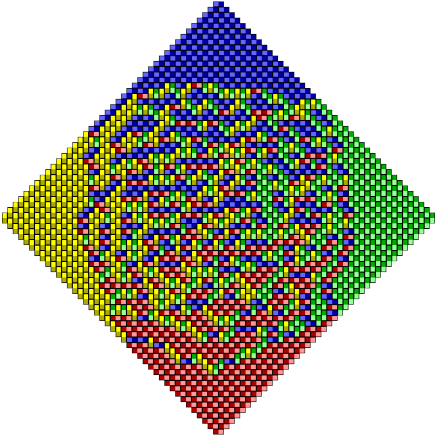


Introduction to Determinantal Point Processes

Jack Poulson (Hodge Star Scientific Computing)
Aussois, France, June 20, 2019

Overview

- We are nominally discussing them due to their popularization, by [Kulesza/Taskar-2012], for the diversification step of a recommendation system. But, in that context, set size is typically only a few hundred.
- We will draw strong connection between techniques for efficiently **factoring matrices** and for **sampling structured subsets** of a ground set.
- The basic bridge: forming a **Schur complement** equates to forming a representation of a **conditional distribution**.
- One can import HPC techniques [P-2019], such as **DAG-scheduled** dense and sparse-direct **blocked algorithms**, from factorizations to **Determinantal Point Processes**.
- Implementations are available in the permissively licensed, header-only C++14 package Catamari [P-2018] available at hodgestar.com/catamari and (partly) in DPPy github.com/guilgautier/DPPy.

Overview

- We are nominally discussing them due to their popularization, by [Kulesza/Taskar-2012], for the diversification step of a recommendation system. But, in that context, set size is typically only a few hundred.
- We will draw strong connection between techniques for efficiently **factoring matrices** and for **sampling structured subsets** of a ground set.
- The basic bridge: forming a **Schur complement** equates to forming a representation of a **conditional distribution**.
- One can import HPC techniques [P-2019], such as **DAG-scheduled** dense and sparse-direct **blocked algorithms**, from factorizations to **Determinantal Point Processes**.
- Implementations are available in the permissively licensed, header-only C++14 package Catamari [P-2018] available at hodgestar.com/catamari and (partly) in DPPy github.com/guilgautier/DPPy.

Overview

- We are nominally discussing them due to their popularization, by [Kulesza/Taskar-2012], for the diversification step of a recommendation system. But, in that context, set size is typically only a few hundred.
- We will draw strong connection between techniques for efficiently **factoring matrices** and for **sampling structured subsets** of a ground set.
- The basic bridge: forming a **Schur complement** equates to forming a representation of a **conditional distribution**.
- One can import HPC techniques [P-2019], such as **DAG-scheduled** dense and sparse-direct **blocked algorithms**, from factorizations to **Determinantal Point Processes**.
- Implementations are available in the permissively licensed, header-only C++14 package Catamari [P-2018] available at hodgestar.com/catamari and (partly) in DPPy github.com/guilgautier/DPPy.

Overview

- We are nominally discussing them due to their popularization, by [Kulesza/Taskar-2012], for the diversification step of a recommendation system. But, in that context, set size is typically only a few hundred.
- We will draw strong connection between techniques for efficiently **factoring matrices** and for **sampling structured subsets** of a ground set.
- The basic bridge: forming a **Schur complement** equates to forming a representation of a **conditional distribution**.
- One can import HPC techniques [P-2019], such as **DAG-scheduled** dense and sparse-direct **blocked algorithms**, from factorizations to **Determinantal Point Processes**.
- Implementations are available in the permissively licensed, header-only C++14 package Catamari [P-2018] available at hodgestar.com/catamari and (partly) in DPPy github.com/guilgautier/DPPy.

Overview

- We are nominally discussing them due to their popularization, by [Kulesza/Taskar-2012], for the diversification step of a recommendation system. But, in that context, set size is typically only a few hundred.
- We will draw strong connection between techniques for efficiently **factoring matrices** and for **sampling structured subsets** of a ground set.
- The basic bridge: forming a **Schur complement** equates to forming a representation of a **conditional distribution**.
- One can import HPC techniques [P-2019], such as **DAG-scheduled** dense and sparse-direct **blocked algorithms**, from factorizations to **Determinantal Point Processes**.
- Implementations are available in the permissively licensed, header-only C++14 package Catamari [P-2018] available at hodgestar.com/catamari and (partly) in DPPy github.com/guilgautier/DPPy.

Main idea: pivots as inclusion probabilities

Sampling a DPP can be reinterpreted as 'factoring' a class of matrices such that the j'th pivot is the probability of including the j'th item.

Flip a coin weighted by the pivot to determine inclusion:

- If the item is kept, proceed as in an LU/LDL factorization.
- If the item is dropped, take the pivot's complement in $[0, 1]$ and negate – i.e., subtract one – and proceed as normal.

The likelihood of the sample is thus the product of the absolute value of the diagonal of the 'factorization'.

Essentially all high-performance techniques for dense and sparse-direct factorizations therefore carry over.

Main idea: pivots as inclusion probabilities

Sampling a DPP can be reinterpreted as 'factoring' a class of matrices such that the j'th pivot is the probability of including the j'th item.

Flip a coin weighted by the pivot to determine inclusion:

- If the item is kept, proceed as in an LU/LDL factorization.
- If the item is dropped, take the pivot's complement in $[0, 1]$ and negate – i.e., subtract one – and proceed as normal.

The likelihood of the sample is thus the product of the absolute value of the diagonal of the 'factorization'.

Essentially all high-performance techniques for dense and sparse-direct factorizations therefore carry over.

Main idea: pivots as inclusion probabilities

Sampling a DPP can be reinterpreted as 'factoring' a class of matrices such that the j'th pivot is the probability of including the j'th item.

Flip a coin weighted by the pivot to determine inclusion:

- If the item is kept, proceed as in an LU/LDL factorization.
- If the item is dropped, take the pivot's complement in $[0, 1]$ and negate – i.e., subtract one – and proceed as normal.

The likelihood of the sample is thus the product of the absolute value of the diagonal of the 'factorization'.

Essentially all high-performance techniques for dense and sparse-direct factorizations therefore carry over.

Main idea: pivots as inclusion probabilities

Sampling a DPP can be reinterpreted as 'factoring' a class of matrices such that the j'th pivot is the probability of including the j'th item.

Flip a coin weighted by the pivot to determine inclusion:

- If the item is kept, proceed as in an LU/LDL factorization.
- If the item is dropped, take the pivot's complement in $[0, 1]$ and negate – i.e., subtract one – and proceed as normal.

The likelihood of the sample is thus the product of the absolute value of the diagonal of the 'factorization'.

Essentially all high-performance techniques for dense and sparse-direct factorizations therefore carry over.

Main idea: pivots as inclusion probabilities

Sampling a DPP can be reinterpreted as 'factoring' a class of matrices such that the j'th pivot is the probability of including the j'th item.

Flip a coin weighted by the pivot to determine inclusion:

- If the item is kept, proceed as in an LU/LDL factorization.
- If the item is dropped, take the pivot's complement in $[0, 1]$ and negate – i.e., subtract one – and proceed as normal.

The likelihood of the sample is thus the product of the absolute value of the diagonal of the 'factorization'.

Essentially all high-performance techniques for dense and sparse-direct factorizations therefore carry over.

What is meant by a 'structured subset'?

The basic mechanism of a (finite) **Point Process** is to define a probability distribution over the power set of a ground set $[n] = [0, ..., n-1]$.

A **determinantal** point process sets the probability of a subset $J \subseteq [n]$ being in the sample equal to the J-minor of a fixed **marginal kernel matrix**.

The kernel matrix is often assumed Hermitian positive semi-definite – with spectrum in $[0, 1]$, but Hermiticity does not hold in some important cases.

Inadmissible combinations of members of the set can therefore be encoded through linear dependencies in the kernel matrix.

Before diving into the details, it will be instructive to describe some Hermitian and non-Hermitian standard DPPs.

What is meant by a 'structured subset'?

The basic mechanism of a (finite) **Point Process** is to define a probability distribution over the power set of a ground set $[n] = [0, ..., n-1]$.

A **determinantal** point process sets the probability of a subset $J \subseteq [n]$ being in the sample equal to the J-minor of a fixed **marginal kernel matrix**.

The kernel matrix is often assumed Hermitian positive semi-definite – with spectrum in $[0, 1]$, but Hermiticity does not hold in some important cases.

Inadmissible combinations of members of the set can therefore be encoded through linear dependencies in the kernel matrix.

Before diving into the details, it will be instructive to describe some Hermitian and non-Hermitian standard DPPs.

What is meant by a 'structured subset'?

The basic mechanism of a (finite) **Point Process** is to define a probability distribution over the power set of a ground set $[n] = [0, ..., n-1]$.

A **determinantal** point process sets the probability of a subset $J \subseteq [n]$ being in the sample equal to the J-minor of a fixed **marginal kernel matrix**.

The kernel matrix is often assumed Hermitian positive semi-definite – with spectrum in $[0, 1]$, but Hermiticity does not hold in some important cases.

Inadmissible combinations of members of the set can therefore be encoded through linear dependencies in the kernel matrix.

Before diving into the details, it will be instructive to describe some Hermitian and non-Hermitian standard DPPs.

What is meant by a 'structured subset'?

The basic mechanism of a (finite) **Point Process** is to define a probability distribution over the power set of a ground set $[n] = [0, ..., n-1]$.

A **determinantal** point process sets the probability of a subset $J \subseteq [n]$ being in the sample equal to the J-minor of a fixed **marginal kernel matrix**.

The kernel matrix is often assumed Hermitian positive semi-definite – with spectrum in $[0, 1]$, but Hermiticity does not hold in some important cases.

Inadmissible combinations of members of the set can therefore be encoded through linear dependencies in the kernel matrix.

Before diving into the details, it will be instructive to describe some Hermitian and non-Hermitian standard DPPs.

What is meant by a 'structured subset'?

The basic mechanism of a (finite) **Point Process** is to define a probability distribution over the power set of a ground set $[n] = [0, ..., n-1]$.

A **determinantal** point process sets the probability of a subset $J \subseteq [n]$ being in the sample equal to the J-minor of a fixed **marginal kernel matrix**.

The kernel matrix is often assumed Hermitian positive semi-definite – with spectrum in $[0, 1]$, but Hermiticity does not hold in some important cases.

Inadmissible combinations of members of the set can therefore be encoded through linear dependencies in the kernel matrix.

Before diving into the details, it will be instructive to describe some Hermitian and non-Hermitian standard DPPs.

What is meant by a 'structured subset'?

The basic mechanism of a (finite) **Point Process** is to define a probability distribution over the power set of a ground set $[n] = [0, ..., n-1]$.

A **determinantal** point process sets the probability of a subset $J \subseteq [n]$ being in the sample equal to the J-minor of a fixed **marginal kernel matrix**.

The kernel matrix is often assumed Hermitian positive semi-definite – with spectrum in $[0, 1]$, but Hermiticity does not hold in some important cases.

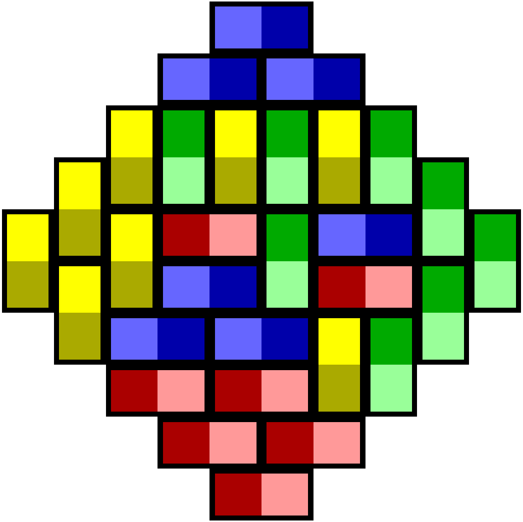
Inadmissible combinations of members of the set can therefore be encoded through linear dependencies in the kernel matrix.

Before diving into the details, it will be instructive to describe some Hermitian and non-Hermitian standard DPPs.

Aztec diamond: $d = 5$

```
$ ./aztec_diamond --diamond_size=5
```

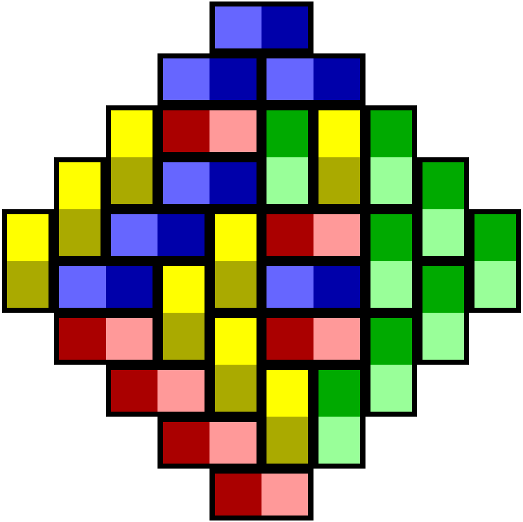
Complex non-Hermitian kernel; Sample likelihoods: $\exp(-10.3972)$



Aztec diamond: $d = 5$

```
$ ./aztec_diamond --diamond_size=5
```

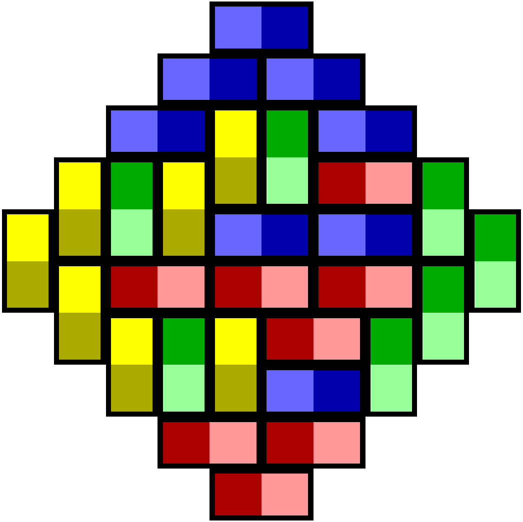
Complex non-Hermitian kernel; Sample likelihoods: $\exp(-10.3972)$



Aztec diamond: $d = 5$

```
$ ./aztec_diamond --diamond_size=5
```

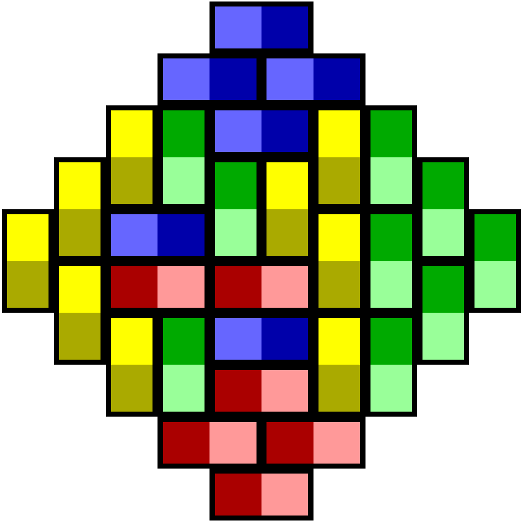
Complex non-Hermitian kernel; Sample likelihoods: $\exp(-10.3972)$



Aztec diamond: $d = 5$

```
$ ./aztec_diamond --diamond_size=5
```

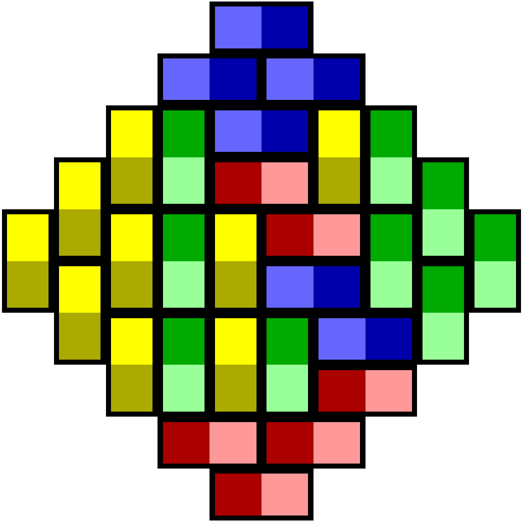
Complex non-Hermitian kernel; Sample likelihoods: $\exp(-10.3972)$



Aztec diamond: $d = 5$

```
$ ./aztec_diamond --diamond_size=5
```

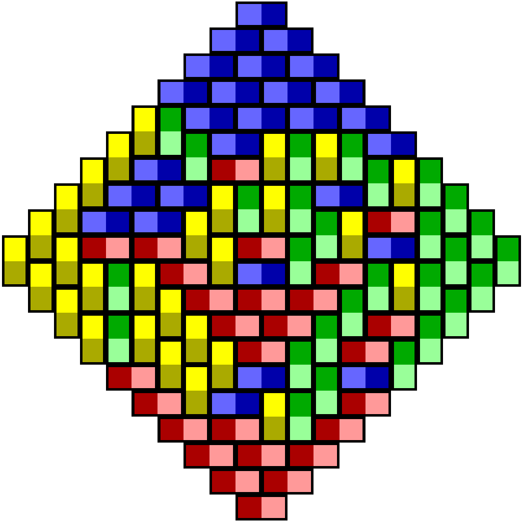
Complex non-Hermitian kernel; Sample likelihoods: $\exp(-10.3972)$



Aztec diamond: $d = 10$

```
$ ./aztec_diamond --diamond_size=10
```

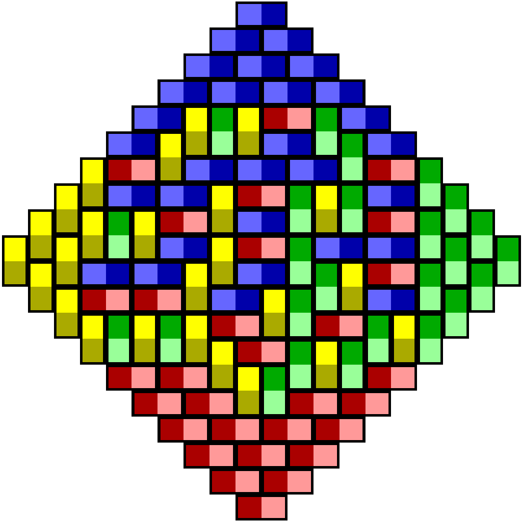
Complex non-Hermitian kernel; Sample likelihoods: $\exp(-38.1231)$



Aztec diamond: $d = 10$

```
$ ./aztec_diamond --diamond_size=10
```

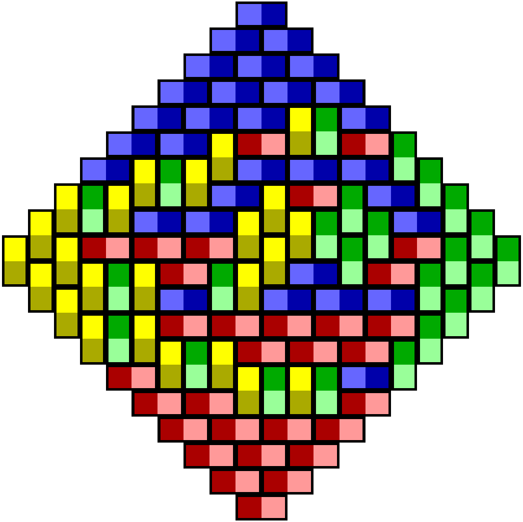
Complex non-Hermitian kernel; Sample likelihoods: $\exp(-38.1231)$



Aztec diamond: $d = 10$

```
$ ./aztec_diamond --diamond_size=10
```

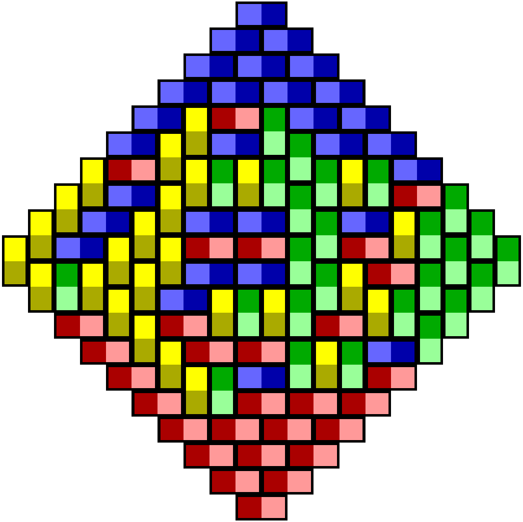
Complex non-Hermitian kernel; Sample likelihoods: $\exp(-38.1231)$



Aztec diamond: $d = 10$

```
$ ./aztec_diamond --diamond_size=10
```

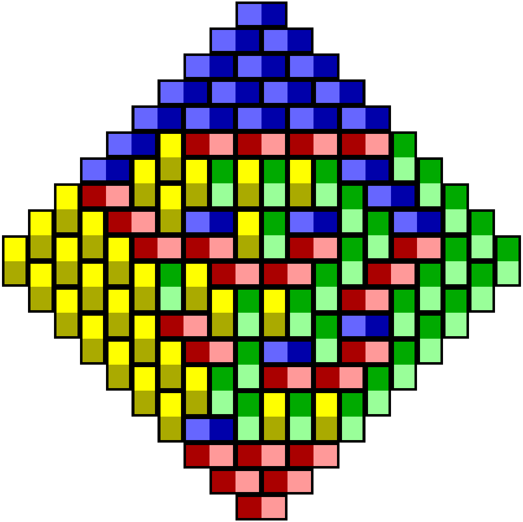
Complex non-Hermitian kernel; Sample likelihoods: $\exp(-38.1231)$



Aztec diamond: $d = 10$

```
$ ./aztec_diamond --diamond_size=10
```

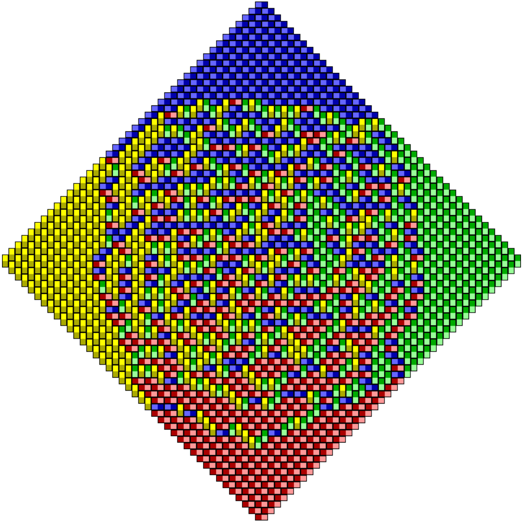
Complex non-Hermitian kernel; Sample likelihoods: $\exp(-38.1231)$



Aztec diamond: $d = 40$

```
$ ./aztec_diamond --diamond_size=40
```

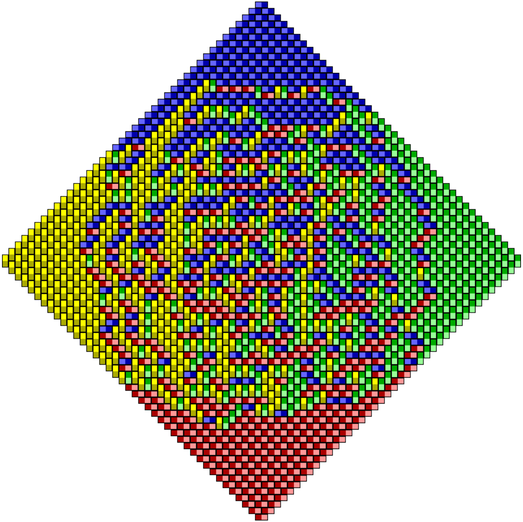
Complex non-Hermitian kernel; Sample likelihoods: $\exp(-568.381)$



Aztec diamond: $d = 40$

```
$ ./aztec_diamond --diamond_size=40
```

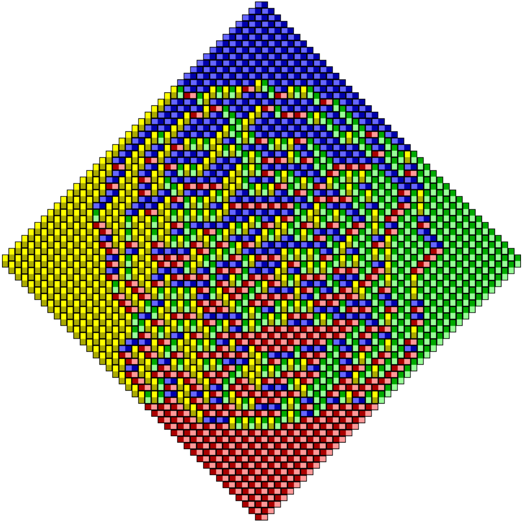
Complex non-Hermitian kernel; Sample likelihoods: $\exp(-568.381)$



Aztec diamond: $d = 40$

```
$ ./aztec_diamond --diamond_size=40
```

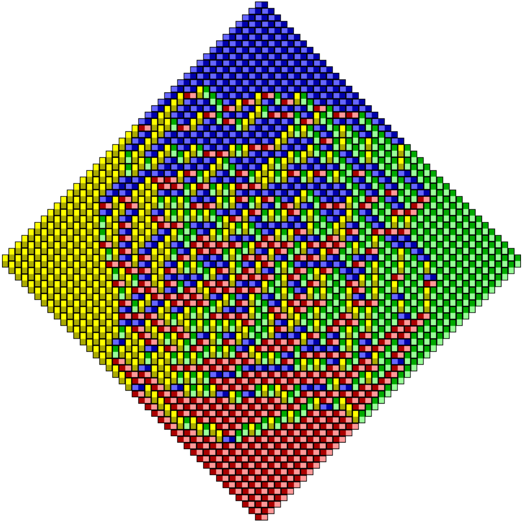
Complex non-Hermitian kernel; Sample likelihoods: $\exp(-568.381)$



Aztec diamond: $d = 40$

```
$ ./aztec_diamond --diamond_size=40
```

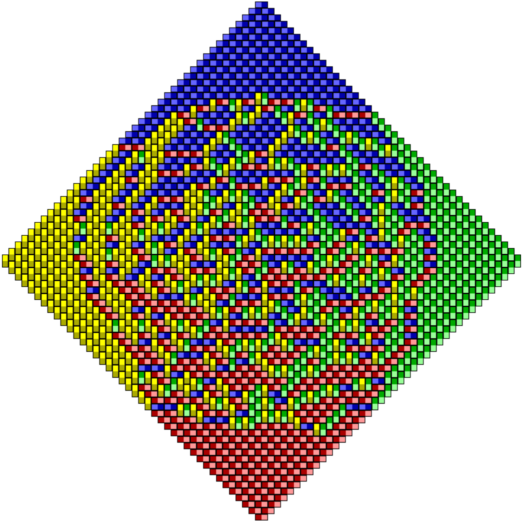
Complex non-Hermitian kernel; Sample likelihoods: $\exp(-568.381)$



Aztec diamond: $d = 40$

```
$ ./aztec_diamond --diamond_size=40
```

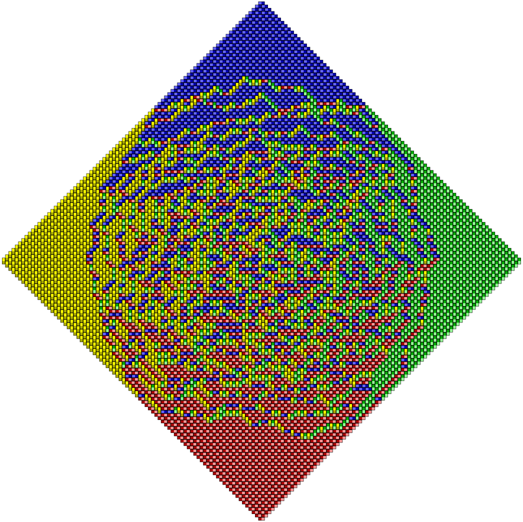
Complex non-Hermitian kernel; Sample likelihoods: $\exp(-568.381)$



Aztec diamond: $d = 80$

```
$ ./aztec_diamond --diamond_size=80
```

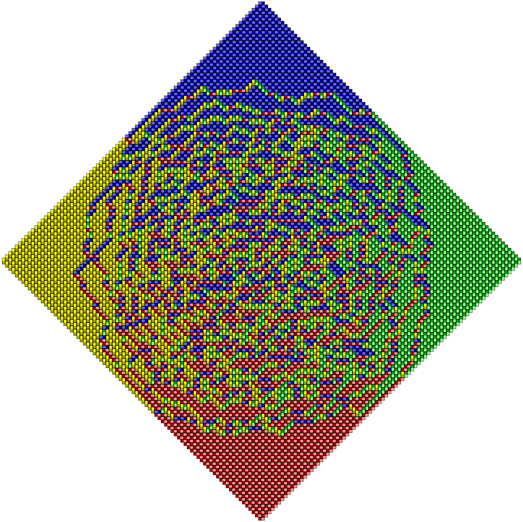
Complex non-Hermitian kernel; Sample likelihoods: $\exp(-2245.8)$



Aztec diamond: $d = 80$

```
$ ./aztec_diamond --diamond_size=80
```

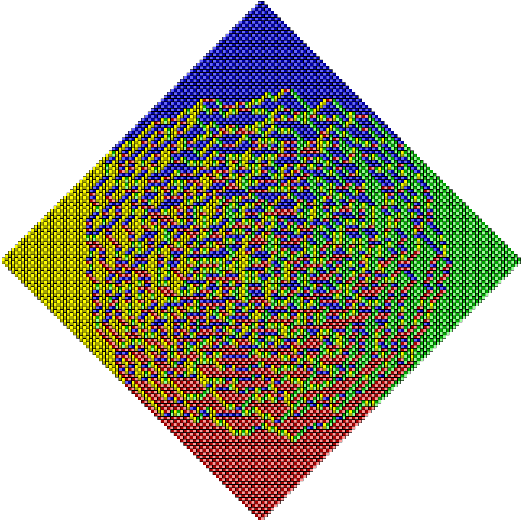
Complex non-Hermitian kernel; Sample likelihoods: $\exp(-2245.8)$



Aztec diamond: $d = 80$

```
$ ./aztec_diamond --diamond_size=80
```

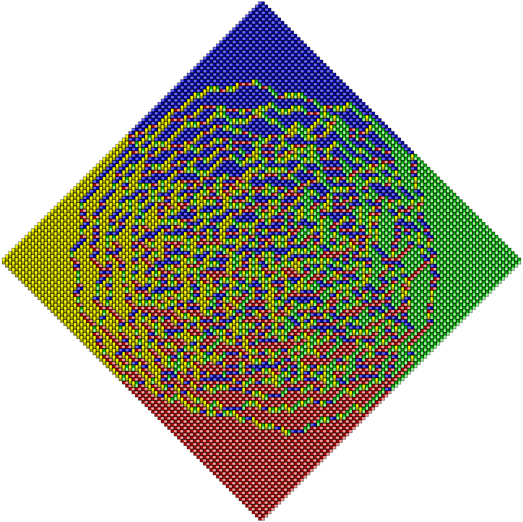
Complex non-Hermitian kernel; Sample likelihoods: $\exp(-2245.8)$



Aztec diamond: $d = 80$

```
$ ./aztec_diamond --diamond_size=80
```

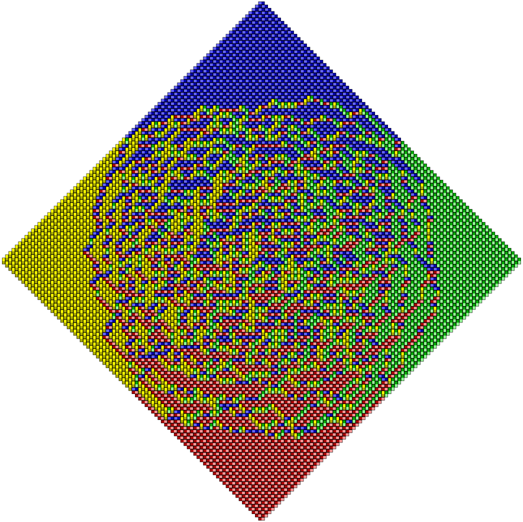
Complex non-Hermitian kernel; Sample likelihoods: $\exp(-2245.8)$



Aztec diamond: $d = 80$

```
$ ./aztec_diamond --diamond_size=80
```

Complex non-Hermitian kernel; Sample likelihoods: $\exp(-2245.8)$



The arctic circle

The phenomenon we just observed is referred to as the **arctic circle**. From [Jockusch/Propp/Shor-1995], *Random Domino Tilings and the Arctic Circle Theorem*:
"We show that when n is sufficiently large, the shape of the central sub-region becomes arbitrarily close to a perfect circle of radius $n/\sqrt{2}$ for all but a negligible proportion of the tilings."

On the other hand, square (non-diamond) tilings
"are statistically homogeneous unless one looks quite close to the boundary."

The arctic circle

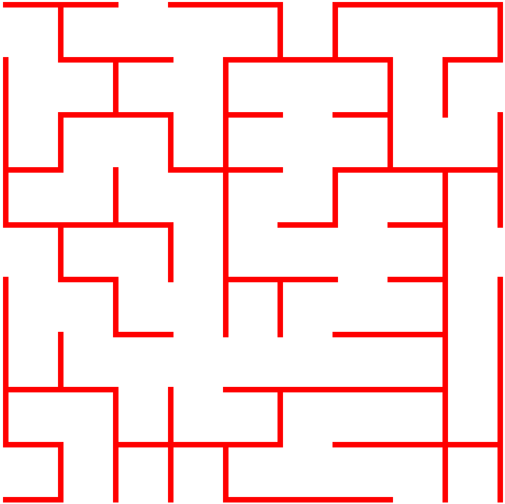
The phenomenon we just observed is referred to as the **arctic circle**. From [Jockusch/Propp/Shor-1995], *Random Domino Tilings and the Arctic Circle Theorem*:
"We show that when n is sufficiently large, the shape of the central sub-region becomes arbitrarily close to a perfect circle of radius $n/\sqrt{2}$ for all but a negligible proportion of the tilings."

On the other hand, square (non-diamond) tilings
"are statistically homogeneous unless one looks quite close to the boundary."

Uniform Spanning Tree in $\mathbb{Z}^2$ ($d = 10$)

```
$ ./uniform_spanning_tree --x_size=10 --y_size=
```

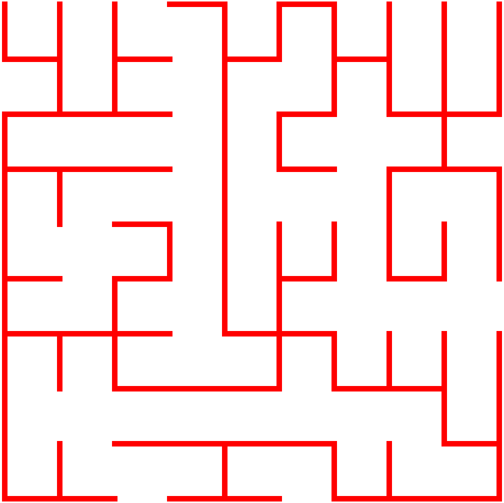
Real-symm' elementary kernel; Sample likelihoods: $\exp(-98.448)$



Uniform Spanning Tree in $\mathbb{Z}^2$ ($d = 10$)

```
$ ./uniform_spanning_tree --x_size=10 --y_size=
```

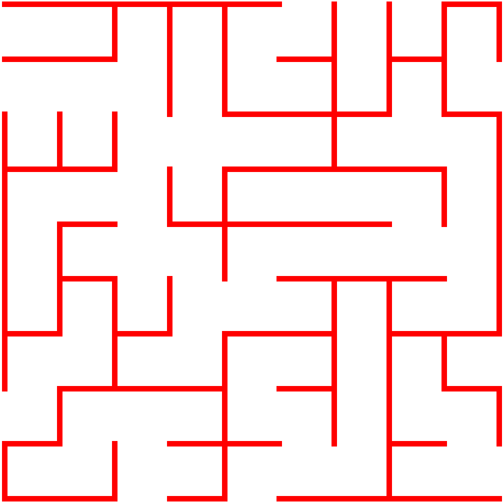
Real-symm' elementary kernel; Sample likelihoods: $\exp(-98.448)$



Uniform Spanning Tree in $\mathbb{Z}^2$ ($d = 10$)

```
$ ./uniform_spanning_tree --x_size=10 --y_size=
```

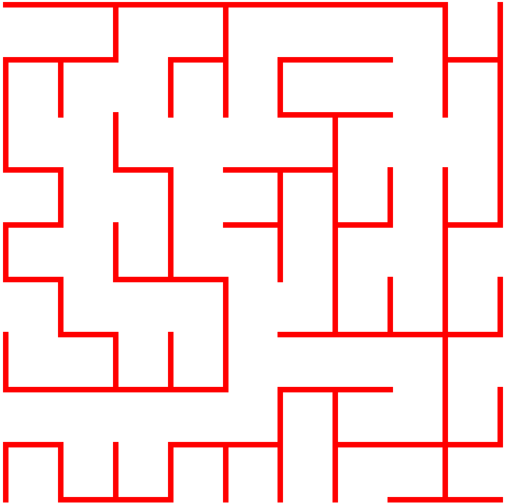
Real-symm' elementary kernel; Sample likelihoods: $\exp(-98.448)$



Uniform Spanning Tree in $\mathbb{Z}^2$ ($d = 10$)

```
$ ./uniform_spanning_tree --x_size=10 --y_size=
```

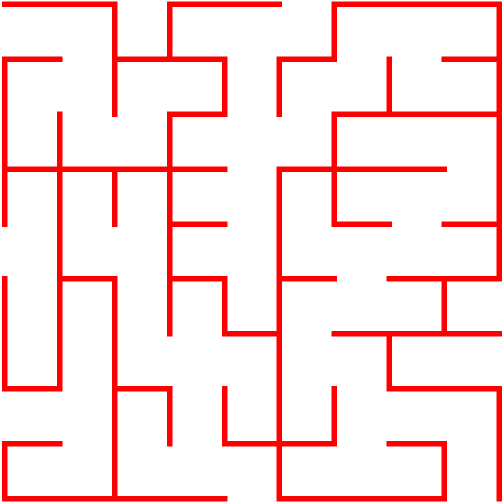
Real-symm' elementary kernel; Sample likelihoods: $\exp(-98.448)$



Uniform Spanning Tree in $\mathbb{Z}^2$ ($d = 10$)

```
$ ./uniform_spanning_tree --x_size=10 --y_size=
```

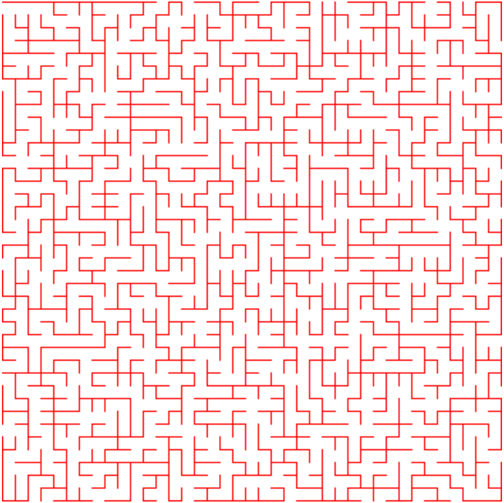
Real-symm' elementary kernel; Sample likelihoods: $\exp(-98.448)$



Uniform Spanning Tree in $\mathbb{Z}^2$ ($d = 40$)

```
$ ./uniform_spanning_tree --x_size=40 --y_size=
```

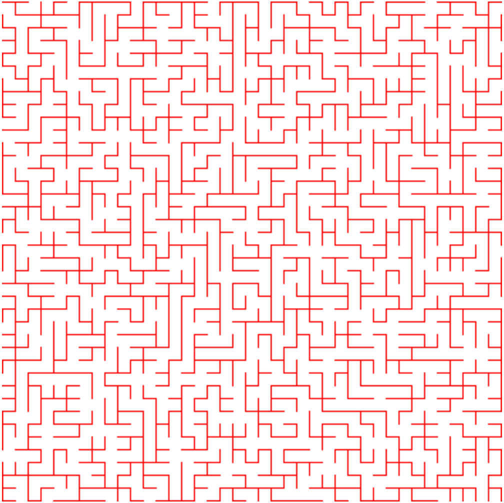
Real-symm' elementary kernel; Sample likelihoods: $\exp(-1794.24)$



Uniform Spanning Tree in $\mathbb{Z}^2$ ($d = 40$)

```
$ ./uniform_spanning_tree --x_size=40 --y_size=
```

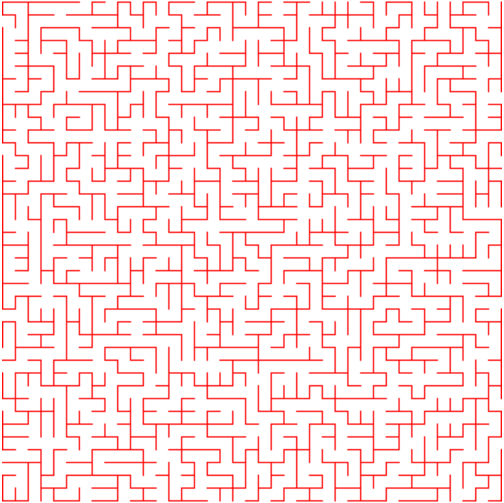
Real-symm' elementary kernel; Sample likelihoods: $\exp(-1794.24)$



Uniform Spanning Tree in $\mathbb{Z}^2$ ($d = 40$)

```
$ ./uniform_spanning_tree --x_size=40 --y_size=
```

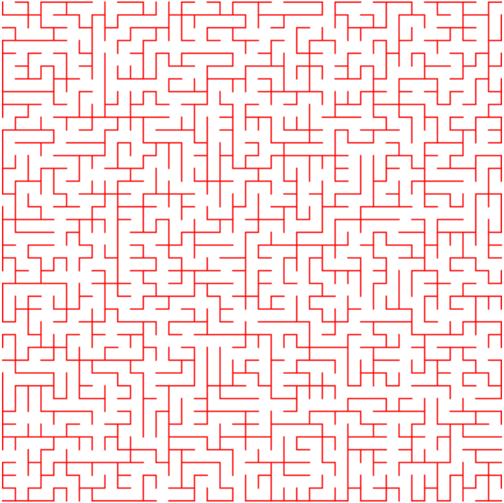
Real-symm' elementary kernel; Sample likelihoods: $\exp(-1794.24)$



Uniform Spanning Tree in $\mathbb{Z}^2$ ($d = 40$)

```
$ ./uniform_spanning_tree --x_size=40 --y_size=
```

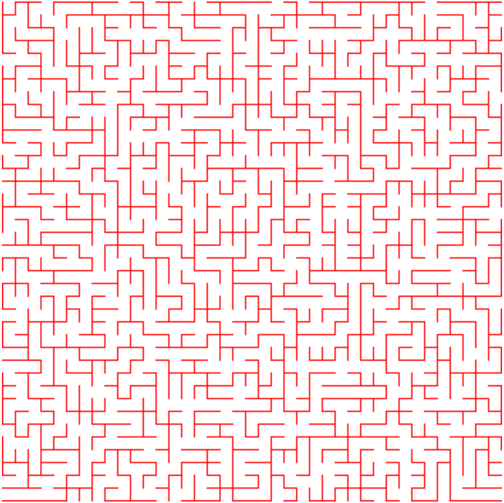
Real-symm' elementary kernel; Sample likelihoods: $\exp(-1794.24)$



Uniform Spanning Tree in $\mathbb{Z}^2$ ($d = 40$)

```
$ ./uniform_spanning_tree --x_size=40 --y_size=
```

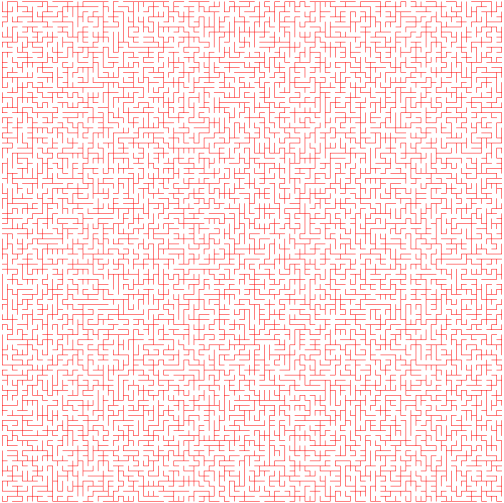
Real-symm' elementary kernel; Sample likelihoods: $\exp(-1794.24)$



Uniform Spanning Tree in $\mathbb{Z}^2$ ($d = 100$)

```
$ ./uniform_spanning_tree --x_size=100 --y_size
```

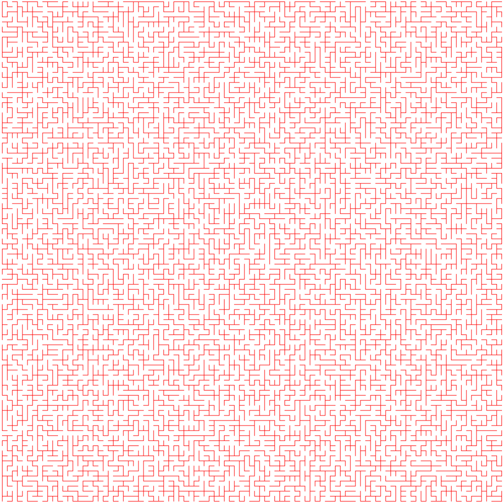
Real-symm' elementary kernel; Sample likelihoods: $\exp(-11,484.5)$



Uniform Spanning Tree in $\mathbb{Z}^2$ ($d = 100$)

```
$ ./uniform_spanning_tree --x_size=100 --y_size
```

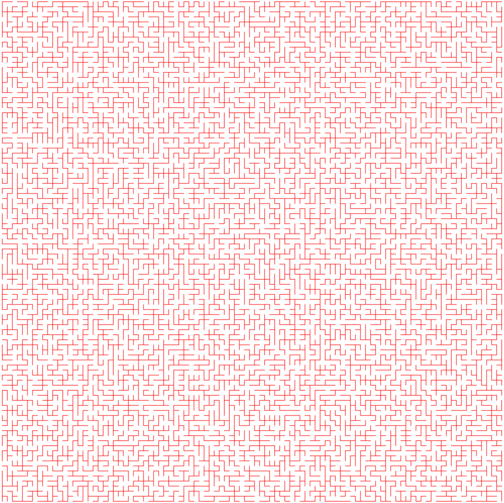
Real-symm' elementary kernel; Sample likelihoods: $\exp(-11, 484.5)$



Uniform Spanning Tree in $\mathbb{Z}^2$ ($d = 100$)

```
$ ./uniform_spanning_tree --x_size=100 --y_size
```

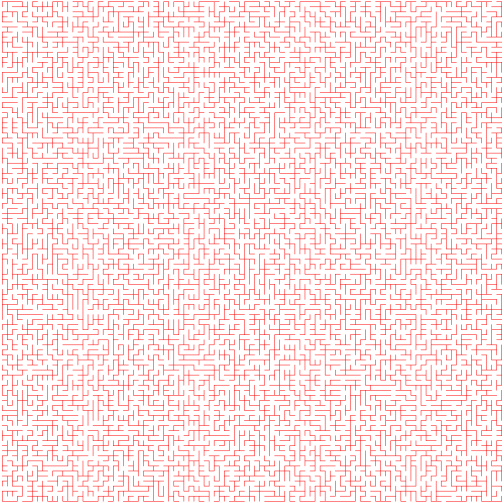
Real-symm' elementary kernel; Sample likelihoods: $\exp(-11, 484.5)$



Uniform Spanning Tree in $\mathbb{Z}^2$ ($d = 100$)

```
$ ./uniform_spanning_tree --x_size=100 --y_size
```

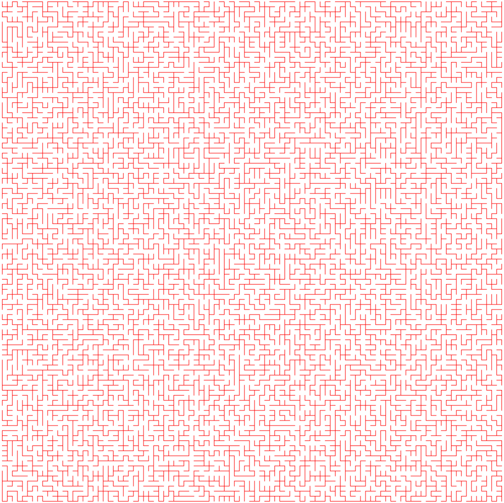
Real-symm' elementary kernel; Sample likelihoods: $\exp(-11,484.5)$



Uniform Spanning Tree in $\mathbb{Z}^2$ ($d = 100$)

```
$ ./uniform_spanning_tree --x_size=100 --y_size
```

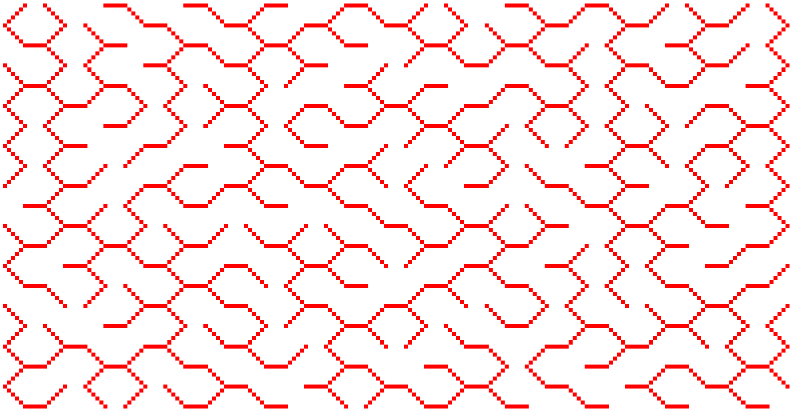
Real-symm' elementary kernel; Sample likelihoods: $\exp(-11,484.5)$



UST for hexagonal tiling of plane ($d = 10$)

```
$ ./uniform_spanning_tree --x_size=10 --y_size=10 --hexagonal=true
```

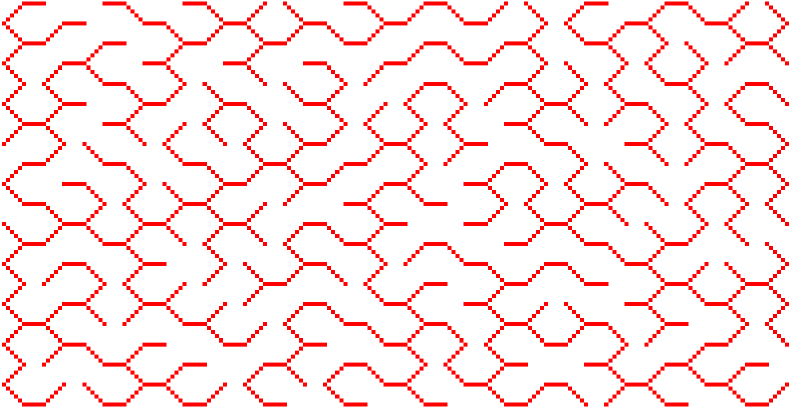
Real-symm' elementary kernel; Sample likelihoods: $\exp(-299.101)$



UST for hexagonal tiling of plane ($d = 10$)

```
$ ./uniform_spanning_tree --x_size=10 --y_size=10 --hexagonal=true
```

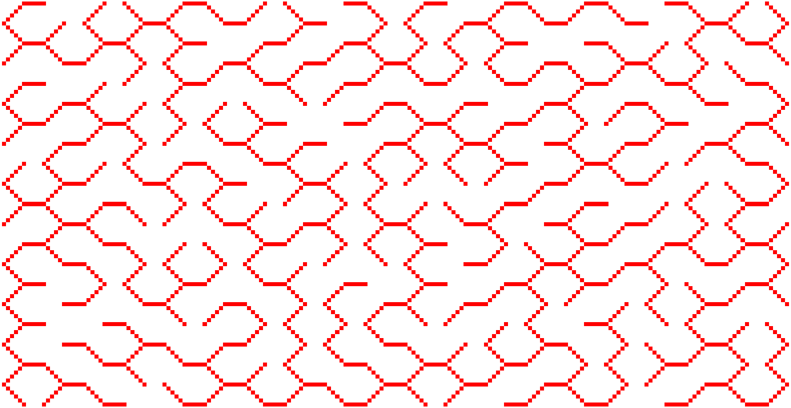
Real-symm' elementary kernel; Sample likelihoods: $\exp(-299.101)$



UST for hexagonal tiling of plane ($d = 10$)

```
$ ./uniform_spanning_tree --x_size=10 --y_size=10 --hexagonal=true
```

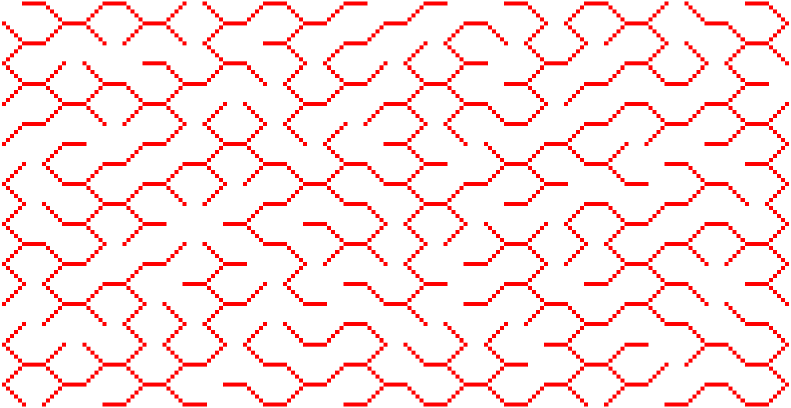
Real-symm' elementary kernel; Sample likelihoods: $\exp(-299.101)$



UST for hexagonal tiling of plane ($d = 10$)

```
$ ./uniform_spanning_tree --x_size=10 --y_size=10 --hexagonal=true
```

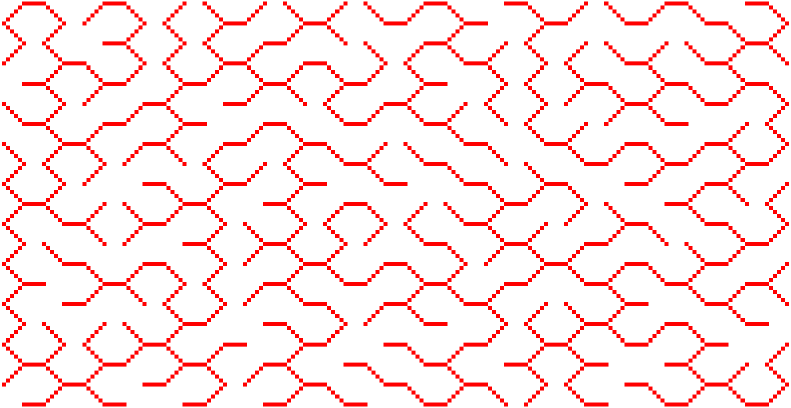
Real-symm' elementary kernel; Sample likelihoods: $\exp(-299.101)$



UST for hexagonal tiling of plane ($d = 10$)

```
$ ./uniform_spanning_tree --x_size=10 --y_size=10 --hexagonal=true
```

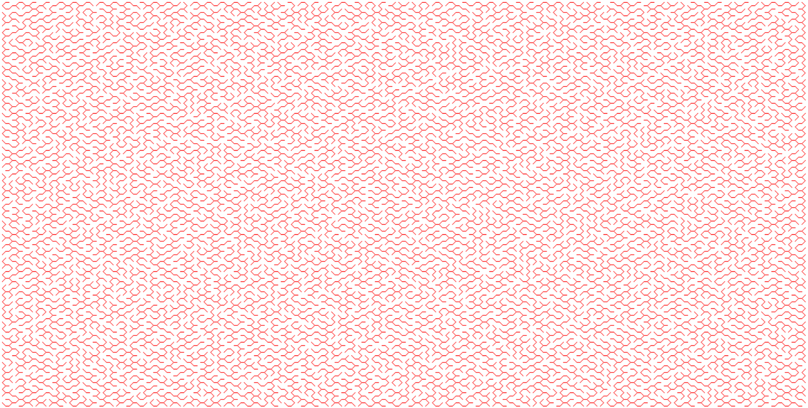
Real-symm' elementary kernel; Sample likelihoods: $\exp(-299.101)$



UST for hexagonal tiling of plane ($d = 60$)

```
$ ./uniform_spanning_tree --x_size=60 --y_size=60 --hexagonal=true
```

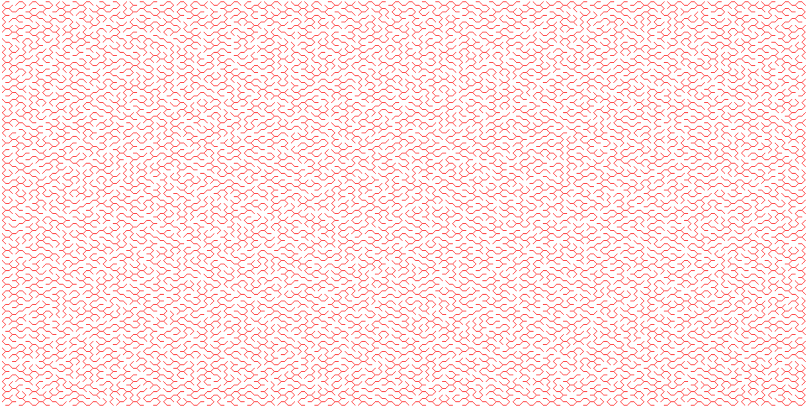
Real-symm' elementary kernel; Sample likelihoods: $\exp(-11,486.8)$



UST for hexagonal tiling of plane ($d = 60$)

```
$ ./uniform_spanning_tree --x_size=60 --y_size=60 --hexagonal=true
```

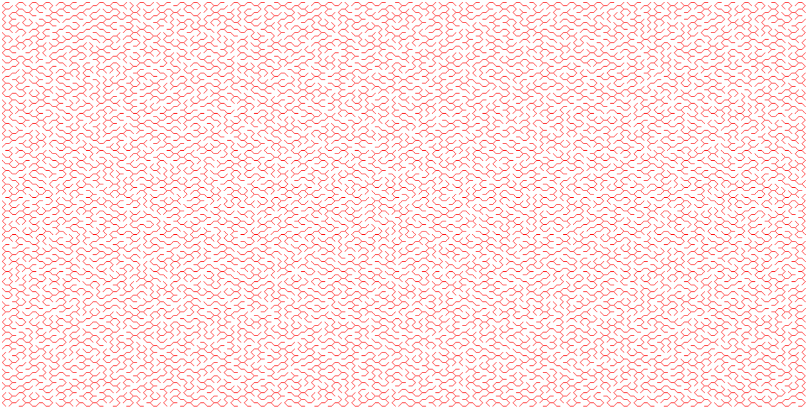
Real-symm' elementary kernel; Sample likelihoods: $\exp(-11,486.8)$



UST for hexagonal tiling of plane ($d = 60$)

```
$ ./uniform_spanning_tree --x_size=60 --y_size=60 --hexagonal=true
```

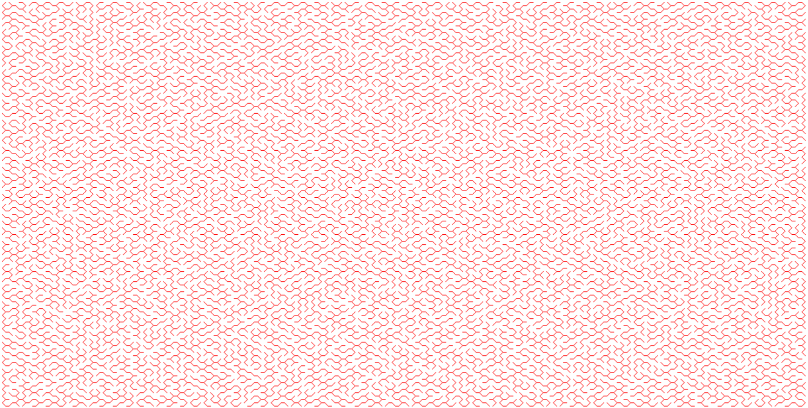
Real-symm' elementary kernel; Sample likelihoods: $\exp(-11,486.8)$



UST for hexagonal tiling of plane ($d = 60$)

```
$ ./uniform_spanning_tree --x_size=60 --y_size=60 --hexagonal=true
```

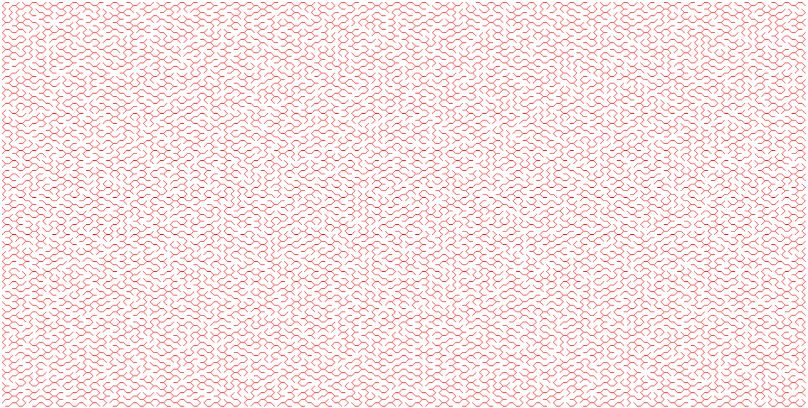
Real-symm' elementary kernel; Sample likelihoods: $\exp(-11,486.8)$



UST for hexagonal tiling of plane ($d = 60$)

```
$ ./uniform_spanning_tree --x_size=60 --y_size=60 --hexagonal=true
```

Real-symm' elementary kernel; Sample likelihoods: $\exp(-11,486.8)$



Hermitian Determinantal Point Processes

Definition 1. A **(Hermitian) marginal kernel matrix** is a (real or complex) Hermitian matrix whose eigenvalues live in $[0, 1]$.

Definition 2. A **(finite, Hermitian) Determinantal Point Process (DPP)** is a random variable $\mathbf{Y}$ over the power set of $\mathcal{Y} = \{0, ..., n-1\} = [n]$ generated by a $n \times n$ (Hermitian) marginal kernel matrix K via the rule

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

where K_Y is the $|Y| \times |Y|$ submatrix of K formed by restricting to the rows and columns in the index set Y.

Definition 3. A (Hermitian) DPP is called **elementary** if the eigenvalues of its marginal kernel matrix all lie in $\{0, 1\}$.

Hermitian Determinantal Point Processes

Definition 1. A **(Hermitian) marginal kernel matrix** is a (real or complex) Hermitian matrix whose eigenvalues live in $[0, 1]$.

Definition 2. A **(finite, Hermitian) Determinantal Point Process (DPP)** is a random variable $\mathbf{Y}$ over the power set of $\mathcal{Y} = \{0, ..., n-1\} = [n]$ generated by a $n \times n$ (Hermitian) marginal kernel matrix K via the rule

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

where K_Y is the $|Y| \times |Y|$ submatrix of K formed by restricting to the rows and columns in the index set Y.

Definition 3. A (Hermitian) DPP is called **elementary** if the eigenvalues of its marginal kernel matrix all lie in $\{0, 1\}$.

Hermitian Determinantal Point Processes

Definition 1. A **(Hermitian) marginal kernel matrix** is a (real or complex) Hermitian matrix whose eigenvalues live in $[0, 1]$.

Definition 2. A **(finite, Hermitian) Determinantal Point Process (DPP)** is a random variable $\mathbf{Y}$ over the power set of $\mathcal{Y} = \{0, ..., n-1\} = [n]$ generated by a $n \times n$ (Hermitian) marginal kernel matrix K via the rule

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

where K_Y is the $|Y| \times |Y|$ submatrix of K formed by restricting to the rows and columns in the index set Y.

Definition 3. A (Hermitian) DPP is called **elementary** if the eigenvalues of its marginal kernel matrix all lie in $\{0, 1\}$.

Star space

Generating the symmetric positive semi-definite DPP kernel matrices for uniformly sampling spanning trees is accomplished using an orthonormal basis of a vector space associated with an arbitrary ordering of a graph called the **star space** [Lyons/Peres-2017], *Probability on Trees and Networks*, www.uni-due.de/ hm0110/book.pdf.

Star space

Given a graph $G = (V, E)$ and an edge orientation $\overrightarrow{E}$, we associate with each oriented edge $e = uv$ a **unit flow along e**,

$$\chi_e(e') = [1_{uv} - 1_{vu}](e') = \begin{cases} 1, & e' = uv, \\ -1, & e' = vu, \\ 0, & \text{otherwise} \end{cases}.$$

We can now define the **star space** of the oriented graph as:

$$S = \operatorname{span}\left\{ \sum_{u:uv \in E} \chi_{uv} \,\middle|\, v \in V \right\},$$

and its orthogonal complement, the **cycle space**:

$$C = \operatorname{span}\left\{ \sum_{i=0}^{n-1} \chi_{e_i} \,\middle|\, e_0, \ldots, e_{n-1} \text{is an oriented cycle} \right\}.$$

Star space

Given a graph $G = (V, E)$ and an edge orientation $\overrightarrow{E}$, we associate with each oriented edge $e = uv$ a **unit flow along e**,

$$\chi_e(e') = [1_{uv} - 1_{vu}](e') = \begin{cases} 1, & e' = uv, \\ -1, & e' = vu, \\ 0, & \text{otherwise} \end{cases}.$$

We can now define the **star space** of the oriented graph as:

$$S = \operatorname{span}\left\{ \sum_{u:uv \in E} \chi_{uv} \mid v \in V \right\},$$

and its orthogonal complement, the **cycle space**:

$$C = \operatorname{span}\left\{ \sum_{i=0}^{n-1} \chi_{e_i} \mid e_0, \ldots, e_{n-1} \text{is an oriented cycle} \right\}.$$

Star space

Given a graph $G = (V, E)$ and an edge orientation $\overrightarrow{E}$, we associate with each oriented edge $e = uv$ a **unit flow along e**,

$$\chi_e(e') = [1_{uv} - 1_{vu}](e') = \begin{cases} 1, & e' = uv, \\ -1, & e' = vu, \\ 0, & \text{otherwise} \end{cases}.$$

We can now define the **star space** of the oriented graph as:

$$S = \operatorname{span}\left\{ \sum_{u:uv \in E} \chi_{uv} \mid v \in V \right\},$$

and its orthogonal complement, the **cycle space**:

$$C = \operatorname{span}\left\{ \sum_{i=0}^{n-1} \chi_{e_i} \mid e_0, \ldots, e_{n-1} \text{is an oriented cycle} \right\}.$$

Burton/Pemantle Theorem

We can now define the **transfer current matrix**, P, of an oriented graph G as the orthogonal projection onto its star space.

Theorem (Burton-Pemantle) Let T denote a uniformly random spanning tree of a graph G. Then for any subset of edges $F = \{e_0, ..., e_{k-1}\}$,

$$\mathbb{P}[F \subseteq T] = \det(P_F).$$

See [Lyons/Peres-2017] or [Burton/Pemantle-1993] for a proof.

Given a graph G, we can thus define the DPP kernel for uniformly sampling its spanning trees as the Gramian of an orthonormal basis for its star space.

Burton/Pemantle Theorem

We can now define the **transfer current matrix**, P, of an oriented graph G as the orthogonal projection onto its star space.

Theorem (Burton-Pemantle) Let T denote a uniformly random spanning tree of a graph G. Then for any subset of edges $F = \{e_0, ..., e_{k-1}\}$,

$$\mathbb{P}[F \subseteq T] = \det(P_F).$$

See [Lyons/Peres-2017] or [Burton/Pemantle-1993] for a proof.

Given a graph G, we can thus define the DPP kernel for uniformly sampling its spanning trees as the Gramian of an orthonormal basis for its star space.

Burton/Pemantle Theorem

We can now define the **transfer current matrix**, P, of an oriented graph G as the orthogonal projection onto its star space.

Theorem (Burton-Pemantle) Let T denote a uniformly random spanning tree of a graph G. Then for any subset of edges $F = \{e_0, ..., e_{k-1}\}$,

$$\mathbb{P}[F \subseteq T] = \det(P_F).$$

See [Lyons/Peres-2017] or [Burton/Pemantle-1993] for a proof.

Given a graph G, we can thus define the DPP kernel for uniformly sampling its spanning trees as the Gramian of an orthonormal basis for its star space.

Traditional Hermitian DPP sampling

Lemma 4 (Hough et al.-2006). Given any $\mathbf{Y} \sim \text{DPP}(K)$, where K has spectral decomposition $Q\Lambda Q^*$, sampling from $\mathbf{Y}$ is equivalent to sampling from the random elementary DPP with kernel $P(Q_{\mathbf{Z}})$, where $P(U) \equiv UU^*$ and $Q_{\mathbf{Z}}$ consists of the columns of Q with indices from $\mathbf{Z} \sim \text{DPP}(\Lambda)$.

"Alg. 1 runs in time $O(Nk^3)$, where k is the number of eigenvectors selected [...] the initial eigendecomposition of [K] 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."
[Kulesza/Taskar-2012]

[Gillenwater-2014] reduced the factored elementary DPP sampling down to cubic complexity via what is equivalent to diagonally-pivoted Cholesky.[1]

[1] [Gillenwater-2014] Approximate inference for determinantal point processes

Traditional Hermitian DPP sampling

Lemma 4 (Hough et al.-2006). Given any $\mathbf{Y} \sim \mathrm{DPP}(K)$, where K has spectral decomposition $Q\Lambda Q^*$, sampling from $\mathbf{Y}$ is equivalent to sampling from the random elementary DPP with kernel $P(Q_{\mathbf{Z}})$, where $P(U) \equiv UU^*$ and $Q_{\mathbf{Z}}$ consists of the columns of Q with indices from $\mathbf{Z} \sim \mathrm{DPP}(\Lambda)$.

"Alg. 1 runs in time $O(Nk^3)$, where k is the number of eigenvectors selected [...] the initial eigendecomposition of $[K]$ 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."
[Kulesza/Taskar-2012]

[Gillenwater-2014] reduced the factored elementary DPP sampling down to cubic complexity via what is equivalent to diagonally-pivoted Cholesky.[1]

[1] [Gillenwater-2014] Approximate inference for determinantal point processes

Traditional Hermitian DPP sampling

Lemma 4 (Hough et al.-2006). Given any $\mathbf{Y} \sim \text{DPP}(K)$, where K has spectral decomposition $Q\Lambda Q^*$, sampling from $\mathbf{Y}$ is equivalent to sampling from the random elementary DPP with kernel $P(Q_{\mathbf{Z}})$, where $P(U) \equiv UU^*$ and $Q_{\mathbf{Z}}$ consists of the columns of Q with indices from $\mathbf{Z} \sim \text{DPP}(\Lambda)$.

"Alg. 1 runs in time $O(Nk^3)$, where k is the number of eigenvectors selected [...] the initial eigendecomposition of $[K]$ 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."
[Kulesza/Taskar-2012]

[Gillenwater-2014] reduced the factored elementary DPP sampling down to cubic complexity via what is equivalent to diagonally-pivoted Cholesky.[1]

[1] [Gillenwater-2014] Approximate inference for determinantal point processes

Rank-revealing Cholesky factorization

Algorithm 1: Unblocked, left-looking, diagonally-pivoted, Cholesky. The computational cost is roughly $O(nk^2)$.

```
d := diag(A); orig_indices := [0:n]
k = 0
for j in range(n):
  # Sample pivot; perform permutations
  ++k
  Draw index t from [j : n] that maximizes d_t
  Perform Hermitian swap of indices j and t of A
  Swap positions j and t of orig_indices and d
  if d_j < tolerance:
    A_j := 0
    break
  A_j := sqrt(d_j)
  # Form new column; update 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]
```

Elementary DPP sampler

Algorithm 2: Unblocked, left-looking, diagonally-pivoted, Cholesky-based sampling of a Hermitian Determinantal Projection Process. Returned matrix will contain the in-place Cholesky factorization of K_Y. The computational cost is $O(nk^2)$.

```
A := K; d := diag(K); orig_indices := [0:n]
for j in range(k):
  # Sample pivot; perform 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 new column; update 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]
```

Non-Hermitian DPP kernels

Definition 5. A (finite) Determinantal Point Process is a random variable $\mathbf{Y}$ over the power set of $\mathcal{Y} = [n]$ generated by an **admissible** $K \in \mathbb{C}^{n \times n}$ that is consistent with the rule:

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

Proposition 1 (Brunel-2018) A matrix $K \in \mathbb{C}^{n \times n}$ is admissible as a DPP marginal kernel iff

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

Non-Hermitian DPP kernels

Definition 5. A (finite) Determinantal Point Process is a random variable $\mathbf{Y}$ over the power set of $\mathcal{Y} = [n]$ generated by an **admissible** $K \in \mathbb{C}^{n \times n}$ that is consistent with the rule:

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

Proposition 1 (Brunel-2018) A matrix $K \in \mathbb{C}^{n \times n}$ is admissible as a DPP marginal kernel iff

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

Equivalence classes of DPP kernels

Proposition 2 (P-2019) The equivalence class of a structurally symmetric DPP kernel $K \in \mathbb{C}^{n \times n}$ is its orbit under the group of diagonal similarity transformations, i.e.,

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

For complex Hermitian and real symmetric K, the entries of D must respectively lie in $U(1)$ and $O(1)$.

Proposition 3 (P-2019) The equivalence class of a structurally nonsymmetric DPP kernel K strictly contains its orbit under the group of diagonal similarity transformations.

Proof.

If structural symmetry is broken at a 2×2 submatrix, we need only observe that:

$$\mathrm{DPP}(\begin{pmatrix} \alpha & 0 \\ \beta & \gamma \end{pmatrix}) = \mathrm{DPP}(\begin{pmatrix} \alpha & 0 \\ 0 & \gamma \end{pmatrix}),$$

but neither is contained in the orbit of the other. ∎

Equivalence classes of DPP kernels

Proposition 2 (P-2019) The equivalence class of a structurally symmetric DPP kernel $K \in \mathbb{C}^{n\times n}$ is its orbit under the group of diagonal similarity transformations, i.e.,

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

For complex Hermitian and real symmetric K, the entries of D must respectively lie in $U(1)$ and $O(1)$.

Proposition 3 (P-2019) The equivalence class of a structurally nonsymmetric DPP kernel K strictly contains its orbit under the group of diagonal similarity transformations.

Proof.

If structural symmetry is broken at a 2×2 submatrix, we need only observe that:

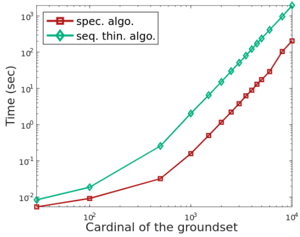
$$\mathrm{DPP}(\begin{pmatrix} \alpha & 0 \\ \beta & \gamma \end{pmatrix}) \equiv \mathrm{DPP}(\begin{pmatrix} \alpha & 0 \\ 0 & \gamma \end{pmatrix}),$$

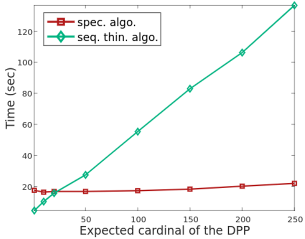
but neither is contained in the orbit of the other. □

Sequential thinning

Recently, authors are noticing connections to LDL^H factorizations.[2][3]

In [Launay et al.-2018], timings are provided for the spectrally-preprocessed and "sequentially thinned" algorithm for elementary real symmetric kernels of rank 20 and varying size (left) and varying rank and size 5000 (right):





We will discuss how to decrease runtimes by 100-1000x, for more general kernels, by importing dense factorization techniques. We then extend to non-Hermitian and, in the next lecture, sparse-direct analogues.

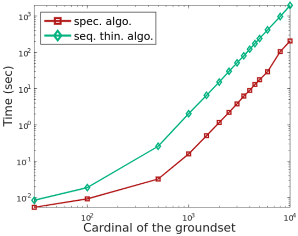
[2] [Chen et al.-2017] Fast Greedy MAP inference for Det' Point Processes

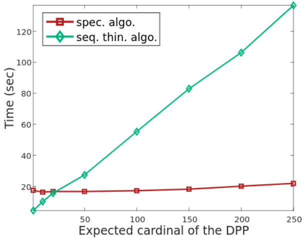
[3] [Launay et al.-2018] Exact sampling of determinantal point processes without eigendecomposition. arxiv.org/abs/1802.08429v3

Sequential thinning

Recently, authors are noticing connections to LDL^H factorizations.[2,3]

In [Launay et al.-2018], timings are provided for the spectrally-preprocessed and "sequentially thinned" algorithm for elementary real symmetric kernels of rank 20 and varying size (left) and varying rank and size 5000 (right):





We will discuss how to decrease runtimes by 100-1000x, for more general kernels, by importing dense factorization techniques. We then extend to non-Hermitian and, in the next lecture, sparse-direct analogues.

[2] [Chen et al.-2017] Fast Greedy MAP inference for Det' Point Processes

[3] [Launay et al.-2018] Exact sampling of determinantal point processes without eigendecomposition. arxiv.org/abs/1802.08429v3

Conditioning on inclusion

Proposition 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:

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

□

Conditioning on inclusion

Proposition 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:

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

□

Conditioning on inclusion

Proposition 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:

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

□

Conditioning on inclusion

Proposition 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:

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

□

Conditioning on element exclusion

Proposition 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. □

Conditioning on element exclusion

Proposition 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. □

Conditioning on element exclusion

Proposition 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. □

Conditioning on element exclusion

Proposition 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. □

Conditioning on element exclusion

Proposition 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. □

Conditioning on set exclusion

The previous two propositions are enough to derive our direct DPP sampling algorithm. But, for the sake of symmetry:

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

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

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. □

Conditioning on set exclusion

The previous two propositions are enough to derive our direct DPP sampling algorithm. But, for the sake of symmetry:
Proposition Given disjoint subsets $A, B \subset \mathcal{Y}$, almost surely

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

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. □

Unblocked LU factorization

Algorithm 3: Unblocked, right-looking, LU factorization.

for j in range(n):
  $A_{[j+1:n],\,j} \mathrel{/}= A_j$
  $A_{[j+1:n]} \mathrel{-}= A_{[j+1:n],\,j}\, A_{j,\,[j+1:n]}$
return A

Due to the lack of pivoting, completion is not guaranteed over $\mathrm{GL}_n(\mathbb{R})$ or $\mathrm{GL}_n(\mathbb{C})$.

The majority of the work is in rank-1 updates.

Unblocked DPP sampling factorization

Algorithm 4: Unblocked, right-looking, non-Hermitian DPP sampling. Returned matrix A contains in-place LU factorization of $K - I_{Y^c}$, where I_{Y^c} is diagonal indicator for entries not in sample.

```
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
```

Small tweak of unblocked, unpivoted LU factorization – completes almost surely. Specializable to LDL^H and LDL^T for Hermitian and complex symmetric matrices.

The majority of the work is in rank-1 updates. And the standard optimizations apply (e.g., blocking and sparse-direct factorization)!

The likelihood of the sample is equal to the product of the absolute value of the diagonal of the result.

Factorization-based DPP sampling

Theorem (Factorization-based DPP sampling) Given a marginal kernel K of order n, the unblocked DPP factorization almost surely provides a sample from DPP(K). And the likelihood of any returned sample will be given by the product of the absolute value of the diagonal of the result.

Proof.

We show the sampling claim by induction on the loop invariant that, at the start of iteration j, $A_{[j:n]}$ represents equivalence class for the DPP over indices $[j:n]$ conditioned on the inclusion decisions for indices $0, ..., j-1$.

Since the diagonal entries represent the likelihoods of the corresponding index being in the sample, the invariant implies that index j is kept with the correct conditional probability. Our conditional inclusion prop'n shows that the loop invariant is almost surely maintained when the Bernoulli draw is successful, and our conditional element exclusion prop'n handles the alternative.

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. □

Factorization-based DPP sampling

Theorem (Factorization-based DPP sampling) Given a marginal kernel K of order n, the unblocked DPP factorization almost surely provides a sample from $\text{DPP}(K)$. And the likelihood of any returned sample will be given by the product of the absolute value of the diagonal of the result.

Proof.

We show the sampling claim by induction on the loop invariant that, at the start of iteration j, $A_{[j:n]}$ represents equivalence class for the DPP over indices $[j:n]$ conditioned on the inclusion decisions for indices $0, ..., j-1$.

Since the diagonal entries represent the likelihoods of the corresponding index being in the sample, the invariant implies that index j is kept with the correct conditional probability. Our conditional inclusion prop'n shows that the loop invariant is almost surely maintained when the Bernoulli draw is successful, and our conditional element exclusion prop'n handles the alternative.

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. □

Factorization-based DPP sampling

Theorem (Factorization-based DPP sampling) Given a marginal kernel K of order n, the unblocked DPP factorization almost surely provides a sample from DPP(K). And the likelihood of any returned sample will be given by the product of the absolute value of the diagonal of the result.

Proof.

We show the sampling claim by induction on the loop invariant that, at the start of iteration j, $A_{[j:n]}$ represents equivalence class for the DPP over indices $[j:n]$ conditioned on the inclusion decisions for indices $0, ..., j-1$.

Since the diagonal entries represent the likelihoods of the corresponding index being in the sample, the invariant implies that index j is kept with the correct conditional probability. Our conditional inclusion prop'n shows that the loop invariant is almost surely maintained when the Bernoulli draw is successful, and our conditional element exclusion prop'n handles the alternative.

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. □

Factorization-based DPP sampling

Theorem (Factorization-based DPP sampling) Given a marginal kernel K of order n, the unblocked DPP factorization almost surely provides a sample from DPP(K). And the likelihood of any returned sample will be given by the product of the absolute value of the diagonal of the result.

Proof.

We show the sampling claim by induction on the loop invariant that, at the start of iteration j, $A_{[j:n]}$ represents equivalence class for the DPP over indices $[j:n]$ conditioned on the inclusion decisions for indices $0, ..., j-1$.

Since the diagonal entries represent the likelihoods of the corresponding index being in the sample, the invariant implies that index j is kept with the correct conditional probability. Our conditional inclusion prop'n shows that the loop invariant is almost surely maintained when the Bernoulli draw is successful, and our conditional element exclusion prop'n handles the alternative.

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. □

Unblocked, greedy, MAP DPP sampling

Algorithm 5: Unblocked, right-looking, non-Hermitian, greedy maximum-likelihood DPP sampling. Returned matrix A will contain in-place LU factorization of $K - I_{Y^C}$, where I_{Y^C} is diagonal indicator for entries not in sample.

```
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
```

Greedy MAP sampling is a trivial tweak of the standard sampler, and the blocked extension is essentially identical.

The likelihood of the sample is equal to the product of the absolute value of the diagonal of the result.

Blocked LU factorization

Algorithm 6: Blocked LU factorization without pivoting.

$j := 0$
while $j < n$:
  bsize $:= \min(\text{blocksize}, n - j)$
  $J_1 = [j : j + \text{bsize}]$; $J_2 = [j + \text{bsize} : n]$
  $A_{J_1} = \text{unblocked\_lu}(A_{J_1})$
  $A_{J_2, J_1} := A_{J_2, J_1}\, \text{triu}(A_{J_1})^{-1}$
  $A_{J_1, J_2} := \text{unit\_tril}(A_{J_1})^{-1}\, A_{J_1, J_2}$
  $A_{J_2} \mathrel{-}= A_{J_2, J_1}\, A_{J_1, J_2}$
  $j \mathrel{+}=$ bsize
return sample , A

Due to lack of pivoting, not guaranteed to complete over $\mathrm{GL}_n(\mathbb{R})$ or $\mathrm{GL}_n(\mathbb{C})$.

OpenMP 4.0 tasks – say, with tile sizes of 128 – can be readily used to provide shared-memory, DAG-scheduled parallelism [Agullo/Langou/Luszczek-2010, Yarkhan et al.-2011, Chan et al.-2007].

Blocked non-Hermitian DPP factorization

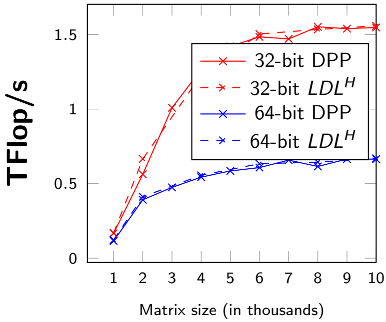
Algorithm 7: Returned matrix A will contain in-place LU factorization of $K - I_{Y^C}$, where I_{Y^C} is diagonal indicator for entries not in sample, $\mathbf{Y}$.

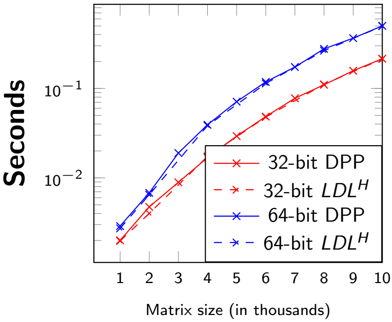
```
sample := []; A := K; j := 0
while j < n:
   bsize := min(blocksize, n − j)
   J_1 = [j : j + bsize]; J_2 = [j + bsize : n]
   subsample, A_{J_1} = unblocked_dpp(A_{J_1})
   sample.append(subsample + j)
   A_{J_2,J_1} := A_{J_2,J_1} triu(A_{J_1})^{-1}
   A_{J_1,J_2} := unit_tril(A_{J_1})^{-1} A_{J_1,J_2}
   A_{J_2} -= A_{J_2,J_1} A_{J_1,J_2}
   j += bsize
return sample, A
```

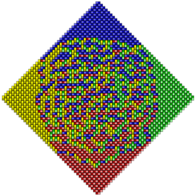
OpenMP 4.0 tasks – say, with tile sizes of 128 – can be readily used to provide shared-memory, DAG-scheduled parallelism [Agullo/Langou/Luszczek-2010, Yarkhan et al.-2011, Chan et al.-2007].

Dense, real LDL^H-based DPP sampler [P-2019].
For comparison, stock DPPy v0.1.0 [Gautier-2019] takes 250 seconds for each sample when $n = 5000$.

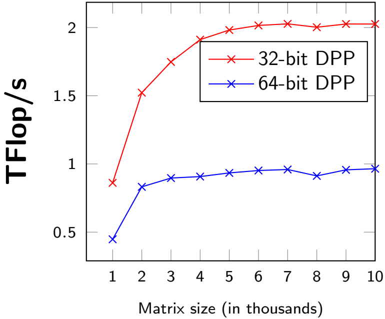
```
$ OMP_NUM_THREADS=16 ./dense_dpp
```

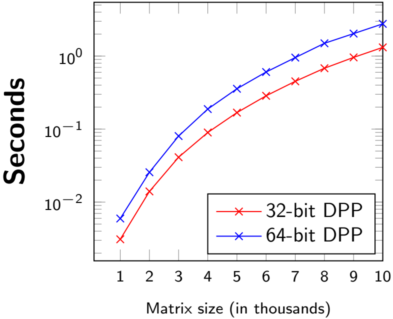






Dense, complex LU-based DPP sampler [P-2019].*

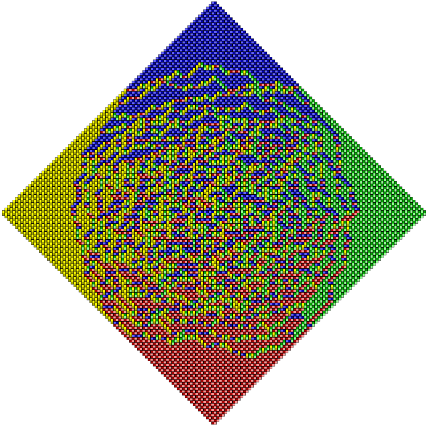




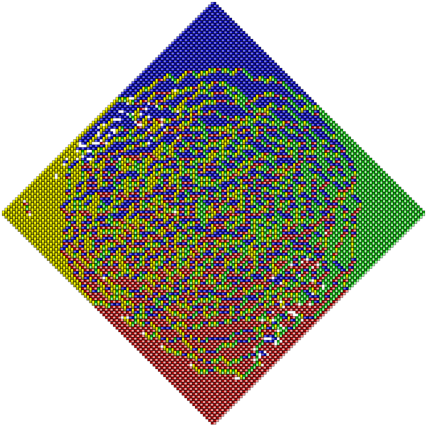
*E.g., generated from the Kenyon formula over the Kasteleyn matrix [Chhita et al.-2015].

Low precision corrupting sampling

```
$ ./aztec_diamond --diamond_size=80
```



Double-precision sample



Single-precision sample (visibly erroneous)

Basic questions for DPP factorizations

Given the close connection between DPP sampling and dense factorization:

- One should be able to probabilistically generalize element growth and numerical stability bounds.
- Use maximum-entropy diagonal pivot selection? Minimizes worst case pivot.
- High-performance techniques for backpropagating through Cholesky are now known [Murray-2016].[4] Do these blocked algorithms extend to DPPs?
- Extension to Permanental Point Process factorization? The 2×2 permanent

$$\text{perm}\left(\begin{pmatrix} \alpha_{0,0} & \alpha_{0,1} \\ \alpha_{1,0} & \alpha_{1,1} \end{pmatrix}\right) = \alpha_{0,0}\alpha_{1,1} + \alpha_{1,0}\alpha_{0,1},$$

  which suggests negating the sign of the Schur complement.

[4] [Murray-2016] Differentiation of the Cholesky decomposition. arxiv.org/abs/1602.07527

Basic questions for DPP factorizations

Given the close connection between DPP sampling and dense factorization:

- One should be able to probabilistically generalize element growth and numerical stability bounds.
- Use maximum-entropy diagonal pivot selection? Minimizes worst case pivot.
- High-performance techniques for backpropagating through Cholesky are now known [Murray-2016].[4] Do these blocked algorithms extend to DPPs?
- Extension to Permanental Point Process factorization? The 2×2 permanent

$$\text{perm}\left(\begin{pmatrix} \alpha_{0,0} & \alpha_{0,1} \\ \alpha_{1,0} & \alpha_{1,1} \end{pmatrix}\right) = \alpha_{0,0}\alpha_{1,1} + \alpha_{1,0}\alpha_{0,1},$$

  which suggests negating the sign of the Schur complement.

[4] [Murray-2016] Differentiation of the Cholesky decomposition. arxiv.org/abs/1602.07527

Basic questions for DPP factorizations

Given the close connection between DPP sampling and dense factorization:

- One should be able to probabilistically generalize element growth and numerical stability bounds.
- Use maximum-entropy diagonal pivot selection? Minimizes worst case pivot.
- High-performance techniques for backpropagating through Cholesky are now known [Murray-2016].[4] Do these blocked algorithms extend to DPPs?
- Extension to Permanental Point Process factorization? The 2×2 permanent

$$\text{perm}\left(\begin{pmatrix} \alpha_{0,0} & \alpha_{0,1} \\ \alpha_{1,0} & \alpha_{1,1} \end{pmatrix}\right) = \alpha_{0,0}\alpha_{1,1} + \alpha_{1,0}\alpha_{0,1},$$

which suggests negating the sign of the Schur complement.

[4] [Murray-2016] Differentiation of the Cholesky decomposition. arxiv.org/abs/1602.07527

Basic questions for DPP factorizations

Given the close connection between DPP sampling and dense factorization:

- One should be able to probabilistically generalize element growth and numerical stability bounds.
- Use maximum-entropy diagonal pivot selection? Minimizes worst case pivot.
- High-performance techniques for backpropagating through Cholesky are now known [Murray-2016].[4] Do these blocked algorithms extend to DPPs?
- Extension to Permanental Point Process factorization? The 2×2 permanent

$$\text{perm}\left(\begin{pmatrix} \alpha_{0,0} & \alpha_{0,1} \\ \alpha_{1,0} & \alpha_{1,1} \end{pmatrix}\right) = \alpha_{0,0}\alpha_{1,1} + \alpha_{1,0}\alpha_{0,1},$$

  which suggests negating the sign of the Schur complement.

[4] [Murray-2016] Differentiation of the Cholesky decomposition. arxiv.org/abs/1602.07527

More on Permanental Point Process factorization

If one works out the negated Schur complement for a 3×3 example, two second-order $\alpha_{0,0}^{-1}$ terms remain, rather than cancelling out as they do for the determinantal case.

One could handle the 3×3 case via **dual numbers**, with the second component introduced for the Schur complement updates.

But computing the permanent is $\sharp P$-hard; the determinant is polynomial because it is a homomorphism over $\mathrm{GL}_n(\mathbb{C})$.

More on Permanental Point Process factorization

If one works out the negated Schur complement for a 3×3 example, two second-order $\alpha_{0,0}^{-1}$ terms remain, rather than cancelling out as they do for the determinantal case.

One could handle the 3×3 case via **dual numbers**, with the second component introduced for the Schur complement updates.

But computing the permanent is $\sharp P$-hard; the determinant is polynomial because it is a homomorphism over $\mathrm{GL}_n(\mathbb{C})$.

More on Permanental Point Process factorization

If one works out the negated Schur complement for a 3×3 example, two second-order $\alpha_{0,0}^{-1}$ terms remain, rather than cancelling out as they do for the determinantal case.

One could handle the 3×3 case via **dual numbers**, with the second component introduced for the Schur complement updates.

But computing the permanent is $\sharp P$-hard; the determinant is polynomial because it is a homomorphism over $\mathrm{GL}_n(\mathbb{C})$.

Discussion

Availability:
Catamari is available under the Mozilla Public License 2.0 at
hodgestar.com/catamari/ and gitlab.com/hodge_star/catamari.
This talk is based on version 0.3.

These slides are available at:
hodgestar.com/G2S3/

Acknowledgements:

- Alex Kulesza and Jenny Gillenwater:
  For answering my initial DPP sampling questions.

Questions/comments?
Chatroom at:

`https://gitter.im/hodge_star/G2S3`