



# MOPS: Multivariate orthogonal polynomials (symbolically)

Ioana Dumitriu<sup>a,\*</sup>, Alan Edelman<sup>b</sup>, Gene Shuman<sup>c</sup>

<sup>a</sup> *Department of Mathematics, University of Washington, Seattle, WA 98195, United States*

<sup>b</sup> *Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA 02139, United States*

<sup>c</sup> *Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, United States*

Received 15 October 2005; accepted 11 January 2007

Available online 3 February 2007

---

## Abstract

In this paper we present a Maple library (MOPS) for computing Jack, Hermite, Laguerre, and Jacobi multivariate polynomials, as well as eigenvalue statistics for the Hermite, Laguerre, and Jacobi ensembles of random matrix theory. We also compute multivariate hypergeometric functions, and offer both symbolic and numerical evaluations for all these quantities.

We prove that all algorithms are well-defined, analyze their complexity, and illustrate their performance in practice. Finally, we present a few applications of this library.

© 2007 Elsevier Ltd. All rights reserved.

---

## 1. Introduction

### 1.1. Motivation

There is no need for us to review the impact that classical orthogonal polynomial and special functions theory has had for applications in mathematics, science, engineering and computations. By the middle of the last century, handbooks had been compiled that could be found on nearly everyone's bookshelf. In our time, handbooks join forces with mathematical software and new applications making the subject as relevant today as it was over a century ago.

---

\* Corresponding author. Tel.: +1 206 616 8164; fax: +1 206 543 0397.  
E-mail address: [dumitriu@math.washington.edu](mailto:dumitriu@math.washington.edu) (I. Dumitriu).

We believe that the modern day extensions of these scalar functions are the multivariate orthogonal polynomials or MOPS along with their special function counterparts.

The multivariate cases are far richer, yet at this time they are understudied, underapplied, and important applications may be missed. A few steps in the direction of writing and analyzing software for the computation of multivariate special functions have already been taken; for a fairly comprehensive list of software packages that are available, see Zabrocki's webpage (Zabrocki, 2007). However, at this time, there seem to be no systematic studies of such algorithms, no handbooks, and no collections of applications, though since April 2004 entries are being introduced into Eric Weinstein's Mathworld website.<sup>1</sup>

Development and analysis of such software may thus be seen as an area of research ripe for study. This paper is taking another step in this direction; undoubtedly better and more general software will emerge in time.

### 1.2. Contributions of this paper

We are presenting here a collection of algorithms for computing a number of multivariate orthogonal polynomials and special functions associated with Jack polynomial theory. Some formulas that form the basis for particular ( $\alpha = 1, 2$ ) cases of these algorithms appeared in papers (see James (1968, 1975)) and books (see Muirhead (1982)) before; some of the recurrences we mention here have been known for a while (see Kaneko (1993)); most of them, in their general form ( $\alpha > 0$ ), appear here for the first time.

To make the distinction between what is new and what was known before, we label every new result as a Lemma or Theorem, and all old results as Propositions.

As far as we know, there has been no systematic attempt to write code that computes multivariate Hermite, Laguerre, or Jacobi polynomials.

Regarding the Jack polynomials, several attempts at implementing computational formulas have been made; we mention here the Symmetric Functions Maple package (Stembridge, 2007) by John Stembridge (which implements an algorithm for computing Jack polynomials in elementary functions basis, and can convert the result to monomial basis) and the small Maple package LUC (Zeilberger, 2007) by Doron Zeilberger, which implements an explicit formula for the Jack polynomials (in monomial basis) found by Vinet and Lapointe (Lapointe and Vinet, 1995). We have found both of them to be slower than MOPS in calculating and expressing Jack polynomials in monomial basis. In the case of the former, this is partly because conversion to monomial basis is ulterior to the actual computation; in the case of the latter, the slow-down is due to the explicit (rather than recursive) character of the computation.

We also note that, while SF 2.4 is faster at *simultaneously computing all Jack polynomials* (expressed in elementary function basis) corresponding to a given partition size (up to partition size 21), MOPS can achieve much higher partition sizes (35 and higher) in computations of *individual Jack polynomials* (when expressed in monomial basis).

In addition to providing recurrence-based algorithms for computing multivariate orthogonal polynomials, we prove that these recurrences are well-defined. This is a non-trivial issue, since many of them involve division by polynomials in the recurrence parameters, which we prove to be non-zero.

Finally, we provide a complexity analysis for our codes, and provide sample running times.

---

<sup>1</sup> Mathworld, URL <http://mathworld.wolfram.com/>.

### 1.3. Orthogonal polynomials

We recall that scalar orthogonal polynomials are defined by a positive weight function  $w(x)$  defined on an interval  $I \subset \mathbb{R}$ . We define the inner product

$$\langle f, g \rangle_w = \int_I f(x)g(x)w(x)dx$$

and the sequence of polynomials  $p_0(x), p_1(x), p_2(x), \dots$ , such that  $p_k(x)$  has degree  $k$ , and such that  $\langle p_i, p_j \rangle_w = 0$  if  $i \neq j$ . This sequence is the sequence of orthogonal polynomials with respect to the weight function  $w(x)$ .

There is a (scalar) complex version of this inner product ( $I \subset \mathbb{C}$ ) where we use  $\bar{g}$  instead of  $g$ ; this induces a different set of orthogonal polynomials.

We now define the multivariate version of the inner product, and the corresponding orthogonal polynomials. We take any weight function  $w(x)$  defined on a segment  $I$ , and create an  $n$ -dimensional weight function which is symmetric in each of its  $n$  coordinates, and incorporates a repulsion factor which depends on a “Boltzmann” constant  $\beta$  (or a temperature factor  $\alpha = 2/\beta$ ) which is not seen in the univariate case:

$$W(x_1, \dots, x_n) = \prod_{1 \leq i < j \leq n} |x_i - x_j|^{2/\alpha} \prod_{i=1}^n w(x_i). \tag{1}$$

We define multivariate orthogonal polynomials  $p_\kappa^\alpha(x_1, \dots, x_n)$  with respect to the weight  $W(x_1, \dots, x_n)$ . The polynomials are symmetric: they take the same value for any permutation of the  $n$  coordinates  $x_i$ , and they satisfy

$$\int_{I^n} p_\kappa^\alpha(x_1, \dots, x_n) p_\mu^\alpha(x_1, \dots, x_n) \prod_{i < j} |x_i - x_j|^\beta \prod_{j=1}^n w(x_j) dx_1 \dots dx_n = \delta_{\kappa\mu},$$

where  $\kappa$  represents the “multivariate degrees” of  $p_\kappa^\alpha$  (the exponent of the leading term).

We begin with our fourth example: symmetric multivariate Hermite polynomials. We take  $w(x) = e^{-x^2/2}$ , so that the integral is over all of  $\mathbb{R}^n$ . The polynomials are denoted  $H_\kappa^\alpha(x)$ . Our second and third examples are  $w(x) = x^a e^{-x}$  and  $w(x) = x^{a_1} (1 - x)^{a_2}$ . These are the Laguerre  $L_\kappa^{\alpha,a}$  and Jacobi  $J_\kappa^{\alpha,a_1,a_2}$  polynomials. Special cases of the Jacobi polynomials are the Chebyshev and Legendre polynomials.

Our first example, the Jack polynomials, generalizes the monomial scalar functions,  $x^k$ . These polynomials are orthogonal on the unit circle:  $w = 1$  and  $I =$  the unit circle in the complex plane. Therefore  $I^n$  may be thought of as an  $n$ -dimensional torus. The orthogonality of the Jack polynomials may be found in formula (10.35) in Macdonald’s book (Macdonald, 1995, p. 383).

Tables 1–4 give the coefficients of the Jack, Hermite, Laguerre, and Jacobi polynomials in terms of the monomial symmetric functions (for the first) and the Jack polynomials (for the last three). We take all degrees up to total degree 4 for the Jack polynomials, up to total degree 3 for the Hermite polynomials, and up to degree 2 for Laguerre and Jacobi; the coefficients can be seen by a simple call to the procedures, e.g.,<sup>2</sup>

- > **jack**(a, [2], 'J');
- > **hermite**(a, [1, 1, 1], n, 'C');
- > **laguerre**(a, [1, 1], g, n, 'C');
- > **jacobi**(a, [1], g<sub>1</sub>, g<sub>2</sub>, n, 'C');

<sup>2</sup> Note the use of  $a$  for  $\alpha$  and  $g$  for  $\gamma$  in the calls.

Table 1  
Coefficients of the Jack “J” polynomial expressed in monomial basis

$k = 1$	$m_{[1]}$			$k = 2$	$m_{[2]}$	$m_{[1,1]}$
$J_{[1]}^\alpha$	1			$J_{[2]}^\alpha$	$1 + \alpha$	2
				$J_{[1,1]}^\alpha$	0	2
$k = 3$	$m_{[3]}$	$m_{[2,1]}$	$m_{[1,1,1]}$			
$J_{[3]}^\alpha$	$(1 + \alpha)(2 + \alpha)$	$3(1 + \alpha)$	6			
$J_{[2,1]}^\alpha$	0	$2 + \alpha$	6			
$J_{[1,1,1]}^\alpha$	0	0	6			
$k = 4$	$m_{[4]}$	$m_{[3,1]}$	$m_{[2,2]}$	$m_{[2,1,1]}$	$m_{[1,1,1,1]}$	
$J_{[4]}^\alpha$	$(1 + \alpha)(1 + 2\alpha)(1 + 3\alpha)$	$4(1 + \alpha)(1 + 2\alpha)$	$6(1 + \alpha)^2$	$12(1 + \alpha)$	24	
$J_{[3,1]}^\alpha$	0	$2(1 + \alpha)^2$	$4(1 + \alpha)$	$2(5 + 3\alpha)$	24	
$J_{[2,2]}^\alpha$	0	0	$2(2 + \alpha)(1 + \alpha)$	$4(2 + \alpha)$	24	
$J_{[2,1,1]}^\alpha$	0	0	0	$2(3 + \alpha)$	24	
$J_{[1,1,1,1]}^\alpha$	0	0	0	0	24	

1.4. History and connection to random matrix theory

The Jack polynomials have a very rich history. They represent a family of orthogonal polynomials dependent on a positive parameter  $\alpha$ , and some of them are more famous than others. There are three values of  $\alpha$  which have been studied independently, namely,  $\alpha = 2, 1, 1/2$ . The Jack polynomials corresponding to  $\alpha = 1$  are better known as the Schur functions; the  $\alpha = 2$  Jack polynomials are better known as the zonal polynomials, and the Jack polynomials corresponding to  $\alpha = 1/2$  are known as the quaternion zonal polynomials.

In an attempt to evaluate the integral (2) in connection with the non-central Wishart distribution, James (1960) discovered the zonal polynomials in 1960:

$$\int_{O(n)} (\text{tr}(AHBH^T))^k (H^T dH) = \sum_{\kappa \vdash k} c_\kappa Z_\kappa(A) Z_\kappa(B). \tag{2}$$

Inspired by the work of James (1960) and Hua (1963), in his own attempt to evaluate (2), Jack was lead to define the polynomials eventually associated with his name (Jack, 1970). More concretely, he discovered an explicit one-parameter ( $\alpha$ ) family of polynomials (which, for  $\alpha = 2$ , he called “pre-zonal”) and provided a procedure for obtaining from them the one-parameter family of polynomials now bearing his name. Jack proved that the new, constructively defined family of polynomials coincided with the Schur functions for  $\alpha = 1$ , and conjectured that they were the zonal polynomials for  $\alpha = 2$  (he proved this conjecture for a special case). He also noted that for  $\alpha = -1$ , the “pre-zonal” polynomials are the (not so well-known) augmented monomial symmetric functions.

During the next decade, the study of Jack polynomials intensified; Macdonald (1995, p. 387) points out that in 1974, Foulkes (1974) raised the question of finding combinatorial interpretations for the Jack polynomials. This question was satisfactorily answered in 1997 by Knop and Sahi (1997).

Table 2  
Coefficients of the Hermite polynomial expressed in the Jack “C” polynomial basis

$k = 1$	$C_{[1]}^\alpha$				
$H_{[1]}^\alpha$	1	$k = 2$	$C_{[2]}^\alpha$	$C_{[1,1]}^\alpha$	$1 = C_{[\ ]}^\alpha$
		$H_{[2]}^\alpha$	1	0	$-\frac{n(n+\alpha)}{\alpha}$
		$H_{[1,1]}^\alpha$	0	1	$\frac{n(n-1)}{1+\alpha}$
$k = 3$	$C_{[3]}^\alpha$	$C_{[2,1]}^\alpha$	$C_{[1,1,1]}^\alpha$	$C_{[1]}^\alpha$	
$H_{[3]}^\alpha$	1	0	0	$\frac{3(n+\alpha)(n+2\alpha)}{(1+2\alpha)(1+\alpha)}$	
$H_{[2,1]}^\alpha$	0	1	0	$-\frac{6(n-1)(n+\alpha)(\alpha-1)}{(1+2\alpha)(2+\alpha)}$	
$H_{[1,1,1]}^\alpha$	0	0	1	$\frac{3\alpha(n-1)(n-2)}{(2+\alpha)(1+\alpha)}$	

Table 3  
Coefficients of the Laguerre polynomial expressed in the Jack “C” polynomial basis

$k = 1$	$C_{[1]}^\alpha$	$1 = C_{[\ ]}^\alpha$		
$L_{[1]}^{\alpha,\gamma}$	-1	$\frac{(\gamma\alpha+n+\alpha-1)n}{\alpha}$		
$k = 2$	$C_{[2]}^\alpha$	$C_{[1,1]}^\alpha$	$C_{[1]}^\alpha$	$1 = C_{[\ ]}^\alpha$
$L_{[2]}^{\alpha,\gamma}$	1	0	$\frac{2(\gamma\alpha+n+2\alpha-1)(n+\alpha)}{\alpha(1+\alpha)}$	$\frac{(\gamma\alpha+n+\alpha-1)(\gamma\alpha+n+2\alpha-1)n(n+\alpha)}{\alpha^2(1+\alpha)}$
$L_{[1,1]}^{\alpha,\gamma}$	0	1	$-\frac{2(\gamma\alpha+n+\alpha-2)(n-1)}{1+\alpha}$	$\frac{(\gamma\alpha+n+\alpha-1)(\gamma\alpha+n+\alpha-2)n(n-1)}{\alpha(1+\alpha)}$

Table 4  
Coefficients of the Jacobi polynomial expressed in the Jack “C” polynomial basis

$k = 1$	$C_{[1]}^\alpha$	$1 = C_{[\ ]}^\alpha$		
$J_{[1]}^{\alpha,g_1,g_2}$	-1	$\frac{(g_1\alpha+n+\alpha-1)n}{g_1\alpha+g_2\alpha+2n-2+2\alpha}$		
$k = 2$	$C_{[2]}^\alpha$	$C_{[1,1]}^\alpha$	$C_{[1]}^\alpha$	$1 = C_{[\ ]}^\alpha$
$J_{[2]}^{\alpha,g_1,g_2}$	1	0	$\frac{2(g_1\alpha+n+2\alpha-1)(n+\alpha)}{(g_1\alpha+g_2\alpha+2n-2+4\alpha)(1+\alpha)}$	$\frac{(g_1\alpha+n+\alpha-1)(g_1\alpha+n+2\alpha-1)n(n+\alpha)}{(g_1\alpha+g_2\alpha+2n-2+4\alpha)(g_1\alpha+g_2\alpha+2n-2+3\alpha)\alpha(1+\alpha)}$
$J_{[1,1]}^{\alpha,g_1,g_2}$	0	1	$-\frac{2\alpha(g_1\alpha+n+\alpha-2)(n-1)}{(g_1\alpha+g_2\alpha+2n-4+2\alpha)(1+\alpha)}$	$\frac{2\alpha(g_1\alpha+n+\alpha-1)(g_1\alpha+n+\alpha-2)n(n-1)}{(g_1\alpha+g_2\alpha+2n-4+2\alpha)(g_1\alpha+g_2\alpha+2n-3+2\alpha)(1+\alpha)}$

Two of the most comprehensive sources of information on Jack polynomial properties are Stanley’s paper (Stanley, 1989) (which appeared in 1989) and Macdonald’s book (Macdonald, 1995) (from 1995); these two authors generalized most of the known properties of the Schur functions and zonal polynomials to Jack polynomials.

As mentioned, an important application of the Jack polynomials came in conjunction with random matrix theory and statistics of the  $2/\alpha$ -ensembles. Below we mention a few of the many researchers who have made significant contributions in this area.

James (1964) was one of the first to make the connection between the zonal polynomials ( $\alpha = 2$  Jack polynomials) and the 1-ensembles, when he calculated statistical averages of zonal polynomials over the 1-Laguerre ensemble (Wishart central and non-central distributions).

At about the same time, Constantine and Muirhead provided a generalization of the hypergeometric series, using the zonal polynomials, and studied the multivariate Laguerre polynomials for  $\alpha = 1$  (for a reference, see Muirhead (1982)).

In a survey paper, James defined and described multivariate Laguerre, Hermite and Jacobi polynomials for  $\alpha = 1$  (James, 1975). Chikuse (1992) studied more extensively the multivariate Hermite polynomials for  $\alpha = 1$ , and Van Diejen (1997) examined in detail the general  $\alpha$  case for the same.

In the early '90s, Kaneko (1993) studied the general  $\alpha$  binomial coefficients, and used them in connection with the study of hypergeometric series and multivariate Jacobi polynomials. He also studied Selberg-type integrals and established the connection with generalized Jacobi polynomials. A few years later, Okounkov and Olshanski (1997) considered shifted Jack polynomials for all  $\alpha$ , and proved that they were the same as the generalized binomial coefficients.

Kadell (1997) was perhaps the first to consider averages of many valued Jack polynomials, with his study of the average of the Jack polynomial of parameter  $1/k$  (with  $k$  an integer) over the corresponding  $2k$ -Jacobi ensemble. Later it was noticed that constraining  $k$  to be an integer was unnecessary.

Lasalle (1991a,b,c) considered all three kinds of general  $\alpha$  multivariate polynomials, and among many other things computed generating functions for them.

The last results that we mention here are those of Baker and Forrester (1997), who studied in detail the multivariate, general  $\alpha$  Hermite and Laguerre polynomials, in connection with the  $2/\alpha$ -Hermite and Laguerre ensembles (some of their work built on Lasalle (1991a,c)). For a good reference on multivariate generalizations of many of the univariate properties of the Hermite and Laguerre ensembles, see Forrester (2001).

## 2. Multivariate functions: Definitions, properties, and algorithms

### 2.1. Partitions and symmetric polynomials

**Definition 2.1.** A partition  $\lambda$  is a finite, ordered, non-increasing sequence of positive integers  $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \dots \geq \lambda_l$ .

Throughout this paper, we will refer to  $l = l(\lambda) = \text{length}(\lambda)$  as the length of  $\lambda$ , and to  $k = |\lambda| = \sum_{i=1}^l \lambda_i$  as the sum of  $\lambda$ .

**Remark 2.2.** Naturally, one can remove the constraint “finite” from the definition of the partition, and replace it with “of finite sum”, since one can always “pad” a partition with 0s at the end; in this context  $l$  becomes the index of the smallest non-zero component of the partition  $\lambda$ .

We will work with two orderings of the partitions. The first one is the *lexicographic* one, denoted by  $\leq$ .

**Definition 2.3.** We say that  $\lambda \leq \kappa$  in lexicographical ordering if for the largest integer  $m$  such that  $\lambda_i = \kappa_i$  for all  $i < m$ , we have  $\lambda_m \leq \kappa_m$ . If  $\lambda_m < \kappa_m$ , we say that  $\lambda < \kappa$ .

**Remark 2.4.** This is a total ordering of the partitions.

The second ordering is the *dominance* ordering, sometimes also called the *natural* ordering.

**Definition 2.5.** We say that  $\lambda \leq \kappa$  (or, equivalently, that  $\kappa$  “dominates”  $\lambda$ ) if, given  $m = \max\{\text{length}(\kappa), \text{length}(\lambda)\}$ ,

$$\sum_{i=1}^j \lambda_i \leq \sum_{i=1}^j \kappa_i, \quad \forall j < m, \quad \text{and}$$

$$\sum_{i=1}^m \lambda_i = \sum_{i=1}^m \kappa_i.$$

If any one of the inequalities above is strict, we say that  $\lambda < \kappa$ .

**Remark 2.6.** Note that we compare two partitions only if they sum to the same integer. Also note that even with this constraint,  $\leq$  is only a partial ordering of the set of partitions of a given number: for example,  $[4, 1, 1]$  and  $[3, 3]$  are incomparable.

The above summarizes what the user should know about partitions in order to use our library.

**Definition 2.7.** A symmetric polynomial of  $m$  variables,  $x_1, \dots, x_m$ , is a polynomial which is invariant under every permutation of  $x_1, \dots, x_m$ .

**Remark 2.8.** The symmetric polynomials form a vector space over  $\mathbb{R}$ .

Over the course of time, combinatorialists have defined a variety of *homogeneous* bases for this vector space; each such basis is indexed by partitions (which correspond to the term of highest order in lexicographical ordering of the polynomial). By homogeneity we mean that all terms of a polynomial in the basis have the same total degree (but this degree varies from polynomial to polynomial).

Some of these homogeneous bases are displayed in the table below:

Name	Definition for $l = 1$	Definition for $l > 1$
Power-sum functions	$p_{\lambda_1} = \sum_{j=1}^m x_j^{\lambda_1}$	$p_{\lambda} = \prod_{i=1}^l p_{\lambda_i}$
Elementary functions	$e_{\lambda_1} = \sum_{j_1 < j_2 < \dots < j_{\lambda_1}} x_{j_1} \dots x_{j_{\lambda_1}}$	$e_{\lambda} = \prod_{i=1}^l e_{\lambda_i}$
Complete homogeneous functions	$h_{\lambda_1} = \sum_{j_1 \leq j_2 \leq \dots \leq j_{\lambda_1}} x_{j_1} \dots x_{j_{\lambda_1}}$	$h_{\lambda} = \prod_{i=1}^l h_{\lambda_i}$

Another important basis is given by the monomial functions  $m$ ,

$$m_{\lambda} = \sum_{\sigma \in S_{\lambda}} x_{\sigma(1)}^{\lambda_1} x_{\sigma(2)}^{\lambda_2} \dots x_{\sigma(m)}^{\lambda_m};$$

here  $S_{\lambda}$  is the set of permutations giving distinct terms in the sum;  $\lambda$  is considered as infinite.

The last basis we mentioned distinguishes itself from the other ones in two ways; the advantage is that it is very easy to visualize, and proving that it is indeed a basis is immediate. The disadvantage is that it is not multiplicative.<sup>3</sup>

<sup>3</sup> For  $x \in \{p, e, h\}$ ,  $x_{\lambda} x_{\mu} = x_{\rho}$ , where  $\rho$  can be obtained in an algorithmic fashion from  $\lambda$  and  $\mu$  (sometimes by mere concatenation and reordering). In general,  $m_{\lambda} m_{\mu}$  is not a monomial.

Monomials seem to be the basis of choice for most people working in statistics or engineering. Combinatorialists often (but not always) prefer to express series in the power-sum basis, because of connections with character theory.

2.2. Multivariate Gamma function

Before we proceed, we will need to define the multivariate Gamma function for arbitrary  $\alpha$ ; the real and complex versions are familiar from the literature, and the arbitrary  $\alpha > 0$  case represents an immediate extension:

$$\Gamma_m^\alpha(a) = \pi^{m(m-1)/(2\alpha)} \prod_{i=1}^m \Gamma\left(a - \frac{i-1}{\alpha}\right). \tag{3}$$

Just as the univariate Gamma function generalizes to the multivariate one, the shifted factorial (Pochhammer symbol, rising factorial)

$$(a)_k = \frac{\Gamma(a+k)}{\Gamma(a)}$$

becomes the generalized shifted factorial. We call

$$(a)_\kappa^\alpha = \prod_{i=1}^{\text{length}(\kappa)} \left(a - \frac{i-1}{\alpha}\right)_{\kappa_i} = \prod_{i=1}^{\text{length}(\kappa)} \frac{\Gamma\left(a - \frac{i-1}{\alpha} + \kappa_i\right)}{\Gamma\left(a - \frac{i-1}{\alpha}\right)} \tag{4}$$

the *generalized shifted factorial*, or *generalized Pochhammer symbol*.

2.3. Jack polynomials (the multivariate monomials)

Jack polynomials allow for several equivalent definitions (up to certain normalization constraints). In addition to the definition presented in the introduction (at the end of Section 1.1), we present here two more (Definitions 2.9 and 2.10). Definition 2.9 arose in combinatorics, whereas Definition 2.10 arose in statistics. We will mainly work with Definition 2.10.

**Definition 2.9** (Following Macdonald (1995)). The Jack polynomials  $P_\lambda^\alpha$  are orthogonal with respect to the inner product defined below on power-sum functions

$$\langle p_\lambda, p_\mu \rangle_\alpha = \alpha^{l(\lambda)} z_\lambda \delta_{\lambda,\mu},$$

where  $z_\lambda = \prod_{i=1}^{l(\lambda)} a_i! i^{a_i}$ ,  $a_i$  being the number of occurrences of  $i$  in  $\lambda$ . In addition,

$$P_\lambda^\alpha = m_\lambda + \sum_{\mu \preceq \lambda} u_{\lambda,\mu}^\alpha m_\mu.$$

There are two main normalizations of the Jack polynomials used in combinatorics, the ‘‘J’’ normalization (which makes the coefficient of the lowest-order monomial,  $[1^n]$ , be exactly  $n!$ ) and the ‘‘P’’ normalization (which is monic, and is given in Definition 2.9). To convert between these normalizations, see Tables 5 and 6. In Table 5,  $I_m = (1, 1, 1, \dots, 1)$ , where the number of variables is  $m$ .

We use the notation  $\kappa \vdash k$  for  $\kappa$  a partition of  $k$ , and  $\rho_\kappa^\alpha$  for  $\sum_{i=1}^m k_i(k_i - 1 - \frac{2}{\alpha}(i - 1))$ .



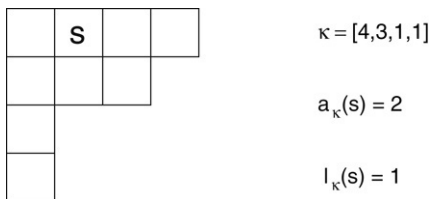


Fig. 1. The Ferrers diagram, arm-length and leg-length.

**Definition 2.10** (Following Muirhead (1982)). The Jack polynomial  $C_\kappa^\alpha$  is the only homogeneous polynomial eigenfunction of the following Laplace–Beltrami-type operator:

$$D^* = \sum_{i=1}^m x_i^2 \frac{d^2}{dx_i^2} + \frac{2}{\alpha} \sum_{1 \leq i \neq j \leq m} \frac{x_i^2}{x_i - x_j} \frac{d}{dx_i},$$

with eigenvalue  $\rho_\kappa^\alpha + k(m - 1)$ , having highest-order term corresponding to  $\kappa$ . In addition,

$$\sum_{\kappa \vdash k, l(\kappa) \leq m} C_\kappa^\alpha(x_1, x_2, \dots, x_m) = (x_1 + x_2 + \dots + x_m)^k.$$

**Remark 2.11.** The “C” normalization for the Jack polynomial allows for defining scalar hypergeometric functions of multivariate (or matrix) argument. These are useful for computing Selberg-type integrals and other quantities which appear in various fields, from the theory of random walks to multivariate statistics and quantum many-body problems.

**Remark 2.12.** Jackson (2003) pointed out that the  $D^*$  operator also appears in algebraic geometry, for example in the context of ramified covers. We thank the anonymous referee who noted that this operator, modulo a similarity transformation, is the Hamiltonian of the quantum Calogero–Sutherland model, which describes  $m$  identical particles on a circle.

**Definition 2.13.** Given a partition  $\kappa$  with  $|\kappa| = n$ , the Ferrers diagram corresponding to  $\kappa$  is an arrangement of  $n$  boxes in  $l(\kappa)$  left-justified rows, the number of boxes in row  $i$  being the same as  $\kappa_i$  (see Fig. 1).

**Definition 2.14.** Given the Ferrers diagram of a partition  $\kappa$  (see Fig. 1), define  $a_\kappa(s)$  (the “arm-length”) as the number of squares to the right of  $s$ ;  $l_\kappa(s)$  (the “leg-length”) as the number of squares below  $s$ ;  $h_\kappa^*(s) = l_\kappa(s) + \alpha(1 + a_\kappa(s))$  (the “upper hook length”) and  $h_\kappa^\kappa(s) = l_\kappa(s) + 1 + \alpha a_\kappa(s)$  (the “lower hook length”).

Finally, a further definition is needed in order to present the conversion table.

**Definition 2.15.** Let

$$c(\kappa, \alpha) = \prod_{s \in \kappa} h_\kappa^*(s),$$

$$c'(\alpha, \kappa) = \prod_{s \in \kappa} h_\kappa^\kappa(s),$$

$$j_\kappa = c(\alpha, \kappa) c'(\alpha, \kappa),$$

where  $h_\kappa^*$  and  $h_\kappa^\kappa$  have been defined above.

To explain the conversions between “J”, “P”, and “C”, we recall the definition of the generalized Gamma function and generalized shifted factorial from Section 2.2.

We can now present Tables 5 and 6; the entries have been filled out using James (1968), Baker and Forrester (1997), and Stanley (1989).

Table 5  
Values of the different normalizations of Jack polynomials of partition  $\kappa$  and parameter  $\alpha$  at  $I_m$

Normalization	Basic property	Value at $I_m = (1, 1, 1, \dots, 1)$
C	Sums to $(x_1 + x_2 + \dots + x_n)^k$	$C_\kappa^\alpha(I_m) = \frac{\alpha^{2k} k!}{j_\kappa} \left(\frac{m}{\alpha}\right)_\kappa$
J	Has trailing coefficient $n!$	$J_\kappa^\alpha(I_m) = \alpha^k \left(\frac{m}{\alpha}\right)_\kappa$
P	Is monic	$P_\kappa^\alpha(I_m) = \frac{\alpha^k}{c(\alpha, \kappa)} \left(\frac{m}{\alpha}\right)_\kappa$

Table 6  
Conversions between the three normalizations for the Jack polynomials; the  $a(V, W)$  entry above is defined as  $V_\kappa^\alpha(x_1, \dots, x_m) = a(V, W)W_\kappa^\alpha(x_1, \dots, x_m)$

	C	J	P
C		$\frac{\alpha^k k!}{j_\kappa}$	$\frac{\alpha^k k!}{c'(\kappa, \alpha)}$
J	$\frac{j_\kappa}{\alpha^k k!}$		$c(\kappa, \alpha)$
P	$\frac{c'(\kappa, \alpha)}{\alpha^k k!}$	$\frac{1}{c(\kappa, \alpha)}$	

2.4. Algorithm used to compute the Jack polynomials

From the Laplace–Beltrami equation, one can find an expansion for the Jack polynomials of the type

$$C_\kappa^\alpha(x_1, x_2, \dots, x_m) = \sum_{\lambda \leq \kappa} c_{\kappa, \lambda}^\alpha m_\lambda(x_1, x_2, \dots, x_m),$$

where  $\lambda$  and  $\kappa$  are both partitions of the same integer  $|\kappa|$ , and the order imposed on partitions is the lexicographic one. The coefficients  $c_{\kappa, \lambda}^\alpha$  depend on all three parameters;  $m_\lambda(x_1, x_2, \dots, x_m)$  is the monomial function corresponding to  $\lambda$ .

Note that as a consequence of the above, if  $l(\kappa) > m$ ,  $C_\kappa^\alpha(x_1, x_2, \dots, x_m) = 0$  (“there is no highest-order term”).

Using the eigenfunction equation

$$D^* C_\kappa^\alpha = (\rho_\kappa^\alpha + k(m - 1))C_\kappa^\alpha, \tag{5}$$

where

$$\rho_\kappa^\alpha = \sum_{i=1}^m k_i \left( k_i - 1 - \frac{2}{\alpha}(i - 1) \right)$$

one can obtain a recurrence for  $c_{\kappa, \lambda}^\alpha$  from which the Jack polynomials can be explicitly calculated.

The following result can be found, without proof, as an exercise (Macdonald, 1995, Exercise 3(d), p. 327), and it was proved in Lapointe et al. (2000); we have re-discovered it independently.

**Proposition 2.16.** *The coefficients  $c_{\kappa,\lambda}^\alpha$  satisfy the recurrence*

$$c_{\kappa,\lambda}^\alpha = \frac{\frac{2}{\alpha}}{\rho_\kappa^\alpha - \rho_\lambda^\alpha} \sum_{\lambda < \mu \leq \kappa} \left( (l_i + t) - (l_j - t) \right) c_{\kappa,\mu}^\alpha, \tag{6}$$

where  $\lambda = (l_1, \dots, l_i, \dots, l_j, \dots, l_m)$ ,  $\mu = (l_1, \dots, l_i + t, \dots, l_j - t, \dots, l_m)$ , and  $\mu$  has the property that, when properly reordered, it is between  $\lambda$  (strictly) and  $\kappa$  in lexicographic order.

The proof of Proposition 2.16 is technical and we omit it; it is however noteworthy that the proof was generalized to larger families of multivariate orthogonal polynomials in Van Diejen et al. (2004).

Using two propositions found in Macdonald’s book (Macdonald, 1995, (10.13), (10.15)), we can clean up the formula a little bit. Roughly, the content of the two propositions is that the Jack polynomials, in “P” normalization, can be written as

$$P_\kappa^\alpha = m_\kappa + \sum_{\lambda < \kappa} u_{\kappa,\lambda}^\alpha m_\lambda,$$

with  $u_{\kappa,\lambda}^\alpha > 0$  whenever  $\kappa > \lambda$  (the ordering imposed on partitions here is the dominance ordering).

Thus it follows that recurrence (6) can be improved to

$$c_{\kappa,\lambda}^\alpha = \frac{\frac{2}{\alpha}}{\rho_\kappa^\alpha - \rho_\lambda^\alpha} \sum_{\lambda < \mu \leq \kappa} \left( (l_i + t) - (l_j - t) \right) c_{\kappa,\mu}^\alpha, \tag{7}$$

where  $\lambda = (l_1, \dots, l_i, \dots, l_j, \dots, l_m)$ ,  $\mu = (l_1, \dots, l_i + t, \dots, l_j - t, \dots, l_m)$ , and  $\mu$  has the property that, when properly reordered, it is between  $\lambda$  (strictly) and  $\kappa$  in domination order.

This recurrence, at first glance, seems to be enough to compute all coefficients  $c_{\kappa,\lambda}^\alpha$ , once  $c_{\kappa,\kappa}^\alpha$  is found. However, for well-definedness, we need to show that  $\rho_\kappa^\alpha \neq \rho_\lambda^\alpha$  for any  $\lambda$  different from  $\kappa$ .

We first need the following well-known proposition.

**Proposition 2.17.** *The dominance ordering is a lattice on the set of partitions of a given number. In particular, between any partitions  $\kappa$  and  $\lambda$  such that  $\kappa > \lambda$ , there exists a “path” on this lattice,  $\sigma^0 = \kappa > \sigma^1 > \dots > \sigma^t = \lambda$ , such that  $\sigma^{i+1}$  differs from  $\sigma^i$  in the following way: there exists  $i_1 < i_2$  such that  $\sigma^{i+1}$  and  $\sigma^i$  agree in all places but  $i_1$  and  $i_2$ ,  $(\sigma^{i+1})_{i_1} = (\sigma^i)_{i_1} - 1$ , and  $(\sigma^{i+1})_{i_2} = (\sigma^i)_{i_2} + 1$ .*

The fact that recurrence (6) yields a well-defined polynomial is proved below.

**Lemma 2.18.** *If  $\lambda < \kappa$ , then  $\rho_\lambda^\alpha \neq \rho_\kappa^\alpha$ , for all  $\alpha > 0$ .*

**Proof.** Let  $\lambda < \kappa$  be two partitions, let  $m = \max\{\text{length}(\kappa), \text{length}(\lambda)\}$ , and assume that there is some  $\alpha > 0$  such that

$$\rho_\lambda^\alpha = \rho_\kappa^\alpha.$$

Since the two partitions sum to the same number, the above is equivalent to

$$\sum_{i=1}^m k_i^2 - \lambda_i^2 = \frac{2}{\alpha} \sum_{i=1}^m (k_i - \lambda_i)(i - 1).$$

The right-hand side is non-positive (as an immediate consequence of the strict ordering).

We show that the left-hand side is positive by induction, thus obtaining a contradiction. For that we will use Proposition 2.17, which shows that it is enough to prove that

$$\sum_{i=1}^m k_i^2 - \lambda_i^2 \geq 0$$

in the case when  $\kappa$  and  $\lambda$  differ only in two places,  $i_1 < i_2$ . Note that if  $\kappa_{i_1} = \lambda_{i_1} + 1$  and  $\kappa_{i_2} = \lambda_{i_2} - 1$ , this implies that  $\kappa_{i_1} \geq \kappa_{i_2} + 2$ . Hence

$$\sum_{i=1}^m k_i^2 - \lambda_i^2 = k_{i_1}^2 - \lambda_{i_1}^2 + k_{i_2}^2 - \lambda_{i_2}^2 = 2k_{i_1} - 1 - 2k_{i_2} - 1 \geq 2 > 0,$$

and we are done.  $\square$

Proposition 2.17 ensures thus that once  $c_{\kappa\kappa}^\alpha$  is determined, every other non-zero coefficient is uniquely determined.

Finally, for  $c_{\kappa\kappa}^\alpha$  we use the following formula (deduced on the basis of Table 5 and the fact that  $P_\kappa^\alpha$  has highest-order coefficient 1):

$$c_{\kappa\kappa}^\alpha = \frac{\alpha^k k!}{c'(\kappa, \alpha)}.$$

**Remark 2.19.** It is worth mentioning that, from the recurrence (7), by letting  $\alpha \rightarrow \infty$ , the coefficient  $c_{\kappa,\lambda}^\alpha$  goes to 0 faster than  $c_{\kappa,\kappa}^\alpha$ , for any  $\lambda \neq \kappa$ . Thus, at  $\alpha = \infty$ , the Jack “P” polynomial (which is monic) is the symmetric monomial. This could also be seen from the weight functions, as at  $\alpha = \infty$ , the “interdependence” term  $\prod_{1 \leq i < j \leq n} |x_i - x_j|^{2/\alpha}$  (see for example (1)) disappears and the variables separate.

### 2.5. Multivariate binomial coefficients

Many algebraic quantities (and the identities that they satisfy) can be extended from the univariate case to the multivariate case through Jack polynomials. One such example is the multivariate, or generalized, binomial coefficient.

**Definition 2.20.** We define the multivariate (or generalized) binomial coefficients  $\binom{\kappa}{\sigma}$  as

$$\frac{C_\kappa^\alpha(x_1 + 1, x_2 + 1, \dots, x_m + 1)}{C_\kappa^\alpha(1, 1, \dots, 1)} = \sum_{s=0}^k \sum_{\sigma \vdash s, \sigma \subseteq \kappa} \binom{\kappa}{\sigma} \frac{C_\sigma^\alpha(x_1, x_2, \dots, x_m)}{C_\sigma^\alpha(1, 1, \dots, 1)},$$

where  $\sigma \subseteq \kappa$  means that  $\sigma_i \leq \kappa_i$  for all  $i$ .

The generalized binomial coefficients depend on  $\alpha$ , but are independent of both the number of variables  $m$  and the normalization of the Jack polynomials (the latter independence is easily seen from the definition).

The multivariate binomial coefficients generalize the univariate ones; some simple properties of the former are straightforward generalizations of properties of the latter. For example,

$$\begin{aligned} \binom{\kappa}{(0)} &= 1, \\ \binom{\kappa}{(1)} &= |\kappa|, \\ \binom{\kappa}{\sigma} &= 0 \quad \text{if } \sigma \not\subseteq \kappa, \\ \binom{\kappa}{\sigma} &= \delta_{\kappa\sigma} \quad \text{if } |\kappa| = |\sigma|, \\ \binom{\kappa}{\sigma} &\neq 0 \quad \text{if } |\kappa| = |\sigma| + 1, \text{ iff } \sigma = \kappa_{(i)}, \end{aligned}$$

where  $\kappa_{(i)} = (k_1, \dots, k_i - 1, \dots, k_m)$ . The above are true for all  $\kappa$  and  $\alpha$ , and  $\sigma$  subject to the constraints.

### 2.6. Algorithm used to compute the multivariate binomial coefficients

One can prove, using the eigenfunction equation (5) and the definition of the generalized binomial coefficients, that

$$\sum_i \binom{\sigma^{(i)}}{\sigma} \binom{\kappa}{\sigma^{(i)}} = (k - s) \binom{\kappa}{\sigma}, \tag{8}$$

where  $|\sigma| = s$ ,  $|\kappa| = k$ ,  $\sigma^{(i)} = (\sigma_1 \dots, \sigma_i + 1, \dots, \sigma_m)$ . All generalized binomial coefficients can be found recursively, once one has a way to compute the so-called ‘‘contiguous’’ coefficients  $\binom{\sigma^{(i)}}{\sigma}$ .

To compute the contiguous coefficients, we use Proposition 2 from Kaneko (1993), applied to  $\kappa = \sigma^{(i)}$ , and simplified slightly:

$$\binom{\sigma^{(i)}}{\sigma} = j_{\sigma}^{-1} g_{\sigma 1}^{\sigma^{(i)}}, \tag{9}$$

where  $g_{\sigma 1}^{\sigma^{(i)}}$  is

$$g_{\sigma 1}^{\sigma^{(i)}} = \left( \prod_{s \in \sigma} A_{\sigma^{(i)}} \right) \left( \prod_{s \in \sigma} B_{\sigma^{(i)}} \right).$$

Here

$$\begin{aligned} A_{\sigma^{(i)}} &= \begin{cases} h_{\sigma}^{\sigma}(s), & \text{if } s \text{ is not in the } i\text{th column of } \sigma, \\ h_{\sigma}^*(s), & \text{otherwise.} \end{cases} \\ B_{\sigma^{(i)}} &= \begin{cases} h_{\sigma^{(i)}}^*(s), & \text{if } s \text{ is not in the } i\text{th column of } \sigma, \\ h_{\sigma}^{\sigma^{(i)}}(s), & \text{otherwise.} \end{cases} \end{aligned}$$

Knowing the contiguous coefficients allows computing all generalized binomial coefficients.

**Remark 2.21.** The generalized binomial coefficients are independent of the number of variables. They are rational functions of  $\alpha$ .

2.7. Multivariate orthogonal polynomials

2.7.1. Jacobi polynomials

These polynomials represent the Gram–Schmidt orthogonalization of the Jack polynomials  $C_\lambda^\alpha$  with respect to the Jacobi weight function

$$d\mu_j^\alpha(x_1, \dots, x_m) = \frac{\pi^{\frac{m(m-1)}{\alpha}} (\Gamma(1 + \frac{1}{\alpha}))^m}{\Gamma_m^\alpha(1 + \frac{m}{\alpha})} \times \frac{\Gamma_m^\alpha(g_1 + g_2 + \frac{2}{\alpha}(m - 1) + 2)}{\Gamma_m^\alpha(g_1 + \frac{1}{\alpha}(m - 1) + 1)\Gamma_m^\alpha(g_2 + \frac{1}{\alpha}(m - 1) + 1)} \prod_{i=1}^m [x_i^{g_1} (1 - x_i)^{g_2}] \tag{10}$$

$$\times \prod_{i < j} |x_i - x_j|^{2/\alpha} dx_1 \dots dx_m, \tag{11}$$

on the hypercube  $[0, 1]^m$ . For the purpose of well-definedness we assume

$$g_1, g_2 > -1. \tag{12}$$

Define

$$\delta^* = \sum_i x_i \frac{d^2}{dx_i^2} + \frac{2}{\alpha} \sum_{i \neq j} \frac{x_i}{x_i - x_j} \frac{d}{dx_i}$$

$$E = \sum_i x_i \frac{d}{dx_i}$$

$$\epsilon = \sum_i \frac{d}{dx_i};$$

then the Jacobi polynomials are eigenfunctions of the following Laplace–Beltrami operator:

$$D^* + (g_1 + g_2 + 2)E - \delta^* - (g_1 + 1)\epsilon, \tag{13}$$

with eigenvalue  $\rho_\kappa^\alpha + |\kappa|(g_1 + g_2 + \frac{2}{\alpha}(m - 1) + 2)$ .

2.7.2. Algorithm used to compute the Jacobi polynomials

Using the fact that the Jacobi polynomials are eigenfunctions of the operator (13), one obtains that these polynomials can be written in the Jack polynomial basis as

$$J_{\kappa}^{\alpha, g_1, g_2}(x_1, \dots, x_m) = \left(g_1 + \frac{m - 1}{\alpha} + 1\right)_{\kappa} C_{\kappa}^{\alpha}(I_m) \times \sum_{\sigma \subseteq \kappa} \frac{(-1)^s c_{\kappa\sigma}^{\alpha}}{(g_1 + \frac{m-1}{\alpha} + 1)_{\sigma}} \frac{C_{\sigma}^{\alpha}(x_1, \dots, x_m)}{C_{\sigma}^{\alpha}(I_m)},$$

where the coefficients  $c_{\kappa\sigma}^{\alpha}$  satisfy the recurrence below.

**Theorem 2.22.** *The coefficients  $c_{\kappa, \sigma}^{\alpha}$  satisfy*

$$c_{\kappa\sigma}^{\alpha} = \frac{1}{\left((g_2 + g_1 + \frac{2}{\alpha}(m - 1) + 2)(k - s) + \rho_{\kappa}^{\alpha} - \rho_{\sigma}^{\alpha}\right)} \sum_{i \text{ allowable}} \binom{\kappa}{\sigma^{(i)}} \binom{\sigma^{(i)}}{\sigma} c_{\kappa\sigma^{(i)}}^{\alpha}, \tag{14}$$

with the previous notation for  $\rho_{\kappa}^{\alpha}$  and  $\sigma^{(i)}$ . This recurrence is well-defined.

Once again, we omit the technical proof. The fact that the recurrence (14) is well-defined follows from the lemma below.

**Lemma 2.23.** *With the notation above,  $(g_2 + g_1 + \frac{2}{\alpha}(m - 1) + 2)(k - s) + \rho_\kappa^\alpha - \rho_\sigma^\alpha$  is never 0 if  $\sigma \subset \kappa$ .*

**Proof.** The proof is very similar to the corresponding proof of Section 2.4; the two crucial facts here are that one needs to show it for the case  $\kappa = \sigma^{(i)}$ , and that  $g_1$  and  $g_2$  are both larger than  $-1$  (due to (12)).  $\square$

Letting  $c_{\kappa,\kappa}^\alpha = 1$  for all  $\kappa$  and  $\alpha$  allows all the coefficients to be uniquely determined.

### 2.7.3. Laguerre polynomials

The multivariate Laguerre polynomials are orthogonal with respect to the Laguerre weight function

$$d\mu_L^\alpha(x_1, \dots, x_m) = \frac{\pi^{\frac{m(m-1)}{\alpha}} (\Gamma(1 + \frac{1}{\alpha}))^m}{\Gamma_m^\alpha(1 + \frac{m}{\alpha})} \frac{1}{\Gamma_m^\alpha(\gamma + \frac{m-1}{\alpha} + 1)} \times e^{-\sum_i x_i} \prod_i x_i^\gamma \prod_{i \neq j} |x_i - x_j|^{2/\alpha} dx_1 \dots dx_m, \tag{15}$$

on the interval  $[0, \infty)^m$ . Note that for the purpose of well-definedness, we must have  $\gamma > -1$ .

This weight function can be obtained from the Jacobi weight function (10) of the previous subsection by substituting  $(g_1 + g_2 + \frac{2}{\alpha}(m - 1) + 2)^{-1}(x_1, \dots, x_m)$  for  $(x_1, \dots, x_m)$  and then taking the limit as  $g_2 \rightarrow \infty$ . The same limiting process applied to the Jacobi polynomials yields the Laguerre polynomials.

Under the transformation mentioned above, the Jacobi differential operator becomes

$$\delta^* - E + (\gamma + 1)\epsilon, \tag{16}$$

and the Laguerre polynomials are eigenfunctions of this operator with eigenvalue  $|\kappa|$ .

### 2.7.4. Algorithm used to compute the Laguerre polynomials

The Laguerre polynomials have an explicit expansion in the Jack polynomial basis, which depends on the generalized binomial coefficients:

$$L_\kappa^{\alpha,\gamma}(x_1, \dots, x_m) = \left(\gamma + \frac{m-1}{\alpha} + 1\right)_\kappa C_\kappa^\alpha(I_m) \sum_{\sigma \subseteq \kappa} \frac{(-1)^s \binom{\kappa}{\sigma}}{(\gamma + \frac{m-1}{\alpha} + 1)_\sigma} \frac{C_\sigma^\alpha(x_1, \dots, x_m)}{C_\sigma^\alpha(I_m)}.$$

Note that the coefficient of  $C_\kappa^\alpha(x_1, \dots, x_m)$  in  $L_\kappa^{\alpha,\gamma}(x_1, \dots, x_m)$  is  $(-1)^k$ .

### 2.7.5. Hermite polynomials

The multivariate Hermite polynomials are orthogonal with respect to the Hermite weight function

$$d\mu_H^\alpha(x_1, \dots, x_m) = 2^{-m/2} \pi^{m(m-1)/\alpha - m/2} \frac{(\Gamma(1 + \frac{1}{\alpha}))^m}{\Gamma_m^\alpha(1 + \frac{m}{\alpha})} \tag{17}$$

$$\times e^{-\sum_{i=1}^m x_i^2/2} \prod_{i \neq j} |x_i - x_j|^{2/\alpha} dx_1 \dots dx_m, \tag{18}$$

on  $\mathbb{R}^m$ .

This weight function can be obtained by taking  $(\gamma + \sqrt{\gamma}x_1, \gamma + \sqrt{\gamma}x_2, \dots, \gamma + \sqrt{\gamma}x_m)$  in (15), and then letting  $\gamma$  go to infinity; note that the only remaining parameter is  $\alpha$ .

Under this limiting process, the differential operator becomes

$$\delta^{**} - E, \tag{19}$$

where

$$\delta^{**} = \sum_i \frac{d^2}{dx_i^2} + \frac{2}{\alpha} \sum_{i \neq j} \frac{1}{x_i - x_j} \frac{d}{dx_i}.$$

The Hermite polynomials are eigenfunctions of this operator with eigenvalue  $|\kappa|$ .

**Remark 2.24.** Similarly,

$$\lim_{\gamma \rightarrow \infty} \gamma^{-k/2} L_k^{\alpha, \gamma}(\gamma + \sqrt{\gamma}x_1, \gamma + \sqrt{\gamma}x_2, \dots, \gamma + \sqrt{\gamma}x_m) = (-1)^k H_k^\alpha(x_1, \dots, x_m). \tag{20}$$

2.7.6. Algorithm used to compute the Hermite polynomials

Using the corresponding Hermite differential operator (19), we obtain that the  $H_k^\alpha$  are expressible as

$$H_k^\alpha(x_1, \dots, x_m) = \sum_{\sigma \subseteq \kappa} c_{\kappa, \sigma}^\alpha \frac{C_\sigma^\alpha(x_1, \dots, x_m)}{C_\sigma^\alpha(I_m)};$$

where the coefficients  $c_{\kappa, \sigma}^\alpha$  satisfy the recurrence below.

**Theorem 2.25.** The coefficients  $c_{\kappa, \sigma}^\alpha$  satisfy

$$\begin{aligned} c_{\kappa, \sigma}^\alpha &= \frac{1}{k-s} \left( \sum_i \binom{\sigma^{(i)(i)}}{\sigma^{(i)}} \binom{\sigma^{(i)}}{\sigma} c_{\kappa, \sigma^{(i)(i)}}^\alpha \right. \\ &\quad \left. + \sum_{i < j} \left( \sigma_i - \sigma_j - \frac{1}{\alpha}(i-j) \right) \binom{\sigma^{(i)(j)}}{\sigma^{(j)}} \binom{\sigma^{(j)}}{\sigma} c_{\kappa, \sigma^{(i)(j)}}^\alpha \right). \end{aligned} \tag{21}$$

This recurrence is well-defined. In the above,  $i < j$  take on all admissible values, and the choice  $c_{\kappa, \kappa}^\alpha = C_\kappa^\alpha(I_m)$  determines the normalization.

The proof is technical and we omit it.

Alternatively, we can obtain the coefficients directly through the limiting process described in Remark 2.24; below we have an equivalent formula.

**Theorem 2.26.** Define  $X = (x_1, \dots, x_m)$ . Denote by  $[r^s]F(r)$  the coefficient of the power  $s$  of  $r$  in the polynomial  $F(r)$ . Then

$$\begin{aligned} [C_\sigma^\alpha(X)] H_\kappa^\alpha(X) &= \frac{C_\kappa^\alpha(I_m)}{C_\sigma^\alpha(I_m)} \sum_{j=s}^{\frac{k+s}{2}} (-1)^{k-j} \\ &\quad \times \sum_{\sigma \subseteq \mu \subseteq \kappa; \mu \vdash j} \binom{\kappa}{\mu} \binom{\mu}{\sigma} \left[ r^{\frac{k+s}{2}-j} \right] F(r, \alpha, m, \kappa, \sigma), \end{aligned} \tag{22}$$



where

$$F(r, \alpha, m, \kappa, \sigma) = \frac{(r + \frac{1}{\alpha}(m + \alpha - 1))_{\kappa}}{(r + \frac{1}{\alpha}(m + \alpha - 1))_{\sigma}}.$$

The proof starts with the asymptotical formula (20) and identifies the coefficients of the Jack polynomials on both sides.

We use formula (22) to calculate a single coefficient,  $c_{\kappa, \square}^{\alpha}$ , for reasons of smaller computational complexity (in computing integrals with respect to the Hermite weight function; see Section 3.3).

Note that if  $\sigma \not\subseteq \kappa$  or  $k \neq s \pmod{2}$ , then formula (22) yields that the coefficient of  $C_{\sigma}^{\alpha}(X)$  in  $H_{\kappa}^{\alpha}(X)$  is 0.

### 2.8. Hypergeometric functions

The hypergeometric functions are perhaps the easiest to generalize from univariate to multivariate. For the multivariate versions, a good reference is Forrester’s unpublished book (Forrester, 2001).

**Definition 2.27.** We define the hypergeometric function  ${}_pF_q^{\alpha}$  of parameters  $a_1, \dots, a_p$  and  $b_1, \dots, b_q$ , and of variables  $(x_1, \dots, x_m)$  by

$${}_pF_q^{\alpha}(a_1, \dots, a_p; b_1, \dots, b_q; x_1, \dots, x_m) = \sum_{k=0}^{\infty} \sum_{\kappa \vdash k} \frac{(a_1)_{\kappa} \dots (a_p)_{\kappa}}{k! (b_1)_{\kappa} \dots (b_q)_{\kappa}} C_{\kappa}^{\alpha}(x_1, \dots, x_m).$$

Note that this is a formal definition;  $p \leq q$  is needed in order for the hypergeometric series to converge everywhere, and when  $p = q + 1$ , there is a non-trivial convergence radius. When  $p \geq q + 2$ , the series converges everywhere except at 0, with one notable exception, constituted by the polynomial hypergeometrics, i.e. those for which some  $a_i$  is a negative integer, which forces the series to terminate after a finite number of terms.

This definition of a hypergeometric function assumes an argument

$$(x_1, \dots, x_m) \in \mathbb{R}^m;$$

similarly one can extend the definition to hypergeometric functions of arguments in

$$(x_1, \dots, x_m; y_1, \dots, y_m; \dots) \in \mathbb{R}^m \times \mathbb{R}^m \times \dots$$

by inserting an additional  $C_{\kappa}^{\alpha}(y_1, \dots, y_m)/C_{\kappa}^{\alpha}(1, \dots, 1)$  for each extra vector in  $\mathbb{R}^m$ .

Hypergeometric functions provide answers to many statistics and statistics-related questions; below are two examples.

1. Krishnaiah and Chang (1971) proved in 1971 that the density of the smallest root of a real ( $\alpha = 2$ ) Wishart matrix with  $m$  variables and  $n$  degrees of freedom such that  $p = \frac{n-m-1}{2}$  is an integer, is proportional to

$$\rho(x) = x^p e^{-xm/2} {}_2F_0\left(-p, \frac{m+2}{2}; -2I_{m-1}/x\right).$$

Note that the joint eigenvalue density of the matrix described above is given by  $d\mu_L^{\alpha}$  with  $\alpha = 2$  and  $\gamma = p$ .

In Dumitriu (2003) we extend this to any  $\alpha$  and any positive integer  $\gamma = p$ . We obtain that for this case the density of the smallest eigenvalue is proportional to

$$\rho(x) = x^{pm} e^{-xm/2} {}_2F_0^\alpha \left( -p, \frac{m}{\alpha} + 1; -2I_{m-1}/x \right). \tag{23}$$

2. The largest eigenvalue ( $l_1$ ) distribution for a Wishart real matrix with  $m$  variables and  $n$  degrees of freedom ( $\alpha = 2, \gamma = \frac{n-m-1}{2}$ ) can be expressed as

$$P[l_1 < x] = \frac{\Gamma_m[\frac{1}{2}(m+1)]}{\Gamma_m[\frac{1}{2}(n+m+1)]} \left(\frac{x}{2}\right)^{mn/2} {}_1F_1 \left( \frac{1}{2}n, \frac{1}{2}(n+m+1); -\frac{1}{2}xI_m \right).$$

The above is a corollary of a stronger theorem proved by Constantine Constantine (1963), and it can also be found in Muirhead (1982, p. 421).

This result generalizes for any  $\alpha$  and  $\gamma$  (as noted in Dumitriu (2003)) to

$$P[l_1 < x] = \frac{\Gamma_m[\frac{1}{\alpha}(m-1)+1]}{\Gamma_m[\gamma + \frac{2}{\alpha}(m-1)+2]} \left(\frac{x}{2}\right)^{m(\gamma+(m-1)/\alpha+1)} \times {}_1F_1 \left( \gamma + \frac{(m-1)}{\alpha} + 1, a + \frac{2}{\alpha}(m-1)+2; -\frac{1}{2}xI_m \right).$$

2.9. Computing expected values of polynomials

Let  $P_\lambda(x_1, \dots, x_m)$  be a symmetric polynomial in  $m$  variables, with highest-order term corresponding to the partition  $\lambda$ . To compute the expected value of  $P_\lambda$  with respect to one of the distributions  $d\mu_J^\alpha$  (10),  $d\mu_L^\alpha$  (15), or  $d\mu_H^\alpha$  (17), we write

$$P_\lambda(x_1, \dots, x_m) = \sum_{\kappa} c_{\kappa,\alpha} C_\kappa^\alpha(x_1, \dots, x_m),$$

and by applying the linearity of expectation, we obtain

$$E[P_\lambda(x_1, \dots, x_m)] = \sum_{\kappa} c_{\kappa,\alpha} E[C_\kappa^\alpha(x_1, \dots, x_m)].$$

In the univariate case the Jack polynomials are simply monomials, and we have the following (well-known) moments for the Hermite, Laguerre, and Jacobi weight functions:

$$\begin{aligned} \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} x^k e^{-x^2/2} dx &= (2k-1)!! = (-1)^{k/2} H_k(0), \\ \frac{1}{\Gamma(\gamma+1)} \int_{[0,\infty)} x^k x^\gamma e^{-x} dx &= (\gamma+1)_k = L_k^\gamma(0), \quad \text{and} \\ \frac{\Gamma(2+a+b)}{\Gamma(a+1)\Gamma(b+1)} \int_{[0,1]} x^k x^a (1-x)^b dx &= \frac{(a+1)_k \Gamma(a+b+2)}{\Gamma(a+1)\Gamma(a+b+k+2)} = J_k^{a,b}(0). \end{aligned}$$

In the above,  $k \geq 0$ .

A similar triad of formulas can be established for the multivariate case. In the Laguerre and Jacobi cases, the univariate formulas generalize easily:

$$\int_{[0,\infty)^m} C_\kappa^\alpha(x_1, \dots, x_m) d\mu_L^\alpha(x_1, \dots, x_m) = (\gamma + \frac{m-1}{\alpha} + 1)_\kappa C_\kappa^\alpha(I_m) = L_\kappa^{\alpha,\gamma}(0), \tag{24}$$

$$\begin{aligned} \int_{[0,1]^m} C_\kappa^\alpha(x_1, \dots, x_m) d\mu_J^\alpha(x_1, \dots, x_m) &= \frac{(g_1 + \frac{m-1}{\alpha} + 1)_\kappa}{(g_1 + g_2 + \frac{2}{\alpha}(m-1) + 2)_\kappa} C_\kappa^\alpha(I_m) \\ &= J_\kappa^{\alpha,g^1,g^2}(0). \end{aligned} \tag{25}$$

For a good reference for the first formula, see Baker and Forrester (1997); the second one was obtained by Kadell (1997).

For the Hermite case,

$$\int_{\mathbb{R}^n} C_{\kappa}^{\alpha}(x_1, \dots, x_m) d\mu_H^{\alpha} = (-1)^{k/2} H_{\kappa}^{\alpha}(0), \tag{26}$$

but to the best of our knowledge, no simpler closed-form formula is known. We compute the right-hand side as the 0th-order coefficient of the polynomial  $H_{\kappa}^{\alpha}(x_1, \dots, x_m)$ , using formula (22). Note that if  $\kappa$  sums to an odd integer, the above is trivially 0.

The procedures **expHjacks**, **expljacks**, and **expJjacks** compute the expected value of an expression (allowing not only for addition, but also for multiplication and powers) involving Jack polynomials (all having the same parameter  $\alpha$  as the distribution). They consist of the two steps described in the first paragraph of this section: the first one is reducing the expression to a weighted sum of Jack polynomials (if the expression is a weighted sum of Jack polynomials, this step is skipped), and the second step is replacing each Jack polynomial with its expected value, using the formulas (26), (24), and (25).

The procedures **expH**, **expl**, and **expJ** compute the expected value of an expression involving monomials (allowing for addition, multiplication, and powers), and there are three steps involved: the first is writing the expression in monomial basis, the second rewriting the result in Jack polynomial basis, and the third replacing the Jack polynomials by their expectations, using (26), (24), and (25).

**Example.** Suppose we want to compute the expected value of

$$z(\alpha, x_1, x_2, x_3) := J_{[2,1]}^{\alpha}(x_1, x_2, x_3) C_{[1,1,1]}^{\alpha}(x_1, x_2, x_3)$$

over the  $2/\alpha$ -Hermite distribution. First we have to express  $z$  as a linear combination of Jack “C” polynomials. Note that the number of variables, as well as  $\alpha$ , must be the same in the two terms of  $z$ .

First, we express the two terms in monomial basis:

$$\begin{aligned} J_{[2,1]}^{\alpha}(x_1, x_2, x_3) &= (2 + \alpha) m_{[2,1]}(x_1, x_2, x_3) + 6 m_{[1,1,1]}(x_1, x_2, x_3), \\ C_{[1,1,1]}^{\alpha}(x_1, x_2, x_3) &= \frac{6\alpha^2}{(1 + \alpha)(2 + \alpha)} m_{[1,1,1]}(x_1, x_2, x_3). \end{aligned}$$

Their product thus becomes a linear combination of sums of products of two monomials, which are in turn converted into a linear combination of monomials. Note that here we use the fact that there are three variables:

$$\begin{aligned} m_{[2,1]}(x_1, x_2, x_3) m_{[1,1,1]}(x_1, x_2, x_3) &= m_{[3,2,1]}(x_1, x_2, x_3), \quad \text{while} \\ m_{[1,1,1]}(x_1, x_2, x_3)^2 &= m_{[2,2,2]}(x_1, x_2, x_3). \end{aligned}$$

Putting it all together, in monomial basis,

$$z(\alpha, x_1, x_2, x_3) = \frac{6\alpha^2}{1 + \alpha} m_{[3,2,1]}(x_1, x_2, x_3) + \frac{36\alpha^2}{(1 + \alpha)(2 + \alpha)} m_{[2,2,2]}(x_1, x_2, x_3).$$

All that is left now is to convert from the monomial basis back to the Jack polynomial basis. We obtain that

$$z(\alpha, x_1, x_2, x_3) = \frac{1}{120} \frac{(2 + 3\alpha)(1 + 2\alpha)^2}{\alpha(1 + \alpha)} C_{[3,2,1]}^{\alpha}(x_1, x_2, x_3)$$

We are now able to finish the work:

$$E_H[z(\alpha, x_1, x_2, x_3)] = -\frac{36(\alpha - 1)(\alpha + 3)}{(1 + \alpha)(2 + \alpha)}.$$

### 3. Complexity bounds and running times

In this section we will analyze the performance of the main algorithms, which we divide into four parts:

1. algorithms that compute polynomials;
2. algorithms that evaluate integrals;
3. conversion algorithms;
4. numerical algorithms.

Our complexity bounds are upper bounds, but we believe many of them to be asymptotically correct. They work well for the numerical evaluation of the parameters involved (i.e.  $\alpha$ ,  $m$ ,  $\gamma$ ,  $g_1$ ,  $g_2$ ); symbolic evaluation of the polynomials is considerably slower. We are not aware of the existence of a good symbolic performance model for Maple, and hence it would be difficult to predict how much slower symbolic evaluation is than numerical evaluation. Once parameters are introduced (like  $m$ , the number of variables, or  $\alpha$ , the Jack parameter), the quantities to be computed become rational functions of these parameters, of degrees that can go up to the partition size  $|\kappa|$ . Storage then becomes an issue, and hence one would expect the running times for symbolic evaluation to be orders of magnitude slower than for numerical evaluation, since the coefficients we deal with must be written and stored on “slow” memory (e.g. disk space), and the “transport” time to and from “slow” memory greatly increases the overall running time.

For each algorithm we provide a complexity analysis, and we illustrate the performance in practice by providing running times for different tests (both numerical and symbolic); then we examine the running times and draw a set of conclusions.

Each time we use *N/A* for an entry in a running times table, we have done so because that particular computation has exhausted the memory available to Maple, and hence (regardless of the time it took up to that point) the computation was not finished.

The computer on which we have performed our tests is a Pentium 4 by Dell, 1.8 GHz, 512 MB; the version of Maple used for the tests is Maple 8.

The last thing worth mentioning is that Maple has an *option remember*, that is it allows for storage and recall of a quantity that was computed previously, and that MOPS is taking advantage of that.

#### 3.1. Algorithms that compute polynomials

In this category we have the algorithms that evaluate **jack**, **gbinomial**, **hermite**, **laguerre**, and **jacobi**. We analyze here **gbinomial**, though it is not a polynomial in  $(x_1, \dots, x_m)$ , because it is the main building block for **hermite**, **laguerre**, and **jacobi**, and its complexity determines their computational complexity.

Throughout this section, we will follow the notation given in Table 7.

To make estimates, we have used Ramanujan’s formula:

$$P_{[k]} \sim \frac{1}{4k\sqrt{3}} e^{\pi\sqrt{2k/3}}, \quad (27)$$

Table 7  
Notation to be used throughout Section 3

$k =  \kappa $	Size of partition $\kappa$
$s =  \sigma $	Size of partition $\sigma$
$l = \text{length}(\kappa)$	Length of partition $\kappa$
$n$	Number of variables used for computation
$D_\kappa$	Number of partitions of $k$ smaller in lexicographical ordering than $\kappa$
$P_\kappa$	Number of partitions of $k$ dominated by $\kappa$
$P_{[k]}$	Number of partitions of the number $k$ (each partition of $k$ is dominated by $[k]$ )
$U_\kappa$	Number of subpartitions of $\kappa$
$U_{\kappa,\sigma}$	Number of subpartitions of $\kappa$ which are superpartitions for $\sigma$ (this implies $\sigma$ is a subpartition of $\kappa$ )
$A_\kappa$	Number of subpartitions of $\kappa$ which sum to a number with the same parity with $k$

and the inequalities

$$A_\kappa \leq U_\kappa \leq P_{[k]}, \quad P_\kappa \leq P_{[k]}, \quad \text{and} \quad U_{\kappa,\sigma} \leq P_{[k]} \tag{28}$$

for asymptotical estimates.

1. **jack**. The algorithm uses recurrence (6), together with the ‘boundary conditions’  $c_{\kappa,\lambda} = 0$  if  $\kappa \not\preceq \lambda$  in dominance ordering, and  $c_{\kappa,\kappa} = \frac{\alpha^k k!}{c^{l(\kappa,\alpha)}}$ . The length of the recurrence is at most  $O(k_1 \binom{k+1}{2})$ , with  $k_1$  being the first entry in the partition, and the algorithm will check each of the possible partitions  $\mu$  (at most  $k_1 \binom{k+1}{2}$ ) to see if they are dominated by  $\kappa$  and dominating  $\lambda$  (this involves  $l$  additions and  $l$  comparisons). The rest of the computation has complexity  $O(k)$ .

Thus the complexity of the algorithm is  $O(k_1 k^3 P_\kappa)$ .

Using the inequalities (28), the best asymptotical upper bound that we can get for the complexity of computing a Jack polynomial is thus  $O(k^3 e^{\pi\sqrt{2k/3}})$ , which is superpolynomial.

In Table 8 we illustrate the running times for both numerical and symbolic computations. For numerical computations, we have chosen to make  $\alpha = 1$ , so that the Jack polynomials are the Schur functions. Note that we do not test the partition  $[k]$ ; for that particular partition we have a closed-form formula for the Jack polynomial, due to Stanley (1989), which has complexity  $O(k P_k) \sim O(e^{\pi\sqrt{2k/3}})$ .

Table 8  
Running times (in seconds) for the Jack polynomial computation

$k$	$\kappa$	Running time, $\alpha = 1$	Running time, $\alpha$ symbolic	Ratio
15	$\kappa = [14, 1]$	2.48	4.54	1.83
	$\kappa = [8, 7]$	1.79	3.17	1.77
	$\kappa = [3, 3, 3, 3, 3]$	0.39	0.50	1.28
20	$\kappa = [19, 1]$	16.97	30.45	1.79
	$\kappa = [10, 10]$	11.53	20.32	1.76
	$\kappa = [4, 4, 4, 4, 4]$	2.91	4.02	1.38
25	$\kappa = [24, 1]$	93.42	189.66	2.03
	$\kappa = [9, 8, 8]$	46.85	79.85	1.70
	$\kappa = [5, 5, 5, 5, 5]$	16.08	24.18	1.50
30	$\kappa = [29, 1]$	634.32	1819.65	2.86
	$\kappa = [10, 10, 10]$	214.10	418.19	1.95
	$\kappa = [6, 6, 6, 6, 6]$	73.54	113.55	1.54

**Remark 3.1.** Note that the ratio of the running times increases when the partition size increases. At  $k = 30$ , the number of partitions is 5604, and each of the monomial coefficients is a rational function of  $\alpha$ . Issues like storage and memory access become important, and influence negatively the running times. Another important factor is that in order to make things easier to store and access, not to mention easier to read and interpret, we use the procedures “simplify” and “factor”, which are relatively costly.

*Extrapolation.* Since the speed/memory of a top-of-the-line computer seems to go up by a factor of  $10^3$  every 10 years, one can predict that within a decade, using MOPS, computing  $J_{(59,1)}^\alpha$  will take about 30 minutes.

- gbinomial.** We use (8), together with the boundary conditions listed in Section 2.5 and with the contiguous binomial coefficient formula (9). From (8), it follows that computing a single contiguous binomial coefficient has complexity  $O(k)$ , and one needs to compute no more than  $l$  such coefficients per subpartition  $\tilde{\sigma}$  of  $\kappa$  which is a superpartition of  $\sigma$ .

Thus one immediately obtains the bound  $O(klU_{\kappa,\sigma})$  for the complexity of computing  $\binom{\kappa}{\sigma}$ . This is smaller than  $O(k^2U_{\kappa, [1^2]})$ .

Note that by computing  $\binom{\kappa}{\sigma}$ , one also obtains  $\binom{\kappa}{\mu}$ , for each  $\sigma \subseteq \mu \subset \kappa$ . So we have chosen for our tests to compute  $\binom{\kappa}{[1,1]}$  for different  $\kappa$ , as this yields all the binomial coefficients having  $\kappa$  as top partition (except  $\binom{\kappa}{2}$ , but that requires only an additional complexity  $O(kl)$ ).

By using the inequalities (28), we obtain an asymptotical upper bound of  $O(ke^{\pi\sqrt{2k/3}})$  for computing all the generalized binomial coefficients corresponding to partitions of  $k$ .

**Remark 3.2.** Once again, size and length of the partition increase the symbolic running times; however, note that the running times are relatively small, even for partitions of 30 (see Table 9). We believe that the generalized binomial coefficients are rational functions of  $\alpha$  which can always be factored in small-degree factors, so that they are easy to store and operate with.

Table 9  
Running times (in seconds) for the generalized binomial coefficient computation

$k$	$\kappa$	Running time, $\alpha = 1$	Running time, $\alpha$ symbolic	$U_{\kappa, [1^2]}$
15	[6, 4, 2, 2, 1]	0.22	1.12	139
	[3, 3, 3, 3, 3]	0.05	0.18	56
	[10, 5]	0.03	0.15	51
20	[6, 4, 3, 2, 2, 1, 1, 1]	1.01	6.68	418
	[4, 4, 4, 4, 4]	0.17	0.6	126
	[12, 8]	0.07	0.28	81
25	[7, 5, 4, 3, 2, 2, 1, 1]	3.41	23.37	1077
	[5, 5, 5, 5, 5]	0.41	1.67	252
	[16, 9]	0.15	0.62	125
30	[8, 6, 4, 3, 2, 2, 1, 1, 1, 1]	11.87	89.61	2619
	[6, 6, 6, 6, 6]	0.91	3.95	462
	[20, 10]	0.24	1.20	176

3. **jacobi**. To compute the Jacobi polynomials, we use the format of Section 2.7.2 and recurrence (14). One can easily see that at each step, one needs to compute at most  $l$  contiguous binomial coefficients, each of which has complexity  $O(k)$ ; in addition, one needs to compute another at most  $l$  binomial coefficients; each of these takes only  $O(l)$ , as the contiguous coefficients needed *have already been computed* at the previous step. Thus the total complexity is  $O(kl)$  (since  $l \leq k$ ) at each step, for a total of  $O(klU_{\kappa, [1^2]})$ .

Hence computing numerically the Jacobi polynomials is comparable to computing the generalized binomial coefficients  $\binom{\kappa}{[1,1]}$ ; however, the constant for the Jacobi polynomial complexity is considerably larger (our best guess sets it around 8).

The best asymptotical upper bound we can obtain using the inequalities (28) is thus once again  $O(ke^{\pi\sqrt{2k/3}})$ .

The Jacobi parameters we chose for each of the computations whose runtimes are presented in Table 10 are 0 and 1.

Table 10  
Running times (in seconds) for the Jacobi polynomial computation

$k$	$\kappa$	Running time, $\alpha = 1, m = l$	Running time, $m$ symbolic	Running time, $\alpha, m$ symbolic	$U_{\kappa}$
10	[4, 2, 2, 1, 1]	0.27	0.74	22.12	42
	[4, 3, 3]	0.11	0.35	1.88	30
	[7, 3]	0.10	0.30	1.57	26
15	[6, 4, 2, 2, 1]	1.05	11.08	N/A	139
	[3, 3, 3, 3, 3]	0.39	0.87	63.07	56
	[10, 5]	0.19	1.01	27.98	51
20	[6, 4, 3, 2, 2, 1, 1, 1]	5.94	N/A	N/A	418
	[4, 4, 4, 4, 4]	0.63	8.24	N/A	126
	[12, 8]	0.26	3.51	N/A	81
25	[7, 5, 4, 3, 2, 2, 1, 1]	18.61	N/A	N/A	1077
	[5, 5, 5, 5, 5]	1.23	N/A	N/A	252
	[16, 9]	0.45	N/A	N/A	125

**Remark 3.3.** While the running times for numerical evaluation are reasonable, they explode when a symbolic parameter is introduced. The coefficients of the polynomial are rational functions of that parameter or combination of parameters, of order up to  $k(k - 1)/2$ . We recall that there are  $U_{\kappa, [1^2]}$  of them, a potentially superpolynomial number, which explains the tremendous increase in the running time.

4. **laguerre**. We use the format given in Section 2.7.4; it is easily established that the complexity of computing the Laguerre polynomial is dominated by the cost of computing the binomial coefficients, that is  $O(klU_{\kappa, [1^2]})$ , and once again the best asymptotical upper bound we can obtain using the inequalities (28) is thus  $O(ke^{\pi\sqrt{2k/3}})$ .

The Laguerre parameter we chose for each of the computations whose runtimes are presented in Table 11 is 1.

**Remark 3.4.** For the Laguerre polynomials, even in the all-symbolic case, the computation is very easy, and the storage required is relatively small. This explains why it is possible to obtain them without much effort, in any one of the cases.

Table 11  
Running times (in seconds) for the Laguerre polynomial computation

$k$	$\kappa$	Running time, $\alpha = 1, m = l$	Running time, $m$ symbolic	Running time, $\alpha, m$ symbolic	$U_\kappa$
10	[4, 2, 2, 1, 1]	0.12	0.23	0.54	42
	[4, 3, 3]	0.07	0.14	0.31	30
	[7, 3]	0.07	0.10	0.28	26
15	[6, 4, 2, 2, 1]	0.49	0.82	2.95	139
	[3, 3, 3, 3, 3]	0.18	0.27	0.84	56
	[10, 5]	0.11	0.22	0.81	51
20	[6, 4, 3, 2, 2, 1, 1, 1]	2.26	3.37	16.08	418
	[4, 4, 4, 4, 4]	0.44	0.69	2.74	126
	[12, 8]	0.20	0.37	1.79	81
25	[7, 5, 4, 3, 2, 2, 1, 1]	7.23	11.06	67.92	1077
	[5, 5, 5, 5, 5]	0.96	1.53	8.06	252
	[16, 9]	0.32	0.69	4.21	125

5. *hermite*. We use the format given in Section 2.7.6 and recurrence (21). We only do work for those coefficients that correspond to subpartitions  $\sigma$  of  $\kappa$  such that  $|\sigma| \equiv k \pmod{2}$ . There are  $A_\kappa$  of them. For each, we compute at most  $\binom{l}{2}$  contiguous coefficients, each computed with  $O(k)$  complexity. The complexity of the rest of the computation is  $O(k)$ . Hence the total complexity is  $O(kl^2A_\kappa)$ .

**Remark 3.5.**  $A_\kappa = O(U_\kappa)$ ;  $A_\kappa \sim U_\kappa/2$ .

Hence one asymptotical upper bound that we can obtain for the complexity of computing a Hermite polynomial is  $O(k^2e^{\pi\sqrt{2k/3}})$ .

**Remark 3.6.** Note that when  $m$  is parametrized, but  $\alpha = 1$ , the computation is almost as fast as in the all-numerical case (see Table 12). That happens because the dependence on  $m$  is very simple, and it only involves Pochhammer symbols, which do not get expanded (so that the storage required is minimal). However, the dependence on  $\alpha$  is more complicated, and the rational functions obtained as coefficients are complex and hard to store. Hence the running time for the all-symbolic computation increases dramatically.

Table 12  
Running times (in seconds) for the Hermite polynomial computation

$k$	$\kappa$	Running time, $\alpha = 1, m = l$	Running time, $m$ symbolic	Running time, $\alpha, m$ symbolic	$A_\kappa$
10	[4, 2, 2, 1, 1]	0.21	0.24	0.75	22
	[4, 3, 3]	0.09	0.11	0.33	16
	[7, 3]	0.05	0.06	0.24	14
15	[6, 4, 2, 2, 1]	0.41	2.83	42.92	88
	[3, 3, 3, 3, 3]	0.13	0.17	1.83	38
	[10, 5]	0.10	0.12	1.10	30
20	[6, 4, 3, 2, 2, 1, 1, 1]	1.93	2.39	N/A	211
	[4, 4, 4, 4, 4]	0.35	0.51	N/A	66
	[12, 8]	0.18	0.25	13.49	43
25	[7, 5, 4, 3, 2, 2, 1, 1]	6.23	7.53	N/A	1077
	[5, 5, 5, 5, 5]	0.90	1.20	N/A	252
	[16, 9]	0.29	0.50	106.56	125



### 3.2. Conversion algorithms

There are five conversion algorithms, **m2jack**, **jack2jack**, **m2m**, **p2m**, and **m2p**.

1. **m2jack**. This algorithm computes and then inverts the change of basis matrix from monomials to Jack polynomials, taking advantage of the fact that the matrix is upper triangular. At each turn, the algorithm extracts the highest-order monomial remaining, computes the coefficient of the corresponding Jack polynomial, and then extracts the monomial expansion of the Jack polynomial from the current monomial expression.

Let  $\kappa$  be the highest-order monomial present in the initial expression, and let us assume that the expression is homogeneous.

Then the complexity of the computation is dominated by the complexity of computing the Jack polynomial expansion in terms of monomials for all partitions of  $k$  smaller in lexicographical ordering than  $\kappa$ .

It follows that an upper bound on the complexity is given by  $O(D_\kappa k^4 D_\kappa) = O(k^2 e^{2\pi\sqrt{2/3}\sqrt{k}})$ .

The performance in practice is exemplified below.

Partition sum	Partition	Runtime ( $\alpha = 2$ )	Runtime symbolic $\alpha$	Ratio of the two
$k = 6$	$\kappa = [6]$	0.14	0.45	0.31
$k = 7$	$\kappa = [7]$	0.29	1.04	0.27
$k = 8$	$\kappa = [8]$	0.63	2.91	0.21
$k = 9$	$\kappa = [9]$	1.21	7.49	0.16
$k = 10$	$\kappa = [10]$	2.62	20.25	0.12
$k = 11$	$\kappa = [11]$	4.77	54.75	0.08
$k = 12$	$\kappa = [12]$	8.82	186.09	0.04
$k = 15$	$\kappa = [15]$	52.65	7177.02	<0.01

2. **m2m**. The algorithm takes an expression involving products of monomial functions and writes it in monomial basis by deciding which partitions appear in the expansion and by counting the number of times they appear. Hence this algorithm is an alternative to adding the  $m$  basis as the dual of  $h$  in the **SF** package, and using the **tom** procedure afterward (though the **tom** procedure is more general than this).

We have tested **m2m** against **tom**, and we have found that on partitions where the sum-to-length ratio is high, **m2m** performs much better, while on partitions with the sum-to-length ratio is small, the tables are reversed. Hence we recommend to the user who wants to use our library, but might be working with partitions of the latter case, to also obtain and install **SF** and use it for computations.

Below is a performance comparison. The number of variables  $n$  used in **m2m**, each time, was the sum of the partition lengths (which is the smallest number of variables that requires obtaining all terms).

Input	$n$	Runtime <b>m2m</b>	Runtime <b>tom</b>	Ratio
$m[5, 2, 1, 1] \cdot m[4]$	5	0.07	0.28	0.25
$m[3, 2, 1] \cdot m[5]$	4	0.03	0.10	0.30
$m[5, 3, 2] \cdot m[4, 3] \cdot m[2]$	6	3.52	35.27	0.10
$m[7, 3, 1] \cdot m[4, 2]$	5	0.30	9.89	0.03
$m[4, 3, 2] \cdot m[6, 4]$	5	0.29	35.72	<0.01
$m[2, 2, 1] \cdot m[1, 1]$	5	0.05	0.03	1.66
$m[3, 1] \cdot m[2, 2, 1]$	5	0.12	0.05	2.40
$m[2, 1, 1, 1] \cdot m[2, 1]$	6	0.22	0.04	5.50
$m[3, 2, 1] \cdot m[2, 1, 1, 1]$	7	2.95	0.10	29.5
$m[3, 1, 1] \cdot m[2, 1] \cdot m[1, 1, 1]$	8	13.55	0.13	104.23

3. **p2m**. The algorithm expands a product of simple power-sum functions into monomial basis. This is an alternative to adding the  $m$  basis as the dual of  $h$  in the **SF** package, and calling the **tom** procedure with power-sum functions as inputs. As was the case with **m2m**, our algorithm performs much better on partitions with high sum-to-length ratio, and **tom** performs better on partitions with low sum-to-length ratio, as can be clearly seen from the performance comparison below.

Input	Runtime <b>p2m</b>	Runtime <b>tom</b>	Ratio
$p_4 \cdot p_3^2 \cdot p_2$	0.03	0.14	0.21
$p_8 \cdot p_5 \cdot p_1^3$	0.20	5.46	0.04
$p_7 \cdot p_4 \cdot p_3$	0.01	0.46	0.02
$p_5^2 \cdot p_4^3 \cdot p_2$	0.04	5.40	<0.01
$p_4 \cdot p_2^3 \cdot p_1$	0.12	0.10	1.20
$p_5 \cdot p_2^2 \cdot p_1^3$	0.96	0.17	5.64
$p_3 \cdot p_2 \cdot p_1^5$	1.97	0.04	49.25
$p_2^4 \cdot p_1^4$	16.27	0.15	108.46

4. **m2p**. The algorithm converts an expression of monomials into power-sum functions; it is an alternative to the **top** option in the **SF** package. As before, for high sum-to-length ratio, our algorithm performs better, whereas the reverse is true for low sum-to-length ratio. It is perhaps worth noting that for this case, the ratio sum-to-length has to be between 1 and 2 for a significant outperformance of our **m2p** by **top** to occur. This can be seen in the performance examples below.

Input	Runtime <b>m2p</b>	Runtime <b>top</b>	Ratio
$m[1, 1, 1, 1, 1, 1, 1, 1]$	3.61	0.04	90.25
$m[3, 2, 2, 2, 1, 1] \cdot m[2, 1, 1, 1]$	2.19	0.11	19.91
$m[2, 2, 1, 1, 1] \cdot m[2, 1]$	0.120	0.03	4.00
$m[1, 1, 1, 1] \cdot m[1, 1, 1]$	0.03	0.02	1.50
$m[4, 3, 2, 2]$	0.06	0.18	0.33
$m[10, 1]$	0.01	0.10	0.10
$m[5, 4, 3] \cdot m[3]^2$	0.02	0.23	0.08
$m[5, 4, 3, 3]$	0.08	3.21	0.02
$m[3, 3, 2, 1] \cdot m[5, 2]$	0.07	5.57	0.01

5. **jack2jack**. This algorithm takes an expression in Jack polynomials, and turns it into a linear combination of Jack polynomials, by taking each multiplicative term, expanding it in monomial basis, then using **m2m** to get rid of the resulting multiplicative factors, and finally, **m2jack** to convert the linear combination of monomials back into Jack polynomial basis.

Input	$n$	Runtime (symbolic)	Runtime ( $\alpha = 1$ )
$J[2, 1] \cdot C[1, 1, 1]$	3	0.04	0.03
$J[3, 2, 1] \cdot C[3]$	5	3.53	0.45
$C[2]^2 \cdot J[4, 2]$	5	11.44	1.17
$C[2]^2 \cdot J[3, 1]$	8	29.09	3.35
$P[3, 2] \cdot J[2, 1, 1]$	7	15.00	2.10
$P[2, 1] \cdot J[4, 2] \cdot C[2]$	5	28.95	2.07

### 3.3. Algorithms that evaluate integrals

Here we have **expHjacks**, **expLjacks**, **expJjacks**, **expH**, **expL**, and **expJ**.

These algorithms depend on the length and complexity of the input. Let  $P$  be the polynomial one wishes to analyze; one must first convert  $P$  to a linear combination of Jack polynomials, and then replace each Jack polynomial with its expectation.

*Case 1.* Suppose  $P$  is in monomial format, as an expression which involves sums and products of monomials. First we convert  $P$  to a linear combination of monomials using **m2m**, and then we convert that linear combination of monomials to a linear combination of Jack polynomials using **m2jack**.

For any term of the form  $m_{\lambda^1} m_{\lambda^2} \dots m_{\lambda^p}$ , with  $\lambda^1, \lambda^2, \dots, \lambda^p$  not necessarily distinct partitions, when we expand it in monomial basis, the largest possible number of terms is  $D_\mu$ , where  $\mu$  is the partition which results from the superposition of  $\lambda^1, \lambda^2, \dots, \lambda^p$ , i.e.  $\mu_1 = \lambda_1^1 + \lambda_1^2 + \dots + \lambda_1^p, \mu_2 = \lambda_2^1 + \lambda_2^2 + \dots + \lambda_2^p$ , etc. Let  $u = |\mu|$ .

After the expansion in monomial basis, applying **m2jack** on the resulting expression has complexity  $O(u^4 D_\mu^3) = O(u^2 e^{3\pi\sqrt{2/3}\sqrt{u}})$ .

**Remark 3.7.** This however is a very relaxed upper bound, and if we start off with  $P$  being a sum of a few ( $n$ ) monomials, the call to **m2m** is not executed, and the complexity of the call to **m2jack** is  $O(nu^4 D_\mu^2) = O(nu^2 e^{2\pi\sqrt{2/3}\sqrt{u}})$ .

As explained in Section 2.9, the first step is common to **expH**, **expL**, and **expJ**. The second step is different and its complexity is much higher for **expH** than for **expL** or **expJ**. However, as we can see from the running times in the table below, the calls to **m2m** and **m2jack** (made in the first step) are much more expensive than the substitutions, and so the overall running times are comparable.

In these examples, we consider a symbolic parameter  $a$ , a symbolic number of variables  $n$ ,  $\gamma = 1$ , and  $g_1 = g_2 = 1$ .

Input	Runtime <b>expH</b>	Runtime <b>expL</b>	Runtime <b>expJ</b>
$m[6]$	0.94	0.70	0.80
$m[3, 3, 2]$	1.98	0.85	0.96
$m[5, 2, 1]$	5.69	3.20	4.20
$m[3, 1, 1, 1] \cdot m[2]$	4.23	1.84	2.59
$m[4, 1] \cdot m[1, 1, 1]$	3.94	2.18	3.58
$m[5, 1] \cdot m[2]$	8.86	6.04	9.82
$m[3]^2 \cdot m[2]$	8.80	7.00	13.04
$m[4, 2] \cdot m[3, 1]$	39.85	35.71	68.82

*Case 2.* Suppose  $P$  is in Jack polynomial format; then we use **jack2jack** to write it as a linear combination of Jack polynomials, and finally we replace each Jack term by its expected value. The first step, as before, is common to all three procedures (**expHjacks**, **expLjacks**, **expJjacks**).

While in the case of **expHjacks** the complexity of computing the expectation is  $O(u^4 e^{2\pi\sqrt{2/3}\sqrt{u}})$ , in the cases of **expLjacks** and **expJjacks** the same complexity is only  $O(u)$ . This explains the significant differences recorded in the first three rows of the table. It is also worth noting that in the case of an odd  $u$ , the time it takes to compute the expected value of a Jack polynomial with Hermite weight is 0, as the value of the output is known in advance to be 0.

The complexity of expressing a product of Jack polynomials in a Jack polynomial basis is much higher than the computation of a single Jack polynomial expected value. This explains why, in the last few rows of the table, the entries are no longer so different in magnitude.

In the examples below, we considered a symbolic parameter  $a$ , a symbolic number of variables  $n$ ,  $\gamma = 1$ , and  $g_1 = g_2 = 1$ .

Input	Runtime <b>expHjacks</b>	Runtime <b>expLjacks</b>	Runtime <b>expJjacks</b>
$C[4, 3, 2, 1]$	0.30	0.03	0.03
$C[6, 6, 2]$	1.06	0.04	0.04
$C[7, 5, 3, 1]$	4.70	0.04	0.05
$C[10, 3, 2, 1]$	4.47	0.05	0.05
$C[3, 1] \cdot P[2, 2]$	14.75	12.75	12.93
$C[4, 2] \cdot J[1, 1]$	31.86	29.05	30.11
$J[2, 1, 1] \cdot J[4]$	76.62	81.93	80.14
$C[2, 2] \cdot J[4]$	53.79	54.30	55.07

### 3.4. Numerical algorithms

Some of the symbolic/numerical evaluation routines analyzed in the previous sections include options for polynomial evaluation on numerical values of the  $x$  variables. The routines that compute the polynomials Jack, Hermite, Laguerre, and Jacobi have options that allow for numerical values of the  $x$  variables. This makes it possible to compute quantities like  $C_{[3,2]}^3(2.53, -1.09, 7.33)$ ; this feature can be used for graphics (when one needs to plot some statistic of a random matrix, as we demonstrate in the next section).

The algorithms that we have used to implement these options have been developed and analyzed by Demmel and Koev (2003) for the Jack polynomials; to evaluate the other polynomials, we use the regular expansion in terms of Jack polynomials, then substitute the numerical values for each Jack polynomial.

## 4. Applications

We have written this library for the user who would like to do statistical computations, form or test conjectures, and explore identities. The great benefit is that all computations can be done symbolically, keeping  $\alpha$  as a parameter; the downside of symbolic computations, as we have mentioned before, is that the storage space required is very large, and computations are consequently slowed down. Our experience, however, was that on a machine that is far from top-of-the-line nowadays (see the specifications in Section 3), we have been able to increase the size of the partition enough to make and then satisfactorily test conjectures.

Below are some examples of computations that we imagine are of the type a researcher might want to use in forming conjectures, or of the type that might be useful in practice.

Some of the applications, like the computation of the moments of the trace, can be done with symbolic  $\alpha$  and  $n$  (number of variables); others, like the computation of the moments of the determinant, need an actual value for  $n$ , but allow for symbolic  $\alpha$  computations; yet others, like the level density computation, need all numerical parameters. For each computation, we have tried to indicate upper bounds for the size of the necessary numerical parameters.

1. *Moments of the determinant.* One of the many interesting problems in random matrix theory is computing the moments of the determinant of a square random matrix. If the eigenvalues are chosen to have the  $2/\alpha$ -Hermite distribution (given by the weight function  $\mu_H^\alpha$ ), the problem of computing the determinant is non-trivial. Closed-form answers are known for the cases  $\alpha = 1/2, 1$ , and  $2$  (see Andrews et al. (2003), Delannay and Le Caër (2000), Mehta and Normand (1998)); however, the general  $\alpha$  case does not have an explicit answer (except for some particular situations like in Dumitriu (2003, Chap. 8)).

Since the  $k$ th moment of the determinant's distribution is given as the integral of  $m_{[k^m]}(x_1, \dots, x_m) = C_{[k^m]}^\alpha(x_1, \dots, x_m)/C_{[k^m]}(I_m)$  over the corresponding  $2/\alpha$ -Hermite distribution, **MOPS** can be used in evaluating it for specific values of  $k$  and  $m$ .

For example, for  $k = 2$  and  $m = 5$ , the answer can be obtained by typing in

```
> factor(expHjacks(a, C[2,2,2,2,2], 5)/jackidentity(a, [2,2,2,2,2], 5));
```

and the output is

$$> \frac{a^4 + 10a^3 + 45a^2 + 80a + 89}{a^4}$$

The duality principle between  $\alpha$  and  $1/\alpha$  proved in Dumitriu (2003, Section 8.5.2) linking the expected value of the  $k$ th power of the determinant of an  $n \times n$  matrix to the expected value of the  $n$ th power of a  $k \times k$  matrix is illustrated below:

```
> factor(expHjacks(1/a, C[5,5], 2)/jackidentity(1/a, [5,5], 2));
```

with output

$$> -a \left( a^4 + 10a^3 + 45a^2 + 80a + 89 \right)$$

Note the remarkable similarity between this value and the one obtained 6 lines above.

**Remark 4.1.** In practice, we have observed that computations with  $\alpha$  symbolic and  $k \cdot m \leq 22$  can be performed relatively fast (under two minutes on the computer with specifications given at the beginning of Section 3); for  $k \cdot m > 22$  and  $\alpha$  symbolic, the amount of memory available begins to play an important role. For actual values of  $\alpha$  (for example,  $\alpha = 1$ ), the computation for  $k = 10$  and  $m = 5$  took under 40 seconds.

2. *Expectations of powers of the trace.* Consider the problem of computing the expected value of the 6th power of the trace of a Hermite (Gaussian) ensemble (here  $n$  is an arbitrary integer). This amounts to making a call to **expH**, simplifying, and expanding the answer in Taylor series for a clear format. In short, a one-line command:

```
> taylor(simplify(expH(a, m [6], n)), n);
```

with answer

$$> \frac{15a^3 - 32a^2 + 32a - 15}{a^3}n + \frac{-54a + 32a^2 + 32}{a^3}n^2 + \frac{22a - 22}{a^3}n^3 + \frac{5}{a^3}n^4.$$

**Remark 4.2.** This computation emphasizes best the power of MOPS. It is very quick (it took 0.8 seconds on the test machine; see the specifications in Section 3) and it allows for both  $\alpha$  and  $n$  to be in symbolic form. The same computation for the 12th power of the trace with  $\alpha$  and  $n$  symbolic took less than 8 minutes.

Integrals of powers of the trace are related to Catalan numbers and maps on surfaces of various genera, and are of interest to (algebraic) combinatorialists (Forrester, 2001; Goulden and Jackson, 1997).

3. *Smallest eigenvalue distributions.* One of the quantities of interest in the study of Wishart matrices<sup>4</sup> is the distribution of the smallest eigenvalue. There is an extensive literature on the subject, starting with the work of James (1964) and Constantine (1963). More recent references are Silverstein (1989) and Edelman (1991). In Dumitriu (2003), we find a closed-form answer for general  $\alpha$  and integer values of  $\gamma$ , in terms of a hypergeometric  ${}_2F_0$  function (see also (23)).

<sup>4</sup> The joint eigenvalue distribution of Wishart matrices is given by the Laguerre weight  $\mu_L^{\alpha, \gamma}$  with  $\alpha = 1$  (complex case) or  $\alpha = 2$  (real case).

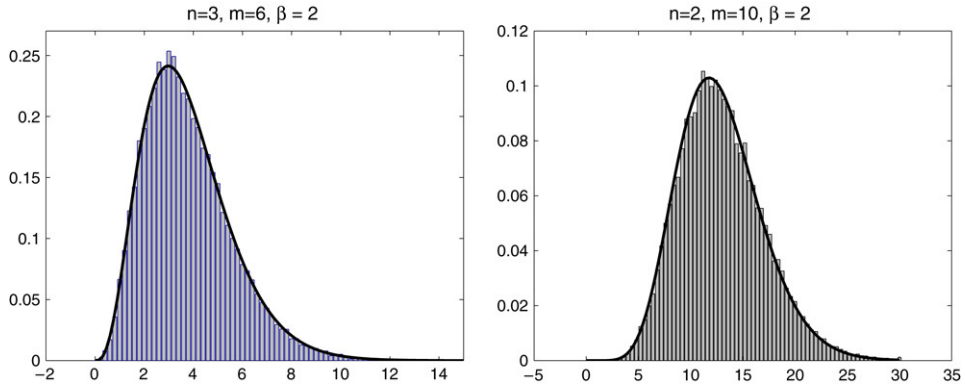


Fig. 2. Histograms of the smallest eigenvalue distribution for the complex Wishart ensembles of size (3, 6) and (2, 10) ( $\alpha = 1$ ), together with the exact distributions as given by (23).

We wrote a small Maple script calling MOPS (presented below) implementing the formula, and used it to compute the exact distribution of the smallest eigenvalue of a Wishart matrix for  $\alpha = 1$  (the complex case) for  $n = 3$ ,  $m = 6$ , and  $n = 2$ ,  $m = 10$ , which we plotted in MATLAB. We have also used a Monte Carlo simulation to plot in MATLAB histograms of the smallest eigenvalue of matrices from the corresponding Wishart ensemble, for comparison (see Fig. 2). For the histograms, we have chosen in each case 30,000 samples from the corresponding Wishart ensemble.

```

smalleig:=proc(n,k,x) local r,t,i, y,inte;
  if (n>1) then r:=[-2/x];
  end if;
  for i from 2 to (n-1) do
    r:=[op(r),-2/x];
  end do;
  t:=x-(k-n)*n * exp(-x*n/2) * ghypergeom(1, [n-k, n+1],[],r,'m');
  return simplify(t);
end proc;

scaledsmalleig:=proc(n,k,x) local inte, yy, z;
  yy :=z->smalleig(n,k,x);
  inte := integrate(yy(z), z=0..infinity);
  return(smalleig(n,k,x)/inte);
end proc;

zz:=scaledsmalleig(3,6, x);
plot(zz, x=0..10);

```

4. *Level densities.* Level density formulas are well known in terms of orthogonal polynomials for  $\alpha = 1/2, 1, 2$ . Baker and Forrester (1997) have computed these densities in terms of a multivariate Hermite polynomial for  $\beta = 2/\alpha$  an even integer (i.e.  $\alpha$  is the inverse of an integer). We have found an equivalent formulation for the level density of the  $n \times n$  Hermite ensemble for which  $\alpha$  is the inverse of an integer (equivalently,  $\beta = 2/\alpha$  is an even integer).

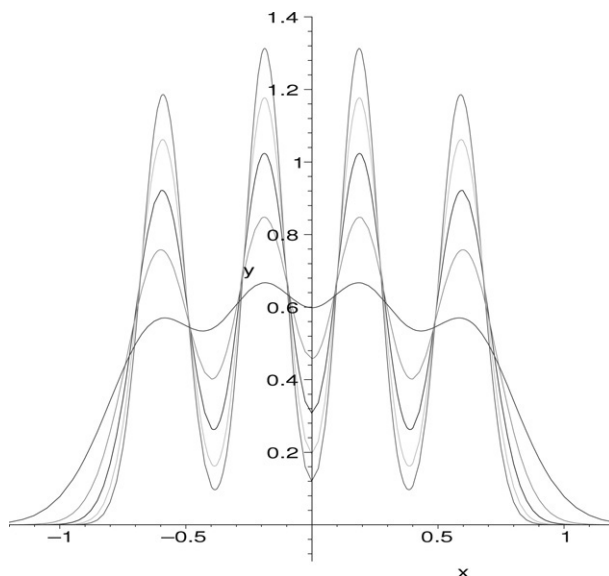


Fig. 3. Level densities for  $n = 4$ ,  $\alpha = 1, 1/2, 1/3, 1/4, 1/5$ ; “bumps” increase as  $\alpha$  decreases.

This formula is presented below:

$$\rho_n(x) = \frac{1}{\sqrt{2\pi}} (-1)^{n/\alpha} \frac{\Gamma\left(1 + \frac{1}{\alpha}\right)}{\Gamma\left(1 + \frac{n}{\alpha}\right)} e^{-x^2/2} H_{[(2/\alpha)^{n-1}]}^\alpha(x I_n),$$

where the partition  $[(2/\alpha)^{n-1}]$  is the partition that consists of  $2/\alpha$  repeated  $n - 1$  times.

To compute the Hermite polynomial, we used the formula (22), and in order to get all the eigenvalues roughly in  $[-1, 1]$  we scaled both the variable and the density function by  $\sqrt{2n\beta} \equiv \sqrt{4n/\alpha}$  (see the script).

We have used the script below to produce Fig. 3, which is an exact plot of the level densities for  $n = 4$ , and  $\beta = 2, 4, 6, 8, 10$  (equivalently,  $\alpha = 1, 1/2, 1/3, 1/4, 1/5$ ).

```

leveldens:=proc(a,k:list, n, x) option remember;
  local s,u,ut,ul,ks,ss,j,i,sp,result,t,t1,r,jp,ul1,c,bbb;
  if(not('MOPS/parvalid'(k))) then return;
  end if;
  result:=0; ks:=sum(k[i],i=1..nops(k)); sp:='MOPS/subPar'(k);
## we compute the Hermite polynomial evaluated at  $x I_n$ , using formula (22)
  for s in sp do
    ss:=0; c:=0; ss:=sum(s[i],i=1..nops(s));
    if not((ss mod 2) = (ks mod 2)) then next;
    end if;
    for j from ss to (ks+ss)/2 do
      jp:='MOPS/Par'(j);
      ul1:=(convert(jp,set) intersect convert(sp,set));
      ul:=[];
      for ut in ul1 do
        if 'MOPS/subPar?'(s,ut) then ul:=[op(ul),ut];
        end if;
      end do;
      t:=0;
      for u in ul do
        t1:='MOPS/GSFact'(a,r+(n+a-1)/a,k)/'MOPS/GSFact'(a,r+(n+a-1)/a,u);

```

```

t:=t+'MOPS/GBC'(a,k,u)*MOPS/GBC'(a,u,s)*coeff(t1,r,(ks+ss)/2-j);
end do;
c:=c+t*(-1)^j;
end do;
bbb:=factor(c*(-1)^(ss/2)*x^(ss));
result:=result+bbb;
end do;
result:= result*(-1)^(ks)*(-1)^(ks/2) * exp(-x^2/2) * 1/sqrt(2*Pi);
result:=result * factor(GAMMA(1+1/a)/GAMMA(1+n/a));
end proc;
## we scale both the variable and the density function by sqrt(2n*beta)
z:=(x,b)->sqrt(2*4*b)*leveldens(2/b, [b,b,b], 4, x*sqrt(2*4*b));

plot(z(x,2), z(x,4), z(x,6), z(x,8), z(x,10), x=-1.2..1.2, y=-.1..1.4);

```

For illustration purposes, here is the exact (scaled) density for  $\alpha = 1/4$  and  $n = 5$ , plotted above:

$$\frac{\sqrt{10}e^{-40x^2}}{50685458503680000\sqrt{\pi}} \times (28147497671065600000000000000000x^{32} - 28147497671065600000000000000000x^{30} + 172051579514388480000000000000000x^{28} - 6963866845682073600000000000000x^{26} + 19434060435475660800000000000000x^{24} - 366252408453464064000000000000x^{22} + 47400557017772851200000000000x^{20} - 6581219726721024000000000000x^{18} + 162266873453346816000000000x^{16} - 31084533121233715200000000x^{14} + 2673909486122434560000000x^{12} - 136819200341311488000000x^{10} + 29341248756019200000000x^8 - 1130060455927603200000x^6 + 67489799891754240000x^4 - 2060099901411552000x^2 + 32632929952848225).$$

5. *Conjectures.* We present here a conjecture that we formulated with the help of MOPS. This conjecture was proved later by Richard Stanley.

**Conjecture 4.3.** *Let  $k$  be an integer,  $\alpha$  a positive real, and consider the representation of the monomial function*

$$m_{[k]} = \sum_{\lambda \vdash k} f_{\lambda,\alpha} C_{\lambda}^{\alpha}.$$

Then for all  $\lambda$

$$f_{\lambda,\alpha} = \frac{1}{n(\lambda)} \prod_{i=1}^{\text{length}(\lambda)} \binom{i-1}{\lambda_i}_{\alpha},$$

where  $n(\lambda)$  is an integer which does not depend on  $\alpha$ .

**Acknowledgements**

We would like to especially thank Plamen Koev for many fruitful discussions, suggestions, and brainstorming sessions; we would also like to thank Eric Rains and John Stembridge for discussions that made this package better and Richard Stanley for beautiful mathematical insights.

Some of the work was conducted while Ioana Dumitriu was a Miller Research Fellow, sponsored by the Miller Institute for Basic Research in Sciences at U.C. Berkeley.

Finally, the authors would like to thank the National Science Foundation (DMS-0411962).



The first author acknowledges support of the Miller Institute for Basic Research in Sciences. The second author acknowledges the support of NSF under grant number DMS-0411962.

## References

- Andrews, G.E., Jackson, D.M., Goulden, I.P., 2003. Determinants of random matrices and Jack polynomials of rectangular shape. *Studies Appl. Math.* 110 (4), 377–390.
- Baker, T., Forrester, P., 1997. The Calogero–Sutherland model and generalized classical polynomials. *Comm. Math. Phys.* 188, 175–216.
- Chikuse, Y., 1992. Properties of Hermite and Laguerre polynomials in matrix argument and their applications. *Linear Algebra Appl.* 176, 237–260.
- Constantine, A.G., 1963. Some noncentral distribution problems in multivariate analysis. *Ann. Math. Statist.* 34, 1270–1285.
- Delannay, R., Le Caër, G., 2000. Distribution of the determinant of a random real-symmetric matrix from the Gaussian orthogonal ensemble. *Phys. Rev. E* 62, 1526–1536.
- Demmel, J., Koev, P., 2003. Efficient and accurate evaluation of Schur and Jack polynomials. Preprint.
- Van Diejen, J.F., 1997. Confluent hypergeometric orthogonal polynomials related to the rational quantum Calogero system with harmonic confinement. *Comm. Math. Phys.* 188, 467–497.
- Van Diejen, J.F., Lapointe, L., Morse, J., 2004. Determinantal construction of orthogonal polynomials associated with root systems. *Composition Math.* 140, 225–273.
- Dumitriu, I., 2003. Eigenvalue statistics for the beta-ensembles. Ph.D. Thesis, Massachusetts Institute of Technology.
- Edelman, A., 1991. The distribution and moments of the smallest eigenvalue of a random matrix of Wishart type. *Lin. Alg. Appl.* 159, 55–80.
- Forrester, P., 2001. Random matrices. Preprint.
- Foulkes, H.O., 1974. A survey of some combinatorial aspects of symmetric functions. In: *Permutations*. Gauthier-Villars, Paris.
- Goulden, I., Jackson, D.M., 1997. Maps in locally orientable surfaces and integrals over real symmetric matrices. *Canadian J. Math.* 49, 865–882.
- Hua, L.K., 1963. Harmonic Analysis of functions of several complex variables in the classical domains. *Transl. Math. Monogr.* 6.
- Jack, H., 1970. A class of symmetric polynomials with a parameter. *Proc. R. Soc. Edinburgh* 69, 1–18.
- Jackson, D.M., 2003. Personal communication, April.
- James, A.T., 1960. The distribution of the latent roots of the covariance matrix. *Ann. Math. Stat.* 31, 151–158.
- James, A.T., 1964. Distributions of matrix variates and latent roots derived from normal samples. *Ann. Math. Stat.* 35, 475–501.
- James, A.T., 1968. Calculation of the zonal polynomial coefficients by use of the Laplace–Beltrami operator. *Ann. Math. Stat.* 39, 1711–1718.
- James, A.T., 1975. Special functions of matrix and single argument in Statistics. In: Askey, R.A. (Ed.), *Theory and Application of Special Functions*. Academic Press, New York, pp. 497–520.
- Kadell, K., 1997. The Selberg–Jack polynomials. *Adv. Math.* 130, 33–102.
- Kaneko, J., 1993. Selberg integrals and hypergeometric functions associated with Jack polynomials. *SIAM J. Math. Anal.* 24, 1086–1110.
- Knop, F., Sahi, S., 1997. A recursion and a combinatorial formula for the Jack polynomials. *Invent. Math.* 128, 9–22.
- Krishnaiah, P.R., Chang, T.C., 1971. On the exact distribution of the smallest root of the Wishart matrix using zonal polynomials. *Ann. I. Math. Stat.* 23, 293–295.
- Lapointe, L., Lascoux, A., Morse, J., 2000. Determinantal expression and recursion for Jack polynomials. *Electron. J. Combin.* 7.
- Lapointe, L., Vinet, L., 1995. A Rodrigues formula for the Jack Polynomials and the Macdonald–Stanley conjecture. *IMRN* 9, 419–424.
- Lasalle, M., 1991a. Polynômes de hermite généralisés. *C. R. Acad. Sci. Paris, Sér. I* 313, 579–582.
- Lasalle, M., 1991b. Polynômes de jacobi généralisés. *C. R. Acad. Sci. Paris, Sér. I* 312, 425–428.
- Lasalle, M., 1991c. Polynômes de laguerre généralisés. *C. R. Acad. Sci. Paris, Sér. I* 312, 725–728.
- Macdonald, I.G., 1995. *Symmetric Functions and Hall Polynomials*. Oxford University Press Inc., New York.
- Mehta, M.L., Normand, J.-M., 1998. Probability density of the determinant of a random hermitian matrix. *J. Phys. A* 31, 5377–5391.
- Muirhead, R.J., 1982. *Aspects of Multivariate Statistical Theory*. John Wiley & Sons, New York.

- Okounkov, A., Olshanski, G., 1997. Shifted Jack polynomials, binomial formula, and applications. *Math. Res. Lett.* 4, 69–78.
- Silverstein, J.W., 1989. On the weak limit of the largest eigenvalue of a large dimensional sample covariance matrix. *J. Multivariate Anal.* 30, 307–311.
- Stanley, R.P., 1989. Some combinatorial properties of Jack symmetric functions. *Adv. Math.* 77, 76–115.
- Stembridge, J., 2007. Symmetric Functions Maple package. Downloadable from: <http://www.math.lsa.umich.edu/~jrs/maple.html#SF>.
- Zabrocki, M., 2007. List of maple functions for computing Macdonald polynomials. Webpage <http://garsia.math.yorku.ca/MPWP/maplefuncs.html>.
- Zeilberger, D., 2007. LUC, a Maple package. Downloadable from: <http://www.math.rutgers.edu/~zeilberg/tokhniot/LUC>.