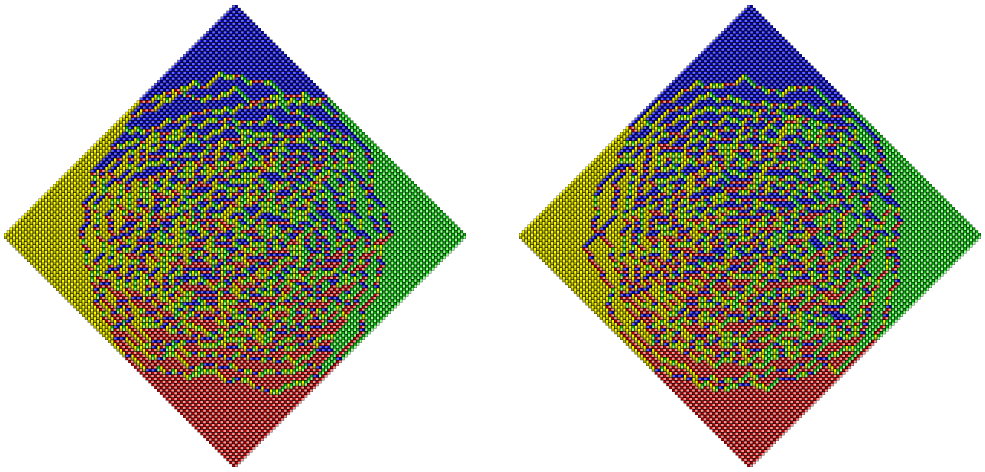


Figure 4: Two samples from the Aztec diamond DPP of order 80, defined via a complex, non-Hermitian kernel based upon Kenyon's formula over the Kasteleyn matrix. Their likelihoods are both $\exp(-2245.8)$.

Proposition 1 (Brunel [19])
A matrix $K \in \mathbb{C}^{n\times n}$ is admissible as a DPP marginal kernel iff

$$(-1)^{|J|} \det(K - I_J) \geq 0, \ \ \forall J \subseteq [n].$$

We can also observe that, because determinants are preserved under similarity transformations, there is a nontrivial equivalence class for marginal kernels:

Proposition 2
The equivalence class of a DPP kernel $K \in \mathbb{C}^{n\times n}$ contains its orbit under the group of diagonal similarity transformations, i.e.,

$$\{D^{-1}KD \,:\, D = diag(d), d \in (\mathbb{C} \setminus \{0\})^n\}.$$

If we restrict to the classes of complex Hermitian or real symmetric kernels, the same statement holds with the entries of d restricted to the circle group $U(1)$ and the scalar orthogonal group $O(1)$, respectively.

Proof. Determinants are preserved under similarity transformations, so the probability of inclusion of each subset is unchanged by global diagonal similarity. That a unitary diagonal similarity preserves Hermiticity follows from recognition that it becomes a Hermitian congruence. Likewise, a signature matrix similarity transformation becomes a real symmetric congruence. □

In most works, with the notable exception of Kasteleyn matrix approaches to studying domino tilings of the Aztec diamond [14–17], the marginal kernel is assumed Hermitian. And Macchi [1] showed (Cf. [20,21]) that a Hermitian matrix is admissibile as a marginal kernel if and only if its eigenvalues all lie in $[0, 1]$. The viewpoint of these eigenvalues as probabilities turns out to be productive, as the most common sampling algorithm for Hermitian DPPs, due to [21] and popularized in the machine learning community by [2], produces an equivalent random orthogonal projection matrix by preserving eigenvectors with probability equal to their eigenvalue:

Theorem 1 (Theorem 7 of [21])
Given any Hermitian marginal kernel matrix $K \in \mathbb{C}^{n\times n}$ with spectral decomposition $Q\Lambda Q^H$, sampling from $\mathbf{Y} \sim DPP(K)$ is equivalent to sampling from a realization of the random DPP with kernel $Q_{\mathbf{Z}}Q_{\mathbf{Z}}^H$, where $Q_{\mathbf{Z}}$ consists of the columns of Q with $\mathbb{P}[j \in \mathbf{Z}] = \Lambda_j$, independently for each j.

Such an orthogonal projection marginal kernel is said to define a **Determinantal Projection Process** [21], or **elementary DPP** [2], and it is known that the resulting samples almost surely have cardinality equal to the rank of the projection (e.g., Lemma 17 of [21]). The algebraic specification in Alg. 1 of [2] of the Alg. 18 of [21] involved an $O(nk^3)$ approach, where k is the rank of the projection kernel. In many important cases, such as uniformly sampling spanning trees or domino tilings, k is a large fraction of n, and the algorithm has quartic complexity (assuming standard matrix multiplication). In the words of [2]:

> Alg. 1 runs in time $O(nk^3)$, where k is the number of eigenvectors selected [...] the initial eigendecomposition [...] is often the computational bottleneck, requiring $O(n^3)$ time. Modern multi-core machines can compute eigendecompositions up to $n \approx 1,000$ at interactive speeds of a few seconds, or larger problems up to $n \approx 10,000$ in around ten minutes.

A faster, $O(nk^2)$ algorithm for sampling elementary DPPs was given as Alg. 2 of [22]. We will later show that this approach is equivalent to a small modification of a diagonally-pivoted, rank-revealing, left-looking, Cholesky factorization [23], where the pivot index is chosen at each iteration by sampling from the probability distribution implied by the diagonal of the remaining submatrix (which is maintained out-of-place).

Researchers have begun proposing algorithms which directly sample from Hermitian marginal kernels by recursively conditioning on index inclusion decisions, such as the *sequential sampling* algorithm of [24] (Cf. [25] for greedy, maximum-likelihood sampling). The primary contribution of this manuscript is to simplify these proposals by demonstrating that they are a simple modification of an unpivoted LDL^H factorization (where L is unit lower-triangular and D is real diagonal) and to extend them to non-Hermitian marginal kernels using an unpivoted LU factorization. It is then demonstrated that high-performance factorizations techniques [26–28] lead to orders of magnitude accelerations, and that sparse-direct techniques [29–31] can yield further orders of magnitude speedups.

2. Prototype factorization-based DPP sampling

To describe factorization processees, we will extend our earlier notation that, for an $n \times n$ matrix K and an index subset $Y \subseteq [n]$, K_Y refers to the restriction of K to the row and column indices

of Y. Given a second index subset $Z \subseteq [n]$, $K_{Y,Z}$ will denote the restriction of K to the $|Y| \times |Z|$ submatrix consisting of the rows indices in Y and the column indices in Z. And we use the notation $[j:k]$ to represent the integer range $\{j, j+1, ..., k-1\}$.

Our derivation will make use of a few elementary propositions on the forms of marginal kernels for conditional DPPs. These propositions will allow us to define modifications to the pivots of an LU factorization so that the resulting Schur complements correspond to the marginal kernel of the DPP over the remaining indices, conditioned on the inclusion decisions of the indices corresponding to the eliminated pivots.

Proposition 3
Given disjoint subsets $A, B \subseteq [n]$ of the ground set of a DPP with marginal kernel K, almost surely

$$\mathbb{P}[B \subseteq \mathbf{Y} | A \subseteq \mathbf{Y}] = \det(K_B - K_{B,A} K_A^{-1} K_{A,B}).$$

Proof. If $A \subseteq \mathbf{Y}$, then $\det(K_A) = \mathbb{P}[A \subseteq \mathbf{Y}] > 0$ almost surely, so we may perform a two-by-two block LU decomposition

$$\begin{pmatrix} K_A & K_{A,B} \\ K_{B,A} & K_B \end{pmatrix} = \begin{pmatrix} I & 0 \\ K_{B,A} K_A^{-1} & K_B - K_{B,A} K_A^{-1} K_{A,B} \end{pmatrix} \begin{pmatrix} K_A & K_{A,B} \\ 0 & I \end{pmatrix}.$$

That $\det : GL(n, \mathbb{C}) \mapsto (\mathbb{C}, \times)$ is a homomorphism yields

$$\det(K_{A \cup B}) = \det(K_A) \det(K_B - K_{B,A} K_A^{-1} K_{A,B}).$$

The result then follows from the definition of conditional probabilities for a DPP:

$$\mathbb{P}[B \subseteq \mathbf{Y} | A \subseteq \mathbf{Y}] = \frac{\mathbb{P}[A, B \subseteq \mathbf{Y}]}{\mathbb{P}[A \subseteq \mathbf{Y}]} = \frac{\det(K_{A \cup B})}{\det(K_A)}.$$

□

Proposition 4
Given disjoint $a \in [n]$ and $B \subset [n]$ for a ground set $[n]$ of a DPP with marginal kernel K, almost surely

$$\mathbb{P}[B \subset \mathbf{Y} | a \notin \mathbf{Y}] = \det(K_B - K_{B,a} (K_a - 1)^{-1} K_{a,B}).$$

Proof.

$$\begin{aligned}
\mathbb{P}[B \subset \mathbf{Y} | a \notin \mathbf{Y}] &= \frac{\mathbb{P}[a \notin \mathbf{Y} | B \subset \mathbf{Y}] \mathbb{P}[B \subset \mathbf{Y}]}{\mathbb{P}[a \notin \mathbf{Y}]} \\
&= \frac{(1 - \mathbb{P}[a \in \mathbf{Y} | B \subset \mathbf{Y}]) \mathbb{P}[B \subset \mathbf{Y}]}{1 - \mathbb{P}[a \in \mathbf{Y}]} \\
&= \frac{\det(K_B) - \det(K_{a \cup B})}{1 - K_a} \\
&= \det(K_B)(1 - K_{a,B} \frac{K_B^{-1}}{K_a - 1} K_{B,a}) \\
&= \det(K_B - K_{B,a} (K_a - 1)^{-1} K_{a,B}),
\end{aligned}$$

where the last equality makes use of the Matrix Determinant Lemma. The formulae are well-defined almost surely. □

Propositions 3 and 4 are enough to derive our direct, non-Hermitian DPP sampling algorithm. But, for the sake of symmetry with Proposition 3, we first generalize to set exclusion:

Proposition 5
Given disjoint subsets $A, B \subset \mathcal{Y}$, almost surely

$$\mathbb{P}[B \subseteq \mathbf{Y} | A \subseteq \mathbf{Y}^c] = \det(K_B - K_{B,A} (K_A - I)^{-1} K_{A,B}).$$

Algorithm 1: Unblocked, right-looking, non-Hermitian DPP sampling. For a sample Y, the returned matrix A will contain the in-place LU factorization of $K - I_{Y^C}$, where I_{Y^C} is the diagonal indicator matrix for the entries not in the sample. Symmetry can be exploited in the outer products when the kernel is Hermitian.

```
sample := []; A := K
for j in range(n):
  sample.append(j) if Bernoulli(A_j) else A_j -= 1
  A_{[j+1:n],j} /= A_j
  A_{[j+1:n]} -= A_{[j+1:n],j} A_{j,[j+1:n]}
return sample, A
```

Proof. The claim follows from recursive formulation of conditional marginal kernels using the previous proposition. The resulting kernel is equivalent to the Schur complement produced from the block LU factorization

$$\begin{pmatrix} K_A - I & K_{A,B} \\ K_{B,A} & K_B \end{pmatrix} = \begin{pmatrix} I & 0 \\ K_{B,A}(K_A - I)^{-1} & K_B - K_{B,A}(K_A - I)^{-1}K_{A,B} \end{pmatrix} \begin{pmatrix} K_A - I & K_{A,B} \\ 0 & I \end{pmatrix},$$

as the subtraction of 1 from each eliminated pivot commutes with the outer product updates. □

Theorem 2 (Factorization-based DPP sampling)
Given a (possibly non-Hermitian) marginal kernel matrix K of order n, an LU factorization of K can be modified as in Algorithm 1 to almost surely provide a sample from DPP(K). This algorithm involves roughly $\frac{2}{3}n^3$ floating-point operations, and the likelihood of any returned sample will be given by the product of the absolute value of the diagonal of the result. If the marginal kernel is Hermitian, the work can be roughly halved by exploiting symmetry in the Schur complement updates.

Proof. We first demonstrate, by induction, that our factorization algorithm samples the DPP generated by the marginal kernel K. The loop invariant is that, at the start of the iteration for pivot index j, $A_{[j:n]}$ represents the equivalence class of kernels for the DPP over indices $[j:n]$ conditioned on the inclusion decisions for indices $0, ..., j-1$.

Since the diagonal entries of a kernel matrix represent the likelihood of the corresponding index being in the sample, the loop invariant implies that index j is kept with the correct conditional probability. Prop. 3 shows that the loop invariant is almost surely maintained when the Bernoulli draw is successful, and Prop. 4 handles the alternative. Thus, the loop invariant holds almost surely, and, upon completion, the proposed algorithm samples each subset with the correct probability by sequentially iterating over each index, making an inclusion decision with the appropriate conditional probability.

The likelihood of a sample produced by the algorithm is thus the product of the likelihoods of the results of the Bernoulli draws: when a draw for a diagonal entry p_j is successful, its probability was p_j, and, when unsuccessful, $1 - p_j$. In both cases, the multiplicative contribution is the absolute value of the final state of the j'th diagonal entry. □

Theorem 2 provides us with another interpretation of the generic DPP kernel admissibility condition of [19]:

Proof of Proposition 1. When Algorithm 1 produces a sample Y, the resulting upper and strictly-lower triangular portions of the resulting matrix respectively contain the U and strictly-lower

portion of the unit-diagonal L from the LU factorization of $K - I_{Y^C}$. And we have:

$$\det(K - I_{Y^C}) = \det(U) = \prod_{j=0}^{n-1} U_j = \prod_{j \in Y} p_j \prod_{j \in Y^C} (p_j - 1),$$

where p_j is the inclusion probability for index j, conditioned on the inclusion decisions of indices $0, ..., j-1$.

One can inductively show, working backwards from the last pivot, that $(-1)^{|Y^C|} \det(K - I_{Y^C})$ always being non-negative is equivalent to all potential pivots produced by our sampling algorithm, the set of conditional inclusion probabilities, living in $[0, 1]$. Otherwise, there would exist an index inclusion decision change which would not change the sign of $\det(K - I_{Y^C})$. □

By replacing the Bernoulli samples of algorithm 2 with the maximum-likelihood result for each index inclusion, we arrive at a greedy maximum-likelihood sampling algorithm:

Algorithm 2: Unblocked, right-looking, non-Hermitian, greedy maximum-likelihood DPP sampling. For a sample Y, the returned matrix A will contain the in-place LU factorization of $K - I_{Y^C}$, where I_{Y^C} is the diagonal indicator matrix for the entries not in the sample. Symmetry can be exploited in the outer products when the kernel is Hermitian.

```
sample := []; A := K
for j in range(n):
  sample.append(j) if A_j ≥ 1/2 else A_j -= 1
  A_{[j+1:n],j} /= A_j
  A_{[j+1:n]} -= A_{[j+1:n],j} A_{j,[j+1:n]}
return sample, A
```

In both cases, the same specializations that exist for modifying an unpivoted LU factorization into a Cholesky or (unpivoted) LDL^H or LDL^T factorization apply to our DPP sampling algorithms. And as we will see in the next two sections, so do high-performance dense and sparse-direct factorization techniques.

Theorem 3 (Factorization-based elementary DPP sampling)
Given a marginal kernel matrix K of order n which is an orthogonal projection of rank k, performing k steps of diagonally-pivoted Cholesky factorization, where each pivot index is chosen by sampling from the diagonal as in Algorithm 3, is equivalent to sampling from DPP(K). This approach has complexity $O(nk^2)$ and almost surely completes and returns a sample of cardinality k; the likelihood of the resulting sample is the square of the product of the first k diagonal entries of the partially factored matrix.

Proof. Alg. 3 is, up to permutation, algebraically equivalent to the sampling phase of Alg. 2 of [22]: for the sake of simplicity, it assumes the Gramian is preformed rather than forming it on the fly from the factor. That the probability of a cardinality k sample Y is, almost surely, equal to the product of the squares of the first k diagonal entries of the result follows from recognizing that the lower triangle of the top-left $k \times k$ submatrix will be the Cholesky factor of K_Y, and

$$\mathbb{P}[Y = \mathbf{Y}] = \mathbb{P}[Y \subseteq \mathbf{Y}] = \det(K_Y) = \det(L_Y L_Y^H) = \det(L_Y) \det(L_Y^H) = \prod_{j=0}^{k-1} A_j^2.$$

□

The permutations in Alg. 3 were introduced to solidify the connection to a traditional diagonally-pivoted Cholesky factorization. There is the additional benefit of simplifying the usage of Basic Linear Algebra Subprograms (BLAS) [32–34] calls for the matrix/vector products. But

Algorithm 3: Unblocked, left-looking, diagonally-pivoted, Cholesky-based sampling of a Hermitian Determinantal Projection Process. For a sample Y, the returned matrix will contain the in-place Cholesky factorization of K_Y. The computational cost is roughly $nk^2/3$.

```
A := K; d := diag(K); orig_indices := [0:n]
for j in range(k):
  # Sample the pivot index and perform the permutations
  Draw index t from [j:n] with probability d_t/(k-j)
  Perform Hermitian swap of indices j and t of A
  Swap positions j and t of orig_indices and d
  A_j := sqrt(d_j)
  if j == k - 1:
    break
  # Form the new column and update the remaining diagonal
  A_{[j+1:n],j} -= A_{[j+1:n],[0:j]} A^H_{j,[0:j]}
  for t in range(j+1, n):
    A_{t,j} /= A_j
    d_t -= |A_{t,j}|^2
return orig_indices[0:k], A_{[0:k]}
```

the bulk of the work of this approach will be in rank-one updates, which have essentially no data reuse, and are therefore not performant on modern machines, where the peak floating-point performance is substantially faster than what can be read directly from main memory. The Linear Algebra PACKage (LAPACK) [26] was introduced in 1990 as proof that dense factorizations can be recast in terms of matrix/matrix multiplications; we present analogues in the following section, as well as multi-core, tiled extensions similar to [27,28].

3. High-performance, dense, factorization-based DPP sampling

The main idea of LAPACK [26] is to reorganize the computations within dense linear algebra algorithms so that as much of the work as possible is recast into composing matrices with nontrivial minimal dimensions. Basic operations rich in such matrix compositions – typically referred to as *Level 3 BLAS* [34] – are the building blocks of LAPACK. In most cases, such *block sizes* range from roughly 32 to 256, with 64 to 128 being most common.

In the case of triangular factorizations, such as Cholesky, LDL^H, and LU, a *blocked* algorithm can be produced from the unblocked algorithm by carefully matricizing each of the original operations. In the case of a right-looking LU factorization without pivoting, one arrives at Alg. 4, where A_{J_1} takes the place of the scalar pivot and must be factored – typically, using the unblocked algorithm being generalized – before its components are used to solve against A_{J_2,J_1} and A_{J_1,J_2}.

Algorithm 4: Blocked LU factorization without pivoting.

```
j := 0
while j < n:
  bsize := min(blocksize, n - j)
  J_1 = [j : j + bsize]; J_2 = [j + bsize : n]
  A_{J_1} = unblocked_lu(A_{J_1})
  A_{J_2,J_1} := A_{J_2,J_1} triu(A_{J_1})^{-1}
```

```
    A_{J1,J2} := unit_tril(A_{J1})^{-1} A_{J1,J2}
    A_{J2} -= A_{J2,J1} A_{J1,J2}
    j += bsize
return sample, A
```

The correct form of these solves can be derived via the relationship:

$$\begin{pmatrix} A_1 & A_{1,2} \\ A_{2,1} & A_2 \end{pmatrix} = \begin{pmatrix} L_1 & 0 \\ L_{2,1} & L_2 \end{pmatrix} \begin{pmatrix} U_1 & U_{1,2} \\ 0 & U_2 \end{pmatrix} = \begin{pmatrix} L_1U_1 & L_1U_{1,2} \\ L_{2,1}U_1 & L_{2,1}U_{1,2} + L_2U_2 \end{pmatrix},$$

where we used shorthand of the form $A_{1,2}$ to represent A_{J_1,J_2}. An unblocked factorization can be used to compute the triangular factors L_1 and U_1 of the diagonal block A_1, and then triangular solves of the form $L_{2,1} := A_{2,1}U_1^{-1}$ and $U_{1,2} := L_1^{-1}A_{1,2}$ yield the two panels. After forming the Schur complement $S_2 := A_2 - L_{2,1}U_{1,2}$, the problem has been reduced to an LU factorization with fewer variables – that of $L_2U_2 = S_2$. Asymptotically, all of the work is performed in the outer-product updates which form the Schur complements.

The conversion of Alg. 1, an unblocked DPP sampler, into Alg. 5, a blocked DPP sampler, is essentially identical to the formulation of Alg. 4 from an unblocked LU factorization. We emphasize that, while it is well-known that LU factorizations without pivoting fail on large classes of nonsingular matrices – for example, any matrix with a zero in the top-left position – the analogue for DPP sampling succeeds almost surely for any marginal kernel.

Algorithm 5: Blocked, factorization-based, non-Hermitian DPP sampling. For a sample Y, the returned matrix A will contain the in-place LU factorization of $K - I_{Y^C}$, where I_{Y^C} is the diagonal indicator matrix for the entries not in the sample, $\mathbf{Y}$. The usual specializations from LU to LDL^H factorization applies if the kernel is Hermitian.

```
sample := []; A := K; j := 0
while j < n:
    bsize := min(blocksize, n - j)
    J1 = [j : j + bsize]; J2 = [j + bsize : n]
    subsample, A_{J1} = unblocked_dpp(A_{J1})
    sample.append(subsample + j)
    A_{J2,J1} := A_{J2,J1} triu(A_{J1})^{-1}
    A_{J1,J2} := unit_tril(A_{J1})^{-1} A_{J1,J2}
    A_{J2} -= A_{J2,J1} A_{J1,J2}
    j += bsize
return sample, A
```

Beyond the order-of-magnitude improvement provided by such algorithms, even on a single core of a modern computer, they also simplify the incorporation of parallelism. Lifting algorithms into blocked form allows for each core to be dynamically assigned tasks corresponding to individual updates of a *tile* [27,28], a roughly tile_size × tile_sizesubmatrix which, in our experiments, was typically most performant for tile_size = 256.

Within the context of Alg. 5, our tiled algorithm assigns each `unblocked_dpp` call an individual OpenMP 4.0 [35] task which depends upon the last Schur complement update of its tile. The triangular solves against U_{J_1} are split into submatrices mostly of the form tile_size × block_size, those of the solves against L_{J_1} are roughly of the transpose dimensions, while the Schur complement tasks are roughly of size tile_size × tile_size. In each case, tile reads and writes are scheduled using dependencies on previous updates of the same tile.

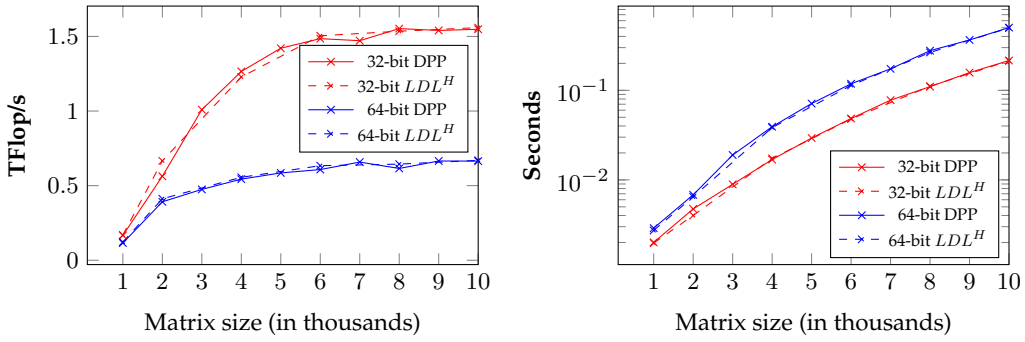


Figure 5: Dense, real LDL^H-based DPP sampling performance; tile sizes of 128 and 256 were used for matrix sizes less than or equal to, and greater than, 2000, respectively.

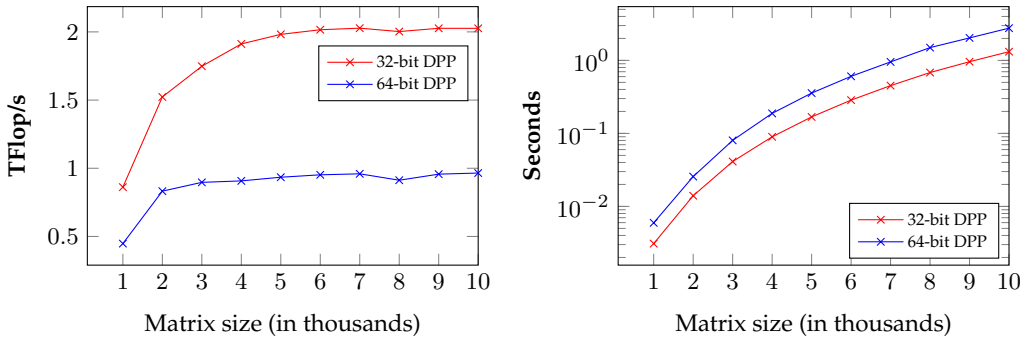


Figure 6: Dense, complex LU-based DPP sampling performance. For single-precision, a tile size of 128 was used up to matrix sizes of 3000, and tile sizes of 256 were used thereafter. For double-precision, the switch occurred above matrices of size 4000.

The performance of such a dynamically-scheduled parallelization of the Hermitian specialization of Alg. 5 is demonstrated for arbitrary dense, Hermitian marginal kernels on a 16-core Intel i9-7960x in Fig. 5. The performance of a similarly parallelized unpivoted LDL^H factorization is similarly plotted, and one readily observes that their runtimes are essentially identical. The runtime of Alg. 5 for arbitrary, complex, non-Hermitian marginal kernels is similarly shown in Fig. 6. For purpose of comparison, we note that the double-precision High-Performance LINPACK benchmark [36] achieves roughly 1 TFlop/second on this machine.

It is worth emphasizing the compounding performance gains from both the formulation of DPP sampling as a small modification of a level-3 BLAS focused dense matrix factorization and from the 16-way parallelism. When combined, a factor of 2500x speedup is observed relative to the timings on the same machine of DPPy v0.1.0 [37] when sampling a spectrally-preprocessed arbitrary, dense Hermitian 5000×5000 matrix. Similar speedups exist relative to the timings of both the "sequentially thinned" and spectrally-preprocessed algorithms of [24].

Timings for the $\mathbb{Z}^2$ and hexagonal-tiling Uniform Spanning Tree DPPs can be extracted from Fig. 5 via the formulae $n \approx 2d^2$ and $n \approx 6d^2$, respectively, where d is the order of the grid. And timings for the uniform domino tilings can be recovered from Fig. 6 via the formula $n \approx 4d^2$. When computing results for the latter, it was noticed that sampling domino tilings from the Kenyon formula DPP in single-precision, the results are essentially always inconsistent once the diamond size exceeds 60, but no consistencies have yet been observed with double-precision sampling. See Fig. 7 for an example.

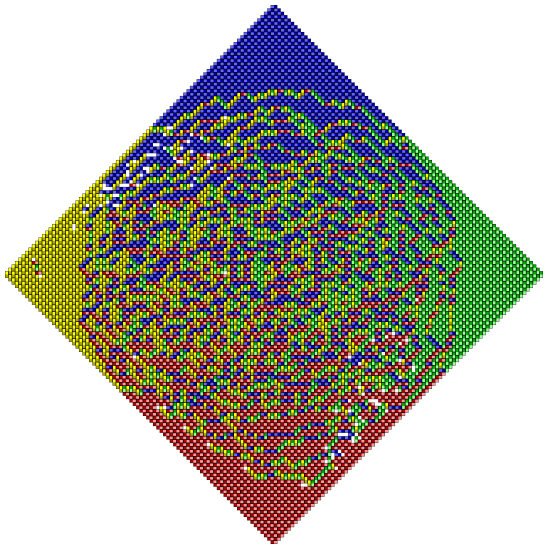


Figure 7: Corrupted single-precision sample from an Aztec diamond of size 80. Missing tiles are clearly visible in the image.

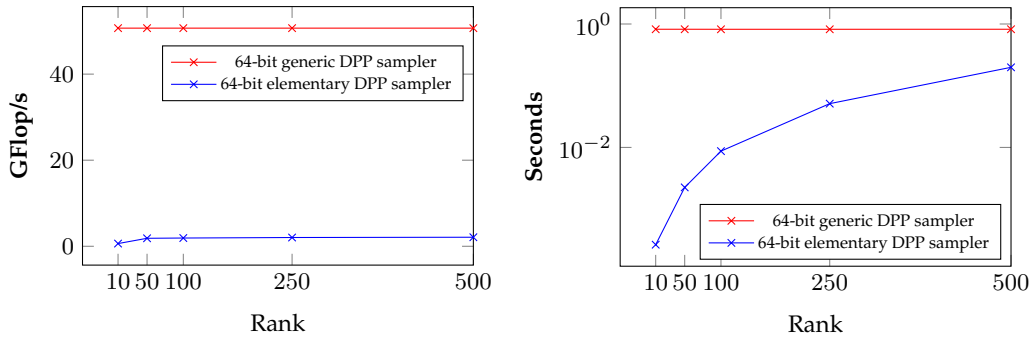


Figure 8: Sequential, dense, real LDL^H-based elementary DPP sampling performance for a ground set of size 5000 and varying rank. Multithreaded BLAS was explicitly disabled to ensure execution on a single core of the Intel i9-7960x.

Such a numerical instability in an unpivoted dense matrix factorization would lead any numerical analyst to wonder if dynamic pivoting can mitigate error accumulation. Indeed, our factorization-based approach frees us to perform arbitrary diagonal pivoting, as long as the pivot is decided before its corresponding Bernoulli draw. While it will be the subject of future work, the author conjectures that *maximum-entropy* pivoting, that is, pivoting towards a diagonal entry with conditional probability as close to $\frac{1}{2}$ as possible, would be the most beneficial, as it maximizes the magnitude of the smallest possible pivot.

Before moving on to sparse-direct DPP sampling, we demonstrate the multiple orders of magnitude speedup that are possible for Determinantal Projection Processes of sufficiently low rank. Unlike our other experiments, Fig. 8 only executes on a single core, as the unblocked, left-looking rank-revealing elementary DPP sampling approach of Alg. 3 spends the majority of its time in matrix/vector multiplication. Studying the benefits of parallelizations of elementary DPP samplers is left for future work.

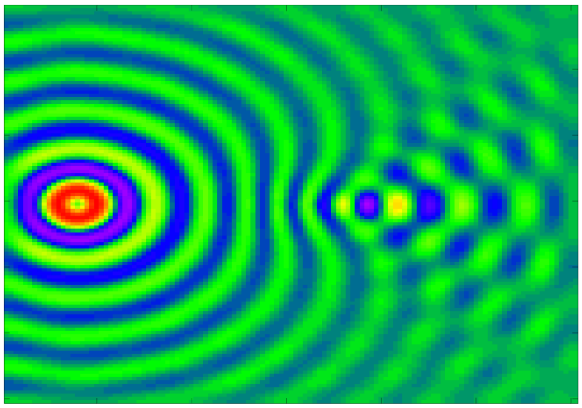


Figure 9: A 2D slice for a solution of a 3D Helmholtz equation discretized with trilinear, hexahedra elements and PML boundary conditions for a Gaussian point source near the boundary and a converging lens near the center of the domain.

4. Sparse-direct DPP sampling

As was demonstrated in Fig. 5, the performance of a generic Hermitian DPP sampler can be made to match that of a high-performance Hermitian matrix factorization. This correspondence should not be a surprise, as we have shown that matrix factorizations can be trivially modified to yield Hermitian and non-Hermitian DPP samplers alike. The same insight applies to the conversion of sparse-direct, unpivoted LDL^H factorization into sparse-direct Hermitian DPP samplers. Such an implementation, mirroring many of the techniques from CholMod [31], was implemented within Catamari.

At a high level, the approach is use a dynamically scheduled – via OpenMP 4.0 task scheduling – multifrontal method [30,38] when the computational intensity of the factorization is deemed high enough after the symbolic phase, otherwise a sequential, up-looking simplicial approach [31] is used. The multifrontal method is parallelized by using the a tile-based, dynamically scheduled DPP sampler or dense factorization on the diagonal blocks, combined with nested OpenMP tasks launched for each (relaxed) supernode's [30] subtree.

The majority of the development time was dedicated to performance improvements such as parallelizing the symbolic factorization, avoiding unnecessary initializations of large buffers, and truncating nested OpenMP task assignments in subtrees of the elimination tree [29] whose total work is below a fixed threshold. Such a work-based task recursion mechanism was critical to ensuring high-performance on problems from Davis's Sparse Matrix Collection [39] which result in heavily imbalanced elimination trees.

The author was unaware of any prominent examples of sparse marginal DPP kernels. But since sparse-direct techniques can lead to drastic speedups and memory decreases relative to dense approaches, and formulations with such structure might appear in the future, we implemented tandem sparse-direct factorization and sparse-direct Hermitian DPP samplers to connect performance results of synthetic sparse DPPs and discretized Partial Differential Equation solvers. Such a statically-pivoted sparse-direct LDL^H factorization is also the critical computational kernel of primal-dual Interior Point Methods [40,41], where the first-order optimality conditions are symmetric quasi-semidefinite [42,43].

Performance results for the dynamically scheduled sparse-direct solver on a 3D Helmholtz equation discretized with trilinear, hexahedral elements Fig. 9 are shown in Fig. 10; results for a 2D DPP analogue, where the marginal kernel takes the form of a scaled 2D Laplacian, are shown in Fig. 11.

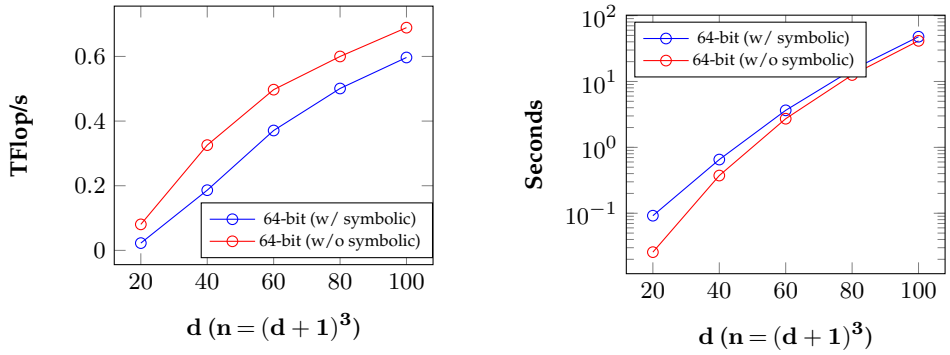
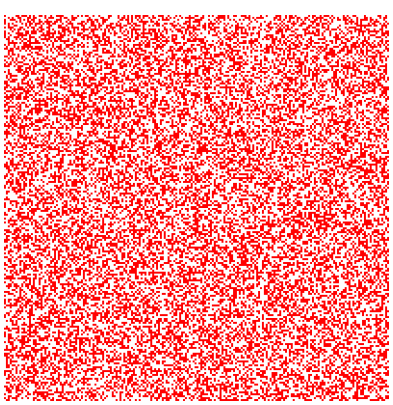
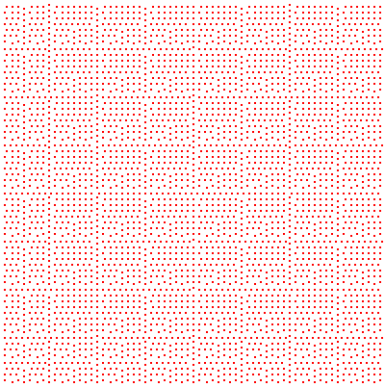


Figure 10: Sparse-direct performance on trilinear, hexahedral discretization of 3D Helmholtz equation with Perfectly Matched Layer boundary conditions.



(a) Sample with log-likelihood of -27472.2.



(b) Greedy maximum-likelihood sample with log-likelihood of -26058.

Figure 11: Samples from $-\sigma\Delta$ over a 200 x 200 grid, where $\sigma = 0.72$.

In the case of the 2D Laplacian sparse-direct DPP sampling over a 200×200 grid, the timings were roughly 0.01 seconds on the 16-core i9-7960x. Extrapolating from Fig. 5 to a dense matrix size of $40,000$, it is clear that several more orders of magnitude of efficiency are gained with the sparse-direct formulation.

We close this section by noting that static pivoting should apply equally well to non-Hermitian sparse-direct DPP sampling, as the nature of small pivots being, by definition, rare, suggests a certain degree of stability. As in the case of dense DPP sampling, probabilistic error analysis is warranted.

5. Conclusions

A unified, generic framework for sequentially sampling both Hermitian and non-Hermitian Determinantal Point Processes directly from their marginal kernel matrices was presented. The prototype algorithm consisted of a small modification of an unpivoted LU factorization – in the Hermitian case, it reduces to a modification of an unpivoted LDL^H factorization – and high-performance, tiled algorithms were presented in the dense case, and an analogue of CholMod [31] was presented for the sparse case. Further, it was shown that both greedy, maximum-likelihood

sampling and elementary DPP sampling can be understood and efficiently implemented using small modifications of traditional matrix factorization techniques.

In addition to generalizing sampling algorithms from Hermitian to non-Hermitian marginal kernels, the proposed approaches were implemented within the open source, permissively licensed, Catamari package and shown to lead to orders of magnitude speedups, even in the dense regime. The Hermitian sparse-direct DPP sampler leads to further asymptotic speedups. Future work includes theoretical and empirical exploration of the stability of dynamic pivoting techniques, both for dense and sparse-direct factorization-based DPP sampling, including the incorporation of maximum-entropy pivoting for large instances of Kenyon-formula Aztec domino tilings.

Further exploration of the performance tradeoffs between left-looking and right-looking elementary DPP sampling, and at which rank it becomes beneficial to use our generic sampling approach, is warranted – especially on multi-core and GPU-accelerated architectures. And, lastly, the incorporation of a tiled extension of [44] for converting a Hermitian L-ensemble kernel into a marginal kernel via $K = I - (L + I)^{-1}$ [22] is a natural extension of our proposed techniques and could be expected to require the equivalent of 3 samples worth of time due to the computational complexity.

Reproducibility

The experiments in this paper were performed using version 0.2.5 of Catamari, with Intel's Math Kernel Library BLAS on an Ubuntu 18.04 workstation with 64 GB of RAM and an Intel i9-7960x processor. Catamari can be downloaded from `https://gitlab.com/hodge_star/catamari`. Figs. 1 and 2 were generated using the TIFF output from example driver `example/uniform_spanning_tree.cc`, while Figs. 3, 4 and 7 similarly used `example/aztec_diamond.cc`.

Figs. 5 and 6 were generated using data from `example/dense_dpp.cc` and `example/dense_factorization.cc`; we emphasize that explicitly setting both `OMP_NUM_THREADS` and `MKL_NUM_THREADS` to 16 on our 16-core machine led to significant performance improvements (as opposed to the default exploitation of hyperthreading). And Fig. 8 was produced with `example/dense_elementary_dpp.cc` – with `OMP_NUM_THREADS` and `MKL_NUM_THREADS` explicitly set to one. The algorithmic `block_size` was also set to a value at least as large as the rank so that a left-looking approach was used throughout.

Figs. 9 and 10 were generated from `example/helmholtz_3d_pml.cc`, with the 2D-slice of the former taken slightly away from the center plane. Fig. 11 was output from the TIFF support of `example/dpp_shifted_2d_negative_laplacian.cc`.

Acknowledgements

The author would like to sincerely thank Guillaume Gautier and Rémi Bardenet for detailed comments on an early draft of this manuscript, as well as Jennifer A. Gillenwater for suggesting the $I - (L + I)^{-1}$ conversion from an L-ensemble and Alex Kulesza for answering some of my early DPP questions.

References

1. Odile Macchi.
   The coincidence approach to stochastic point processes.
   Advances in Applied Probability, 7(1):83–122, 1975.
2. Alex Kulesza and Ben Taskar.
   Determinantal point processes for machine learning.
   Foundations and Trends in Machine Learning, 5(2–3):123–286, 2012.
3. Odile Macchi.

The Fermion Process — A Model of Stochastic Point Process with Repulsive Points, pages 391–398. Springer Netherlands, Dordrecht, 1977.

4. M. L. Mehta and M. Gaudin.
On the density of eigenvalues of a random matrix.
Nuclear Physics, 18:420–427, 1960.
5. Freeman J. Dyson.
Statistical theory of the energy levels of complex systems. i.
Journal of Mathematical Physics, 3(1):140–156, 1962.
6. Freeman J. Dyson.
Statistical theory of the energy levels of complex systems. ii.
Journal of Mathematical Physics, 3(1):157–165, 1962.
7. Freeman J. Dyson.
Statistical theory of the energy levels of complex systems. iii.
Journal of Mathematical Physics, 3(1):166–175, 1962.
8. Jean Ginibre.
Statistical Ensembles of Complex, Quaternion and Real Matrices.
J. Math. Phys., 6:440–449, 1965.
9. Persi Diaconis.
Patterns in eigenvalues: the 70th Josiah Willard Gibbs lecture.
Bull. Amer. Math. Soc., 40:155–178, 2003.
10. Alan Edelman and N. Raj Rao.
Random matrix theory.
Acta Numerica, 14:233–297, 2005.
11. Robert Burton and Robin Pemantle.
Local characteristics, entropy and limit theorems for spanning trees and domino tilings via transfer-impedances.
The Annals of Probability, 21(3):1329–1371, 1993.
12. Itai Benjamini, Russell Lyons, Yuval Peres, and Oded Schramm.
Uniform spanning forests.
Ann. Probab., 29(1):1–65, 02 2001.
13. Kurt Johansson.
Determinantal processes with number variance saturation.
Communications in Mathematical Physics, 252(1):111–148, Dec 2004.
14. P. W. Kasteleyn.
The statistics of dimers on a lattice : I. The number of dimer arrangements on a quadratic lattice.
Physica, 27:1209–1225, December 1961.
15. H. N. V. Temperley and Michael E. Fisher.
Dimer problem in statistical mechanics-an exact result.
The Philosophical Magazine: A Journal of Theoretical Experimental and Applied Physics, 6(68):1061–1063, 1961.
16. P. W. Kasteleyn.
Dimer statistics and phase transitions.
Journal of Mathematical Physics, 4(2):287–293, 1963.
17. Sunil Chhita, Kurt Johansson, and Benjamin Young.
Asymptotic domino statistics in the aztec diamond.
Ann. Appl. Probab., 25(3):1232–1278, 06 2015.
18. Persi Diaconis and Steven N. Evans.
Immanants and finite point processes.
Journal of Combinatorial Theory, Series A, 91(1):305–321, 2000.
19. Victor-Emmanuel Brunel.
Learning signed determinantal point processes through the principal minor assignment problem.
In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 7365–7374. Curran Associates, Inc., 2018.
20. A Soshnikov.
Determinantal random point fields.

Russian Mathematical Surveys, 55(5):923–975, oct 2000.
21. J. Ben Hough, Manjunath Krishnapur, Yuval Peres, and Balint Virag.
Determinantal processes and independence.
Probab. Surveys, 3:206–229, 2006.
22. Jennifer A. Gillenwater.
Approximate Inference for Determinantal Point Processes.
PhD thesis, University of Pennsylvania, 2014.
23. Nicholas J. Higham.
Analysis of the Cholesky decomposition of a semi-definite matrix.
In *in Reliable Numerical Computation*, pages 161–185. University Press, 1990.
24. Claire Launay, Bruno Galerne, and Agnès Desolneux.
Exact Sampling of Determinantal Point Processes without Eigendecomposition.
arXiv e-prints, page arXiv:1802.08429, Feb 2018.
25. Laming Chen and Guoxin Zhang.
Fast greedy map inference for determinantal point process to improve recommendation diversity.
In *NeurIPS*, 2018.
26. E. Anderson, Z. Bai, J. Dongarra, A. Greenbaum, A. McKenney, J. Du Croz, S. Hammarling, J. Demmel, C. Bischof, and D. Sorensen.
LAPACK: A portable linear algebra library for high-performance computers.
In *Proceedings of the 1990 ACM/IEEE Conference on Supercomputing*, Supercomputing '90, pages 2–11, Los Alamitos, CA, USA, 1990. IEEE Computer Society Press.
27. Ernie Chan, Field G. Van Zee, Paolo Bientinesi, Enrique S. Quintana-Orti, Gregorio Quintana-Orti, and Robert van de Geijn.
SuperMatrix: A multithreaded runtime scheduling system for algorithms-by-blocks.
In *Proceedings of the 13th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '08, pages 123–132, New York, NY, USA, 2008. ACM.
28. Alfredo Buttari, Julien Langou, Jakub Kurzak, and Jack Dongarra.
A class of parallel tiled linear algebra algorithms for multicore architectures.
Parallel Comput., 35(1):38–53, January 2009.
29. Robert Schreiber.
A new implementation of sparse Gaussian elimination.
ACM Trans. Math. Softw., 8(3):256–276, September 1982.
30. Cleve Ashcraft and Roger Grimes.
The influence of relaxed supernode partitions on the multifrontal method.
ACM Trans. Math. Softw., 15(4):291–309, December 1989.
31. Yanqing Chen, Timothy A. Davis, William W. Hager, and Sivasankaran Rajamanickam.
Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate.
ACM Trans. Math. Softw., 35(3):22:1–22:14, October 2008.
32. C. L. Lawson, R. J. Hanson, F. T. Krogh, and D. R. Kincaid.
Algorithm 539: Basic Linear Algebra Subprograms for Fortran usage [F1].
ACM Trans. Math. Softw., 5(3):324–325, September 1979.
33. Jack J. Dongarra, Jeremy Du Croz, Sven Hammarling, and Richard J. Hanson.
An extended set of FORTRAN Basic Linear Algebra Subprograms.
ACM Trans. Math. Softw., 14(1):1–17, March 1988.
34. Jack J. Dongarra, Jeremy Du Croz, Sven Hammarling, and I. S. Duff.
A set of level 3 Basic Linear Algebra Subprograms.
ACM Trans. Math. Softw., 16(1):1–17, March 1990.
35. OpenMP Architecture Review Board.
OpenMP application program interface version 4.0, 2013.
36. Jack J. Dongarra, Piotr Luszczek, and Antoine Petitet.
The linpack benchmark: Past, present, and future, 2002.
37. Guillaume Gautier, Rémi Bardenet, and Michal Valko.
DPPy: Sampling determinantal point processes with Python.
CoRR, abs/1809.07258, 2018.
38. I. S. Duff and J. K. Reid.
The multifrontal solution of indefinite sparse symmetric linear.
ACM Trans. Math. Softw., 9(3):302–325, September 1983.

39. Timothy A. Davis and Yifan Hu.
The university of florida sparse matrix collection.
ACM Trans. Math. Softw., 38(1):1:1–1:25, December 2011.
40. Anna Altman and Jacek Gondzio.
HOPDM – a higher order primal-dual method for large scale linear programming.
European Journal of Operational Research, 66(1):158 – 160, 1993.
41. Jacek Gondzio.
Multiple centrality corrections in a primal-dual method for linear programming.
Computational Optimization and Applications, 6(2):137–156, Sep 1996.
42. R. Vanderbei.
Symmetric quasidefinite matrices.
SIAM Journal on Optimization, 5(1):100–113, 1995.
43. A. George, K. Ikramov, and A. Kucherov.
Some properties of symmetric quasi-definite matrices.
SIAM Journal on Matrix Analysis and Applications, 21(4):1318–1323, 2000.
44. Paolo Bientinesi, Brian Gunter, and Robert A. van de Geijn.
Families of algorithms related to the inversion of a symmetric positive definite matrix.
ACM Trans. Math. Softw., 35(1):3:1–3:22, July 2008.