Numerical Experiments on Circular Ensembles and Jack Polynomials with Julia

N. Kemal Ure
Aerospace Controls Lab, MIT

May 15, 2013

Overview

- Brief background on circular ensembles and Jack polynomials
- How can we **efficiently** compute averages of Jack polynomials on circular ensembles ?
- Numerical Results
- Julia Language

Circular Ensembles

- Measures on space of unitary matrices (U such that $UU^* = I$)
  1. Circular Orthogonal Ensemble (COE), $\beta = 1$, symmetric unitary matrices
  2. Circular Unitary Ensemble (CUE), $\beta = 2$, unitary matrices
  3. Circular Symplectic Ensemble (CSE), $\beta == 4$, unitary quaternion matrices
- Eigenvalues are on the unit circle $\lambda_k = e^{i\theta_k}, k = 1, ..., n$.
- Joint Density is given by,

$$p(\theta) = \frac{1}{Z_{n,\beta}} \prod_{1 \le k < j \le n} |e^{i\theta_k} - e^{i\theta_j}|^{\beta}. \tag{1}$$

Jack Polynomials

- Multivariate polynomials parametrized by a parameter $\beta > 0$.
- Serve as a homogeneous base for space of k^{th} order symmetric polynomials in n variables.
- Jack Polynomials orthogonalize the β-circular ensembles ! [3]

$$\int_{[0,2\pi]^n} J_\kappa^\beta(e^{i\theta_1},...,e^{i\theta_n}), \overline{J_\lambda^\beta(e^{i\theta_1},...,e^{i\theta_n})} \prod_{j<k} |e^{i\theta_j} - e^{j\theta_k}|^\beta d\theta_1 ... d\theta_n = \delta_{\kappa,\lambda}.$$

The Experiment

- Jack polynomials with matrix arguments, $J_\kappa^\beta(U) = J_\kappa^\beta(\lambda_1, ..., \lambda_n)$, λ_k are the eigenvalues.
- We want to numerically verify that

$$\mathbb{E}[J_\kappa^\beta(U)\overline{J_\lambda^\beta(U)}] = 0, \qquad (2)$$

- U is a random unitary matrix, drawn according to $p(\theta)$
- We need two subroutines
  - Sample a random unitary matrix and find its eigenvalues
  - Evaluate the Jack function on these eigenvalues

Sampling Unitary Matrices

- Hessenberg matrices: "almost" upper triangular entries with all zero entries above below it's first superdiagonal.
- Example:

$$\begin{bmatrix} 0.5693 + 0.094i & -0.0042 + 0.0132i & 0.2468 + 0.7785i \\ 0.8168 & 0.0014 - 0.0097i & -0.2614 + 0.5153i \\ 0 & 0.9999 & 0.9999 \end{bmatrix}$$

- Ammar et. al [4] showed that there is a one to one correspondence between the Schur parameters $\gamma_j \in \mathbb{C}, j = 1, ..., n$ and $n \times n$ upper unitary Hessenberg matrices.
- Schur parameters: $|\gamma_j| \leq 1, 1 \leq j < n, |\gamma_n| = 1$.

$$H(\{\gamma\}) = G_1(\gamma_1)...G_{n-1}(\gamma_{n-1})\tilde{G_n}(\gamma_n),$$

- Hence Hessenberg matrices are parametrized by $2n - 1$ real numbers !

Evaluation of Jack Polynomials

- Jack polynomials can be expressed in the monomial basis,

$$J_\lambda^\beta = \sum_{T-SSYT} f_T(\beta) x^T,$$

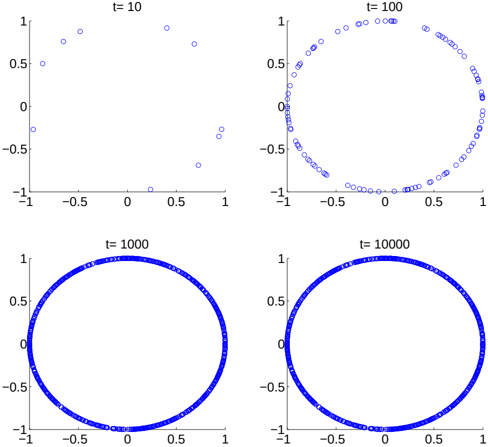
- Sum over SSYT: Semi-standard Young Tableaux. Numerically very inefficient.
- Demmel and Koev developed a recursive algorithm based on the principle of *dynamic programming*

$$J_\lambda^\beta(x_1, ..., x_n) = \sum_{\mu \le \lambda} J_\mu^\beta(x_1, ..., x_{n-1}) x_n^{|\lambda/\mu|} c_{\mu\lambda},$$

- The basic idea is to represent a polynomial in n variables in terms of polynomials in $(n-1)$ variables.
- MATLAB implementation available.

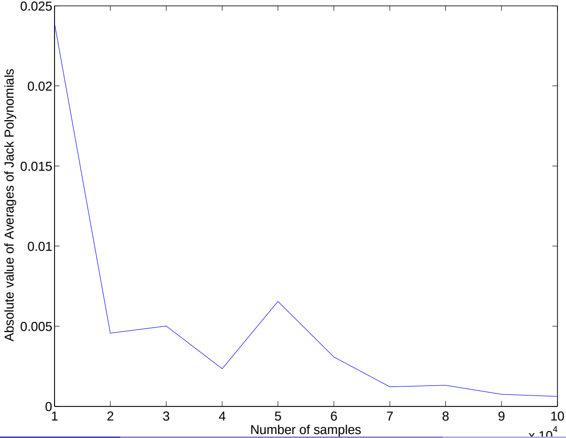
Sampling Unitary Hessenberg Matrices

- Generate random Schur parameters and then construct the unitary upper Hessenberg matrix (Implemented in Julia)
- Plots of eigenvalues for different sample sizes ($n = 3$):



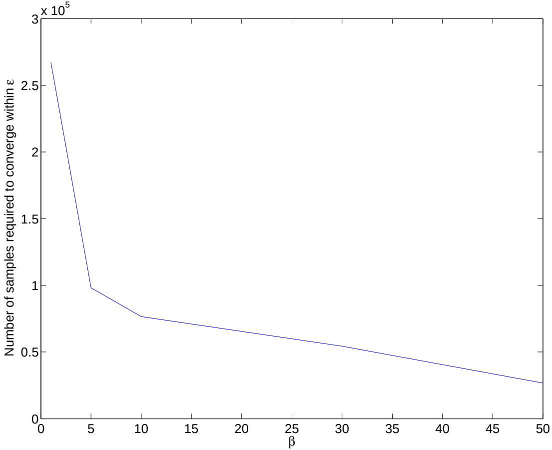
Averaging Jack Polynomials

- Koev's algorithm implemented on Julia and averaged over eigenvalues of random unitary matrices
- Results are also averaged over different partitions (also selected randomly), $\beta = 7$, $n = 3$.



Averaging Jack Polynomials

- Increasing β results in faster convergence
- Plot shows the number of samples required to converge within $\epsilon = 10^{-4}$ as a function of β.



Julia Language

- Scientific computing languages: Trade-off between **ease of coding** and **computational power**
- **Julia:** Easy to learn and code as in MATLAB and comparable to C in performance
  - It is also very easy to parallelize !
- Computing time (in seconds) across MATLAB, Julia and Parallel Julia (4 cores) for the averaging Jack polynomials on circular ensemble:

Number of Samples	**MATLAB**	**Julia**	**Parallel Julia**
500	5.1	1.7	1.3
1000	10.2	1.9	1.8
2000	20.3	2.4	1.9
5000	51.8	3.6	2.1
50000	516.4	25.6	3.9

- MATLAB users: convert your code to Julia !
- Users of other languages (C,C++, Python, Java): Discover Julia's parallel computing power.

References

[1] F.M. Dyson "The threefold way. Algebraic structure of symmetry groups and ensembles in quantum mechanics". J. Math. Phys. 3: 1199. (1962).

[2] Jack, Henry, "A class of symmetric polynomials with a parameter", Proceedings of the Royal Society of Edinburgh, Section A. Mathematics 69: 118, (1971).

[3] Macdonald, I. G., Symmetric functions and Hall polynomials, Oxford Mathematical Monographs (2nd ed.), New York: Oxford University Press, ISBN 0-19-853489-2, MR 1354144 (1995)

[4] Ammar, Gregory, William Gragg, and Lothar Reichel. "Constructing a unitary Hessenberg matrix from spectral data." In Numerical linear algebra, digital signal processing and parallel algorithms, pp. 385-395. Springer Berlin Heidelberg, 1991.

[5] Demmel, James, and Plamen Koev. "Accurate and efficient evaluation of Schur and Jack functions." Mathematics of computation 75, no. 253 (2006): 223-239.

[6] Forrester, Peter J., and Eric M. Rains. "Jacobians and rank 1 perturbations relating to unitary Hessenberg matrices." arXiv preprint math/0505552 (2005).