SAMPLING DETERMINANTAL POINT PROCESSES USING APPROXFUN

DANIEL PICKARD*

Abstract. Determinantal point processes (DPPs) occur in a wide variety of situations involving competing attraction and repulsion. In this term project, we utilize the ApproxFun Julia library to demonstrate how DPPs work in both theory and practice. We will show some samples taken from classic processes such as the gaussian unitary ensemble and jacobi ensembles, but will also demonstrate ApproxFun's capability to construct a wide variety of non-classical processes by specification of a reasonably smooth, positive function over an interal.

Key words. Determinantal Point Process, Random Matrix Ensembles

AMS subject classifications. 60E05, 60G55

1. Introduction. Determinantal point processes arise in the discrete setting when we assign a probability density to the elements of a power set. The power set is the collection of all subsets of a set including the original set and the nullset. In general, when we are sampling from the DPP we mean that we are going to get a single subset of a given set with the probability given by the DPP. To verify that we are correctly sampling the DPP we will have to sample from it many times of course. In this term project, I hope to show how we can construct arbitrary DPPs using ApproxFun, but I hope the reader might also appreciate the use of the accompanying Julia Notebook just as a visualization aid for the more standard DPPs of interest.

2. Some Basics of Projection DPPs. We know that all DPPs can be constructed out of projection DPPs (Elementary DPPs). In the discrete setting we think of a projection matrix $\mathbf{K}$ of some rank r, characterized by $\mathbf{Y}^T\mathbf{Y}$ where we have r different vectors in $\mathbf{Y}$. Here we have ensured all the eigenvalues are non-negative, which is important for defining probability spaces.

We can sample from this DPP, but we are certainly going to get a subset of 1 through N (where N is the size of the vectors) of size r. We can construct more general DPPs from these, but for this term project I will focus on these projection DPPs as all DPPs generated using ApproxFun are projection DPPs. Note that in general we cannot find a matrix $\mathbf{L}$ to use in the formula below

$$\mathbf{L}\begin{pmatrix}\tau\\ \tau\end{pmatrix}\det\left(\mathbf{I}+\mathbf{L}\right)^{-1} \tag{2.1}$$

However, we can still construct such a lower rank matrix $\mathbf{K}$ from some orthogonal $\mathbf{Y}$ and use it to define a DPP. As we have seen in the course notes, a random element of the powerset and a sample from a rank k projection DPP can be used to define more DPPs based on the intersection of these two sets (we could ask for a given set, and a given project DPP, what subset is going to be the intersection of the output and the given set).

3. Outline of Lanczos. For this project, we use ApproxFun's built in Lanczos function to generate some orthogonal polynomials. These polynomials $\pi_i(x)$ are orthogonal in the sense:

*Aerospace Department., Cambridge, MA (pickard@mit.edu).

$$\int \pi_i(x)\pi_j(x)w(x)dx = \delta_{ij} \tag{3.1}$$

Conveniently, we can select any arbitrary weighting $w(x)$ and integrate it over the interval of our choosing! We need to be careful not to select a weighting which is at any point negative such a weightin does not define a norm in the space of functions.

The main steps of Lanczos are summarized as the following:

$$\beta_j = |w_{j-1}| \tag{3.2}$$

$$v_j = \frac{w_{j-1}}{\beta} \tag{3.3}$$

$$\bar{w}_j = Av_j \tag{3.4}$$

$$\alpha_j = \bar{w}_j \cdot v_j \tag{3.5}$$

$$w_j = \bar{w}_j - \alpha_j v_j - \beta v_{j-1} \tag{3.6}$$

Here we use the typical three term recurrence relation to construct orthogonal functions. Internally inside ApproxFun these are represented as either Fourier or Chebyshev Polynomials, but for our purposes they are just general functions. We assemble these functions into a list of r functions $\mathbf{Y}$ which form a rank r projection kernel $\mathbf{K} = \mathbf{Y}\mathbf{Y}^T$ or a discrete identity operator due to their orthogonality relationships $\mathbf{I} = \mathbf{Y}^T\mathbf{Y}$. We sample from the projection DPP using these functions.

4. Sampling DPPs. To sample from a DPP we first construct the probability density function they define. As all the functions $\pi_i(x)$ can be integrated to get unity via: $\int \pi_i \pi_i w \partial x = 1$, the mean of the squares of these functions defines a probability density function which we sample from. We call our sample x_0

Having determined a point sampled from the DPP, we realize that we have now deterimined a rank $r-1$ determinantal point process. Specifically we can ask: what subset of size $r-1$ are we going to get from the remaining samples. We recall that the projection DPP of rank r always returns a subset of size r so we are sure that the remaining $r-1$ elements of that subset need to be assigned. Again, rank $r-1$ projection DPPs always return subsets of size $r-1$ so this is just another DPP hidden inside the first.

To construct this DPP we need to ensure that we have yet another probability density function that integrates to one. We also need this DPP to not ever include the point that we just sampled x_0. In the discrete setting this means that we cannot

have an element appear twice in a subset of the powerset. In the continuous setting, we need to ensure that the probability of selecting x_0 again is exactly zero. This is a curious little detail of this project that I have struggled with conceptually. $P(x_0) = 0$ already in the continuous setting because this is a set of measure zero. By that I mean that it is a point and the integral over a point is zero. Presumably what we really mean in the continuous setting is that the probability density of selecting x_0 is zero, not the probability itself. I am interested in better understanding the rigorous explanation of what follows from this conclusion which works well numerically. In any case, we need to make the density function hit zero at x_0 for the rank $r-1$ DPP that we get to sample the next point.

To perform this step my approach was to construct a discrete vector v_0 of size r from the function evaluations $\pi_i(x_0)$. Then I use the qr factorization to construct $r-1$ vectors which are orthogonal to v_0. Then I define some new orthogonal polynomials as follows

$$\pi_k = Q_{kj}\pi_j \tag{4.1}$$

Because all the rows and columns of the Q matrix are orthogonal to v_0, besides the original v_0, I will get $r-1$ functions which evaluate to zero at x_0. This is because the contraction of the Q with the functions evaluated at the sample is contracting the vector v_0 itself with the Q. Of course one polynomial π_k will evaluate to the norm of v squared at the previous sampling point, so to ensure the next PDF is zero at x_0 we drop this function. Note that multiplying orthogonal functions by this rotation type matrix preserves the orthogonality of the polynomials and the new π_k are also orthogonal! This means they determine a probability density function once again by taking the mean of their squares and integrating with $w(x)$. From here we restart the DPP algorithm. We can continue until we have a rank 0 DPP at which point we are done.

5. Verifying the DPP Sampling Algorithm. To verify this approach I histogram all the samples from the DPP and compare it to the curve I obtain when I plot the mean of the squares of the original function weighted by the $w(x)$. If the histogram of all the values (or of any particular entry of the unsorted output of the sampling for that matter) matches the original function then the DPP is working properly. If there is systematic drift away from this first curve (as I had for a long time while debugging) then we know that the second and third and fourth points are not being sampled correctly.

To check that lanczos is working properly I simply computed the identity matrix that I expect to be able to get using orthogonal functions. I include three sample DPP histograms which are produced with the Julia notebook. The first two are from Hermitian and Jacobi polynomials. The last one is orthogonal with respect to $\sin(x^4\pi)$ and I include it to show the general flexibility of this approach to generating DPP samples. Also found in my notebook, is a sequence of figures which I collected into a movie to demonstrate how the PDF evolves as the DPP is sampled. I found this movie to be very interesting and enjoyed working on this project. Thank you to Professor Edelman and Sung Woo Jeong for many helpful discussions and insights during the term.

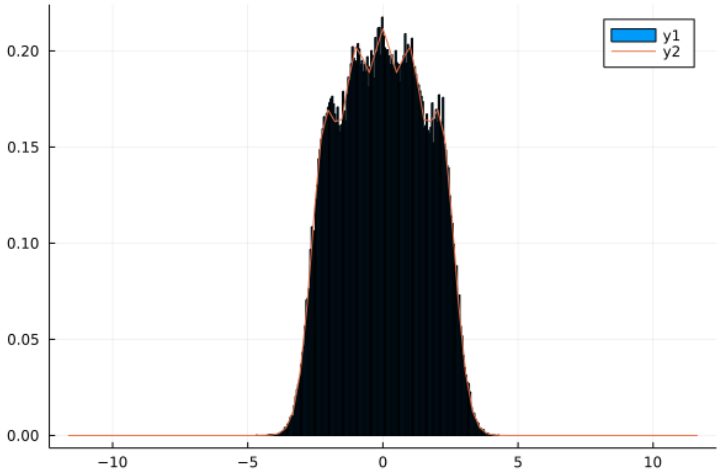

FIG. 1. *Rank 5 projection kernel for the gaussian unitary ensemble.*

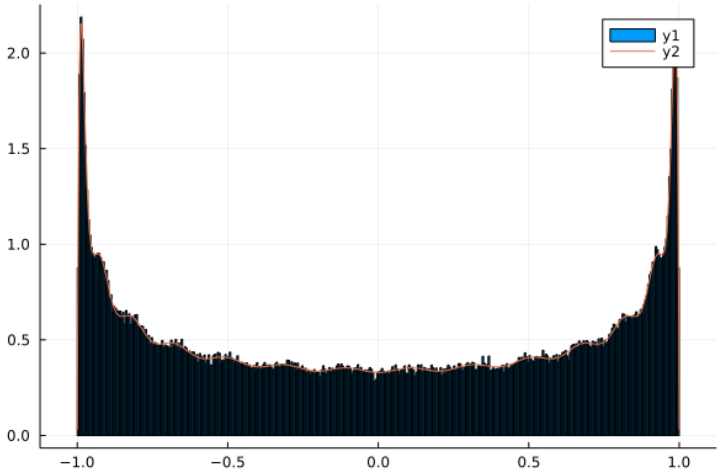

FIG. 2. *Rank 14 projection kernel for jacobi polynomials orthogonal with respect to* $x^2 - 1$. *ie* $\alpha = 1$ *and* $\beta = 1$.

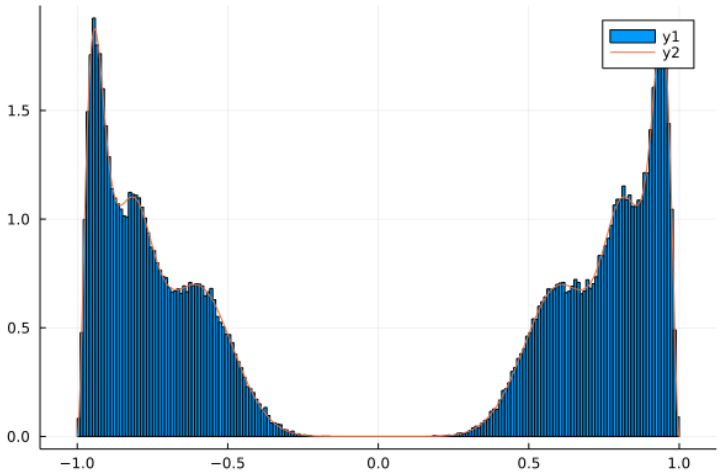

FIG. 3. *Rank 6 operator orthogonal with respect to* $\sin(x^4\ pi)$ *integrated over the interval -1 to 1.*