MOPS: Multivariate Orthogonal Polynomials (symbolically)

Ioana Dumitriu, Alan Edelman, and Gene Shuman

March 26, 2004

Contents

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.

We believe that the modern day extension 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 being missed. Algorithms for their computation have not been studied systematically, and software suitable for scientific computing hardly exists. At this time there are no handbooks, no entries in Mathworld[1], no collections of software, and no collection of applications.

Development of such software may thus be seen as a whole area of research ripe for study. This paper might be thought of as a first step in this direction; undoubtedly better software will emerge in time.

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

$$< f, g >_w \;=\; \int_I f(x)g(x)w(x)dx$$

and the sequence of polynomials $p_0(x), p_1(x), p_2(x), \ldots$, such that $p_k(x)$ has degree k, and such that $< p_i, p_j >_w = 0$ 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 β (or a temperature factor $\alpha = 2/\beta$) which is not seen in the univariate case:

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

We define multivariate orthogonal polynomials $p_\kappa^\alpha(x_1, \ldots, x_n)$ with respect to the weight $W(x_1, \ldots, x_n)$. The polynomials are symmetric: they take the same value for

[1]URL http://mathworld.wolfram.com/

any permutation of the n coordinates x_i, and they satisfy

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

where κ represents the "multivariate degrees" of p_κ^α (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 in Macdonald's book [26, page 383].

Tables 1, 2, 3, and 4 give the coefficients of the Jack, Hermite, Laguerre, and Jacobi 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 3 for the Jack and 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.[2],

```
> jack(a, [2], 'J');
> hermite(a, [1, 1, 1], n, 'C');
> laguerre(a, [1, 1], g, n, 'C');
> jacobi(a, [1], g1, g2, n, 'C');
```

Table 1: Coefficients of the Jack "J" polynomial expressed in monomial basis

$k=1$	$m_{[1]}$
$J_{[1]}^\alpha$	1

$k=2$	$m_{[2]}$	$m_{[1,1]}$
$J_{[2]}^\alpha$	$1+\alpha$	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

1.2 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 α, and some of them are more famous

[2]Note the use of a for α and g for γ in the calls.

Table 2: Coefficients of the Hermite polynomial expressed in Jack "C" polynomial basis

$k=1$	$C^{\alpha}_{[1]}$
$H^{\alpha}_{[1]}$	1

$k=2$	$C^{\alpha}_{[2]}$	$C^{\alpha}_{[1,1]}$	$C^{\alpha}_{[1]}$
$H^{\alpha}_{[2]}$	1	0	$-\frac{n(n+\alpha)}{\alpha}$
$H^{\alpha}_{[1,1]}$	0	1	$\frac{n(n-1)}{1+\alpha}$

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

Table 3: Coefficients of the Laguerre polynomial expressed in Jack "C" polynomial basis

$k=1$	$C^{\alpha}_{[1]}$	$1=C^{\alpha}_{[]}$
$L^{\alpha,\gamma}_{[1]}$	-1	$\frac{(\gamma\alpha+n+\alpha-1)n}{\alpha}$

$k=2$	$C^{\alpha}_{[2]}$	$C^{\alpha}_{[1,1]}$	$C^{\alpha}_{[1]}$	$1=C^{\alpha}_{[]}$
$L^{\alpha,\gamma}_{[2]}$	1	0	$\frac{2(\gamma\alpha+n+2\alpha-1)(n+\alpha)}{\alpha(1+\alpha)}$	$\frac{(\gamma\alpha+n+\alpha-1)(\gamma\alpha+n+2\alpha-1)n(n+\alpha)}{\alpha^2(1+\alpha)}$
$L^{\alpha,\gamma}_{[1,1]}$	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 Jack "C" polynomial basis

$k=1$	$C^{\alpha}_{[1]}$	$1=C^{\alpha}_{[]}$
$J^{\alpha,g_1,g_2}_{[1]}$	-1	$\frac{(g_1\alpha+n+\alpha-1)n}{g_1\alpha+g_2\alpha+2n-2+2\alpha}$

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

than others. There are three values of α 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 [15] discovered the zonal polynomials in 1960.

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

Inspired by the work of James [15] and Hua [12], in his own attempt to evaluate

(2), Jack was lead to define the polynomials eventually associated with his name [13]. More explicitly, he defined a new one-parameter (α) class of polynomials, which for $\alpha = 1$ he proved were the Schur functions, and for $\alpha = 2$ he conjectured to be the zonal polynomials (and proved it in a very special case).

He consequently generalized the α parameter to any real non-zero number, and noted that for $\alpha = -1$ he obtained yet another special class of functions, which he called the "augmented" monomial symmetric functions. Later it was noted that the orthogonalizing inner product was positive definite only if $\alpha > 0$.

During the next decade, the study of Jack polynomials intensified; Macdonald [26, page 387] points out that in 1974, H.O.Foulkes [10] raised the question of finding combinatorial interpretations for the Jack polynomials. This question was satisfactorily answered in 1997 by Knop and Sahi [21].

In the late '80s, the Jack polynomials were the subject of investigation in Macdonald's book [26] and Stanley's paper [31]; these two authors generalized many 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 researchers who have made significant contributions in this area.

James [16] 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 [28]).

In a survey paper, James defined and described multivariate Laguerre, Hermite and Jacobi polynomials for $\alpha = 1$ [18]. Chikuse [3] studied more extensively the multivariate Hermite polynomials for $\alpha = 1$.

In the early '90s, Kaneko [20] studied the general α 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 [29] considered shifted Jack polynomials for all α, and proved that they were the same as the generalized binomial coefficients.

Kadell [19] 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 [23, 24, 25], considered all three types of general α multivariate polynomials, and among many other things computed generating functions for them.

The last results that we mention here are those of Forrester and Baker [2], who studied in detail the multivariate, general α Hermite and Laguerre polynomials, in connection with the $2/\alpha$-Hermite and Laguerre ensembles (some of their work built on Lasalle [23, 25]). For a good reference on multivariate generalizations of many of the univariate properties of the Hermite and Laguerre ensembles, see [9].

2 Multivariate Functions: Definitions, Properties, and Algorithms

2.1 Partitions and Symmetric Polynomials

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

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

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 λ.*

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 $\lambda \preceq \kappa$ (or, equivalently, that κ "dominates" λ) if, given $m = \max\{length(\kappa), length(\lambda)\}$,*

$$\begin{aligned} \sum_{i=1}^{j} \lambda_i &\leq \sum_{i=1}^{j} \kappa_i, \qquad \forall\, j < m \ , \qquad \textit{and} \\ \sum_{i=1}^{m} \lambda_i &= \sum_{i=1}^{m} \kappa_i \ . \end{aligned}$$

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

Remark 2.6. *Note that we compare two partitions only if they sum to the same integer. Also note that even with this constraint, $\preceq$ 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, $x_1, \ldots, x_m$, is a polynomial which is invariant under every permutation of $x_1, \ldots, x_m$.*

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 terms 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	$e_{\lambda_1} = \sum_{j_1 < j_2 < \ldots < j_{\lambda_1}} x_{j_1} \ldots x_{j_{\lambda_1}}$	$e_\lambda = \prod_{i=1}^{l} e_{\lambda_i}$
complete homogeneous functions	$h_{\lambda_1} = \sum_{j_1 \le j_2 \le \ldots \le j_{\lambda_1}} x_{j_1} \ldots x_{j_{\lambda_1}}$	$h_\lambda = \prod_{i=1}^{l} h_{\lambda_i}$

Another important basis is given by the monomial functions m,

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

here S_λ is the set of permutations giving distinct terms in the sum; λ 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[3].

Monomials seem to be the basis of choice for most people working in statistics or engineering. Combinatorialists often 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 α; the real and complex versions are familiar from the literature, and the arbitrary $\alpha > 0$ case represents an immediate extension:

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

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

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

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

becomes the generalized shifted factorial. We call

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

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 [26]) The Jack polynomials P_λ^α 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 λ. 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, $I_m = (1,1,1,\ldots,1)$, where the number of variables is m.

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

Definition 2.10. *(following Muirhead, [28]) The Jack polynomial C_κ^α is the only homogeneous polynomial eigenfunction of the following Laplace-Beltrami-type operator*

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

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

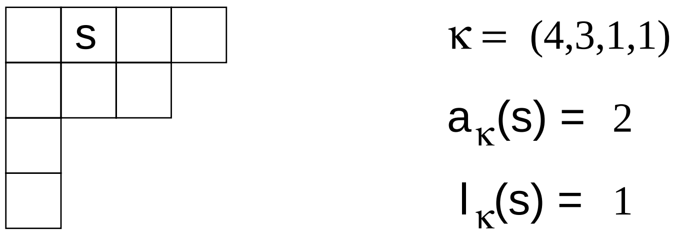
$$\sum_{\kappa \ \vdash \ k, \ \ l(\kappa) \le m} C_\kappa^\alpha(x_1, x_2, \ldots, x_m) = (x_1 + x_2 + \ldots x_m)^k \ .$$

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. *David M. Jackson [14] pointed out that the D^* operator also appears in algebraic geometry, for example in the context of ramified covers.*

Definition 2.13. *Given the diagram of a partition κ (see Figure 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 $h^\kappa_*(s) = l_\kappa(s) + 1 + \alpha a_\kappa(s)$ (the "lower hook length").*

Figure 1: The Arm-length and the Leg-length.



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

Definition 2.14. *Let*

$$\begin{aligned} c(\kappa,\alpha) &= \prod_{s\in\kappa} h^*_\kappa(s)\ , \\ c'(\alpha,\kappa) &= \prod_{s\in\kappa} h^\kappa_*(s)\ , \\ j_\kappa &= c(\alpha,\kappa)\ c'(\alpha,\kappa)\ , \end{aligned}$$

where h^*_κ and h^κ_* 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 [17], Forrester and Baker [2], and Stanley [31].

Table 5: Values of the different normalizations of Jack polynomials of partition κ and parameter α at I_m.

Normalization	Basic Property	Value at $I_m = (1,1,1,\ldots,1)$
C	sums to $(x_1 + x_2 + \ldots + x_n)^k$	$C^\alpha_\kappa(I_m) = \frac{\alpha^{2k}k!}{j_\kappa}\left(\frac{m}{\alpha}\right)_\kappa$
J	has trailing coefficient $n!$	$J^\alpha_\kappa(I_m) = \alpha^k\left(\frac{m}{\alpha}\right)_\kappa$
P	is monic	$P^\alpha_\kappa(I_m) = \frac{\alpha^k}{c(\alpha,\kappa)}\left(\frac{m}{\alpha}\right)_\kappa$

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

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

$$C_\kappa^\alpha(x_1,x_2,\ldots,x_m) = \sum_{\lambda\le\kappa} c_{\kappa,\mu}^\alpha m_\lambda(x_1,x_2,\ldots,x_m) \ ,$$

where λ and κ 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; $m_\lambda(x_1,x_2,\ldots,x_m)$ is the monomial function corresponding to λ.

Note that as a consequence of the above, if $l(\kappa) > m$, $C_\kappa^\alpha(x_1,x_2,\ldots,x_m) = 0$ ("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

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

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

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

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

In fact we can do better, using two propositions found in Macdonald's book [26, (10.13), (10.15)]. Roughly the content of the two propositions is that the Jack polynomials, in "P" normalization, can be written as

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

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

Thus it follows that the recurrence can be improved to

$$c_{\kappa,\lambda}^\alpha = \frac{\frac{2}{\alpha}}{\rho_\kappa^\alpha - \rho_\lambda^\alpha} \sum_{\lambda \prec \mu \preceq \kappa} \Big((l_i + t) - (l_j - t) \Big) c_{\kappa,\mu}^\alpha \ , \tag{7}$$

where $\lambda = (l_1, \ldots, l_i, \ldots, l_j, \ldots, l_m)$, $\mu = (l_1, \ldots, l_i + t, \ldots, l_j - t, \ldots, l_m)$, and μ has the property that, when properly reordered, it is between λ (strictly) and κ 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, one has to account for the possibility that $\rho_\kappa^\alpha = \rho_\lambda^\alpha$ for some λ different from κ; what can one do in that case?

Fortunately, this never happens. We first need the following well known Proposition.

Proposition 2.15. *The dominance ordering is a lattice on the set of partitions of a given number. In particular, between any partitions κ and λ such that $\kappa \succ \lambda$, there exists a "path" on this lattice, $\sigma^0 = \kappa \succ \sigma^1 \succ \ldots \succ \sigma^t = \lambda$, such that σ^{i+1} differs from σ^i in the following way: there exists $i_1 < i_2$ such that σ^{i+1} and σ^i agree in all places but i_1 and i_2, $(\sigma^{i+1})_{i_1} = (\sigma^i)_{i_1} - 1$, and $(\sigma^{i+1})_{i_2} = (\sigma^i)_{i_2} + 1$.*

Now we can prove that we never divide by 0 in computing Recurrence 7.

Lemma 2.16. *If $\lambda \prec \kappa$, then $\rho_\lambda^\alpha \neq \rho_\kappa^\alpha$, for all $\alpha > 0$.*

Proof. Let $\lambda \prec \kappa$ be two partitions, let $m = \max\{length(\kappa), length(\lambda)\}$, 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-negative (as an immediate consequence of the strict ordering).

We show that the left-hand side is positive by induction. For that we will use Proposition 2.15, 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 κ and λ 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 2.15 ensures thus that once $c^{\alpha}_{\kappa\kappa}$ is determined, every other non-zero coefficient is uniquely determined.

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

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

Remark 2.17. *It is worth mentioning that, from the recurrence 7, by letting $\alpha \to \infty$, the coefficient $c^{\alpha}_{\kappa,\lambda}$ goes to 0 faster than $c^{\alpha}_{\kappa,\kappa}$, 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 $\prod_{1 \le i < j \le n} |x_i - x_j|^{2/\alpha}$ (see for example 1) disappears and the variables separate.*

2.5 Multivariate binomial coefficients

Many algebraic quantities (and the identities 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 2.18. *We define the multivariate (or generalized) binomial coefficients $\binom{\kappa}{\sigma}$ as*

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

where $\sigma \subset \kappa$ means that $\sigma_i \le \kappa_i$ for all i.

The generalized binomial coefficients depend on α, 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,

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

where $\kappa_{(i)} = (k_1, \ldots, k_i - 1, \ldots, k_m)$. The above are true for all κ and α, and σ 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 [20], 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

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

Knowing the contiguous coefficients allows for computing all the generalized binomial coefficients.

Remark 2.19. *The generalized binomial coefficients are independent of the number of variables. They are rational functions of α.*

2.7 Multivariate Orthogonal Polynomials

2.7.1 Jacobi Polynomials

These polynomials represent the Gram-Schmidt orthogonalization of the Jack polynomials C_λ^α with respect to the Jacobi weight function

$$\begin{aligned} d\mu_J^\alpha(x_1, \dots, x_m) &= \frac{\Gamma_m^\alpha\left(g_1 + g_2 + \frac{2}{\alpha}(m-1) + 2\right)}{\Gamma_m^\alpha\left(g_1 + \frac{m-1}{\alpha} + 1\right)} \prod_{i=1}^m \left[x_i^{g_1} (1 - x_i)^{g_2} \right] \times & (10) \\ &\times \prod_{i<j} |x_i - x_j|^{2/\alpha} dx_1 \dots dx_m , & (11) \end{aligned}$$

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

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

Define

$$\begin{aligned} \delta^* &= \sum_i x_i \frac{d^2}{dx_i^2} + \frac{2}{\alpha} \sum_{i \neq j} \frac{x_i}{x_i - x_j} \frac{d}{dx_i} \\ E &= \sum_i x_i \frac{d}{dx_i} \\ \epsilon &= \sum_i \frac{d}{dx_i} ; \end{aligned}$$

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 Jack polynomial basis as

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

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

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

with the previous notation for ρ_κ^α and $\sigma^{(i)}$. The question is again whether the denominator is always nonzero.

Proposition 2.20. *Under these assumptions,* $(g_2 + g_1 + \frac{2}{\alpha}(m-1) + 2)(k - s) + \rho_\kappa^\alpha - \rho_\sigma^\alpha$ *is never* 0.

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

Letting $c_{\kappa,\kappa}^\alpha = 1$ for all κ and α 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

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

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

This weight function can be obtained from the Jacobi weight function (10) of the previous subsection by substituting $(g_1 + g_2 + \frac{2}{\alpha}(m-1) + 2)^{-1}(x_1, \ldots, x_m)$ for $(x_1, \ldots, x_m)$ and then taking the limit as $g_2 \to \infty$. 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 Jack polynomial basis, which depends on the generalized binomial coefficients:

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

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

2.7.5 Hermite Polynomials

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

$$\begin{aligned} d\mu_H^\alpha(x_1,\ldots,x_m) &= 2^{-m/2}\,\pi^{m(m-1)/\alpha - m/2}\,\frac{(\Gamma(1+\frac{1}{\alpha}))^m}{\Gamma_m^\alpha(1+\frac{m}{\alpha})} \times & (17)\\ &\times \quad e^{-\sum_{i=1}^m x_i^2/2}\prod_{i\neq j}|x_i-x_j|^{2/\alpha}dx_1\ldots dx_m \ , & (18)\end{aligned}$$

on $\mathbb{R}^m$.

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

Under this limiting process, the differential operator becomes

$$\delta^{**} - E \ , \qquad (19)$$

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 2.21. *Similarly,*

$$\lim_{\gamma\to\infty}\gamma^{-k/2}L_\kappa^{\alpha,\gamma}(\gamma+\sqrt{\gamma}x_1,\gamma+\sqrt{\gamma}x_2,\ldots,\gamma+\sqrt{\gamma}x_m) = (-1)^k H_\kappa^\alpha(x_1,\ldots,x_m) \ . \qquad (20)$$

2.7.6 Algorithm used to compute the Hermite Polynomials

Using the corresponding Hermite differential operator (19), we obtain the following recurrence for the coefficients of the polynomial. Let

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

then

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

In the above, $i < j$ take on all admissible values, and we choose $c^{\alpha}_{\kappa,\kappa} = C^{\alpha}_{\kappa}(I_m)$.

Alternatively, we can obtain the coefficients directly through the limiting process described in Remark 2.21:

$$\left[C^{\alpha}_{\sigma}(X)\right] H^{\alpha}_{\kappa}(X) = (-1)^k \frac{C^{\alpha}_{\kappa}(I_m)}{C^{\alpha}_{\sigma}(I_m)} \sum_{j=s}^{\frac{k+s}{2}} (-1)^{k-j} \sum_{\sigma \subseteq \mu \subseteq \kappa; \mu \vdash j} \binom{\kappa}{\mu} \binom{\mu}{\sigma} \left[r^{\frac{k+s}{2}-j}\right] F(r, \alpha, m, \kappa, \sigma) \quad ,(22)$$

where

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

We use the above formula to calculate a single coefficient, $c^{\alpha}_{\kappa,[]}$, for reasons of smaller computational complexity (in computing integrals with respect to the Hermite weight function; see Section 4.3).

Note that if $\sigma \not\subseteq \kappa$ or $k \neq s \pmod 2$, then the above 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 [9].

Definition 2.22. *We define the hypergeometric function ${}_pF^{\alpha}_q$ of parameters $a_1, \ldots, a_p$, respectively $b_1, \ldots, b_q$ and of variables $(x_1, \ldots, x_m)$ by*

$${}_pF^{\alpha}_q(a_1, \ldots, a_p; b_1, \ldots, b_q; x_1, \ldots, x_m) = \sum_{k=0}^{\infty} \sum_{\kappa \vdash k} \frac{(a_1)_{\kappa} \ldots (a_p)_{\kappa}}{k!\ (b_1)_{\kappa} \ldots (b_q)_{\kappa}} C^{\alpha}_{\kappa}(x_1, \ldots, x_m) .$$

Note that this is a formal definition; additional conditions on the variables a_i and b_j are needed in order for the above series to be well defined or have a non-trivial convergence radius[4].

This definition of a hypergeometric function assumes an argument $(x_1, \ldots, x_m) \in \mathbb{R}^m$; similarly one can extend the definition to hypergeometric functions of arguments in $(x_1, \ldots, x_m; y_1, \ldots, y_m; \ldots) \in \mathbb{R}^m \times \mathbb{R}^m \times \ldots$ by inserting an additional $C^{\alpha}_{\kappa}(y_1, \ldots y_m)/C^{\alpha}_{\kappa}(1, \ldots, 1)$ 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 [22] have 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

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

[4]In general, hypergeometrics may not be defined everywhere on $\mathbb{R}^m$; for example, for $m = 1$, ${}_1F_0(a, x)$ has a convergence radius of $1/a$. One notable exception is made 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.

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 [7] we extend this to any α and any integer $\gamma = p$. We obtain that for this case the density of the smallest eigenvalue is proportional to

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

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

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

The above is a corollary of a stronger theorem proved by Constantine [4], and it can also be found in Muirhead [28, page 421].

This result generalizes to any α and γ (as noted in [7]) to

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

3 Software

3.1 The model for MOPS

We have initially chosen as a model for **MOPS** the Symmetric Functions (**SF**) package by John Stembridge[5]. Our library is compatible with **SF**, and our procedures **m2p**, **p2m**, and **m2m** are designed to complement the **SF** procedures **tom** and **top** (for a comparison, see Section 4). Though in time our vision of **MOPS** has changed, we are grateful to John Stembridge for his ground-breaking work.

3.2 System requirements and installation guide

Our library was initially developed for Maple 7, and later for Maple 8. The version developed for Maple 8 is also compatible with Maple 9.

We have developed **MOPS** on various Unix machines and one Windows XP machine; the same version of the library is compatible with both operating systems. Below we provide an installation guide, based on downloading the MOPS.zip file from URL http://www.math.berkeley.edu/∼dumitriu/MOPS.zip .

For Unix users: in your home directory (e.g. '/home/usr/vis/joesmith/') create a directory named MOPS, and place the file MOPS.zip into the new MOPS directory. Unzip the file using the command

```
unzip MOPS.zip
```

[5]The **SF** package can be found at the URL http://www.math.lsa.umich.edu/∼jrs/maple.html#SF.

This should create 5 files in your MOPS directory: maple.lib, maple.rep, maple.hdb, maple.ind, and MOPsArchive.mws. The last file contains an archive of all the procedures we wrote for **MOPS**.

Return to your home directory (e.g. '/home/usr/vis/joesmith/'), and create a .mapleinit file; write in

$$\begin{aligned} \text{new\_libname} &:= \text{`/home/usr/vis/joesmith/MOPS`} ; \\ \text{libname} &:= \text{libname, new\_libname} ; \end{aligned}$$

and then save the .mapleinit file.

All that is left is to call the library (in Maple) using the standard Maple command for libraries, i.e.

```
> with(MOPS) ;
```

each time you need to use the library.

For Windows users: create a directory called C:\MOPS and place the downloaded file MOPS.zip there. Unzip the file (you will need Winzip or some alternate software) and place the 5 resulting files (maple.lib, maple.rep, maple.hdb, maple.ind, and MOPsArchive.mws) in the new directory C:\MOPS. The file MOPsArchive.mws contains an archive of all the procedures we wrote for MOPS.

1. In the \Maple\bin folder, create a .mapleinit file. In it you should write

   $$\begin{aligned} \text{new\_libname} &:= \text{`C:}\backslash\backslash\text{MOPS`} ; \\ \text{libname} &:= \text{libname, new\_libname} ; \end{aligned}$$

   and then save the file.

   You will need to call (in Maple) the library each time you need to use it, using the standard command

   ```
   > with(MOPS) ;
   ```

2. For some unknown reasons, the instructions in variant 1 do not always work on a Windows XP machine. In that case, you will have to type in the pathway for the library, each time you will need to use it. Upon opening a Maple window, you will have to type in

   ```
   >  new_libname := 'C:\\MOPS' ;
   >  libname := libname, new_libname ;
   >  with(MOPS);
   ```

   each time you need the library.

For Mac users: the instructions are similar to the instructions for Windows.

Regardless of the Operating System, we suggest you perform a check before you start using the library. For example, if you type in

```
>  jack(a, [3], 2, 'P') ;
```

the answer should be

$$\mathrm{m}[3] + \frac{3\ \mathrm{m}[2,1]}{1+2\mathrm{a}}$$

3.3 Routines: name, syntax, and description

We give here a list of the 30 routines, and a brief description of what they do; for more extensive mathematical explanation we refer to Section 2. Note that most of them are set up to do calculations both symbolically and numerically.

We use the following notations:

1. $\kappa,\ \sigma$ for partitions,
2. $k,\ s$ for the corresponding partition sums,
3. $n,\ m$ for the number of variables,
4. α for the Jack parameter,
5. i, j for the location of a square in a partition,
6. $A_p,\ B_q$ for two lists of real parameters, one of length p and the other of length q,
7. l for a computational limit,
8. γ, g_1, g_2 for additional Laguerre or Jacobi parameters,
9. $[x]$ for either a number or a list of variables,
10. r for a real parameter,
11. N for the normalization,
12. *exp* for an expression.

Some parameters are optional.

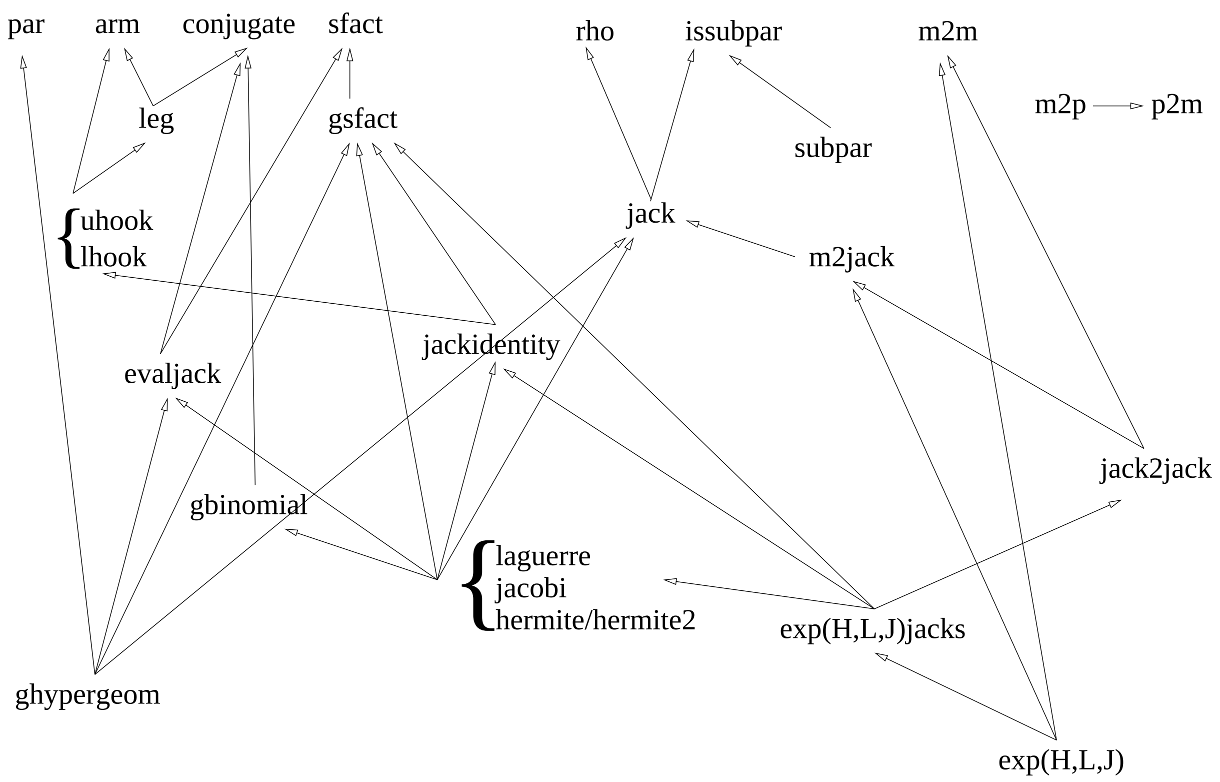
Three of the routines, namely, **m2p**, **p2m**, and **m2m**, are alternatives to the routines **tom** and **top** from the **SF** package to the conversion between the monomial and the power sum basis. For a comparison between our routines and their **SF** counterparts, see Section 4.

Some of the routines are used as building blocks for others; for a tree description of the dependencies, see Figure 2.

Procedure	**Syntax**	*Description*
arm	arm(κ, i, j)	the arm-length of a partition at a square
conjugate	conjugate(κ)	the conjugate partition
expH	expH(α, exp, n)	the expected value of an expression in terms of monomials with respect to the $2/\alpha$-Hermite distribution with n variables
expHjacks	expHjacks(α, exp, n)	the expected value of an expression in terms of Jack polynomials with respect to the $2/\alpha$-Hermite distribution with n variables
expJ	expJ$(\alpha, exp, g_1, g_2, n)$	the expected value of an expression in terms of monomials with respect to the $2/\alpha,\ g_1,\ g_2$-Jacobi distribution with n variables
expJjacks	expJjacks$(\alpha, exp, g_1, g_2, n)$	the expected value of an expression in terms of Jack polynomials with respect to the $2/\alpha,\ g_1,\ g_2$-Jacobi distribution with n variables
expL	expL(α, exp, γ, n)	the expected value of an expression in terms of monomials with respect to the $2/\alpha,\ \gamma$-Laguerre distribution with n variables
expLjacks	expLjacks(α, exp, γ, n)	the expected value of an expression in terms of Jack polynomials with respect to the $2/\alpha,\ \gamma$-Laguerre distribution on n variables
gbinomial	gbinomial(α, κ, σ)	the generalized binomial coefficient
ghypergeom	ghypergeom$(\alpha, A_p, B_q, [x], 'N', l)$	the generalized hypergeometric function corresponding to parameter lists A_p, B_q, evaluated at the point $x \in \mathbb{R}^n$ (or written symbolically for n variables), or (optional) as a series "cut" after terms that sum up to l
gsfact	gsfact(α, r, κ)	the generalized shifted factorial (or generalized Pochhammer symbol)
hermite	hermite$(\alpha, \kappa, [x], 'N')$	the multivariate Hermite polynomial written in Jack polynomial basis
hermite2	hermite2$(\alpha, \kappa, [x], 'N')$	alternative way of computing the multivariate Hermite polynomial written in Jack polynomial basis
issubpar	issubpar(σ, κ)	checks if σ is a subpartition of κ
jack	jack$(\alpha, \kappa, [x], 'N')$	the Jack polynomial as a linear combination of monomials
jack2jack	jack2jack$(\alpha, exp, n, 'N')$	converts a polynomial expression involving Jack polynomials into a linear combination of Jack polynomials
jackidentity	jackidentity(α, κ, m)	the Jack polynomial evaluated at $x_i = 1$, $i = 1..m$
jacobi	jacobi$(\alpha, \kappa, g_1, g_2, [x], 'N')$	the multivariate Jacobi polynomial as a linear combination of Jack polynomials
laguerre	laguerre$(\alpha, \kappa, g, [x], 'N')$	the multivariate Laguerre polynomial as a linear combination of Jack polynomials
leg	leg(κ, i, j)	the leg-length of a partition at the square (i, j)
lhook	lhook(α, κ)	the lower hook of a partition

Procedure	**Syntax**	*Description*
m2jack	m2jack(α, exp, n)	converts an expression involving monomials in Jack polynomial basis
m2m	m2m(exp, n)	converts an expression involving monomials to a linear combination of monomials
m2p	m2p(exp)	converts an expression involving monomials to a linear combination of power sum functions
p2m	p2m(exp, n)	converts an expression involving power sum functions to a linear combination of monomials
par	par(k)	produces and lists all partitions of a given integer
rho	rho(α, κ)	the ρ function of a partition
sfact	sfact(r, k)	the shifted factorial (Pochhammer symbol)
subpar	subpar(κ)	produces and lists all subpartitions of a given partition
uhook	uhook(α, κ)	the upper hook of a partition

Figure 2: Dependence graph for the procedures of MOPS.



3.4 Computing Expected Values / Evaluating Integrals

Let $P_\lambda(x_1, \ldots, x_m)$ be a symmetric polynomial in m variables, with highest order term corresponding to the partition λ. To compute the expected value of P_λ 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

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

and by applying the linearity of expectation, we obtain

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

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 is can be established for the multivariate case. In the Laguerre and Jacobi cases, the univariate formulas generalize easily:

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

$$\int_{[0,1]^m} C_\kappa^\alpha(x_1,\ldots,x_m) d\mu_J^\alpha(x_1,\ldots,x_m) = \frac{(g_1 + \frac{m-1}{\alpha} + 1)_\kappa}{(g_1 + g_2 + \frac{2}{\alpha}(m-1) + 2)_\kappa} C_\kappa^\alpha(I_m) = J_\kappa^{\alpha,g1,g2}(0). \tag{25}$$

For a good reference for the first formula, see Forrester and Baker [2]; the second one was obtained by Kadell [19].

For the Hermite case,

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

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 $H_\kappa^\alpha(x_1,\ldots,x_m)$, using formula (22). Note that if κ 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 α 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 to write the expression in monomial basis, the second – to rewrite the result in Jack polynomial basis, and the third is to replace the Jack polynomials by their expectations, using (26), (24), and (25).

Example. Suppose we want to compute the expected value of

$$z(\alpha, x_1, x_2, x_3) := J^{\alpha}_{[2,1]}(x_1, x_2, x_3) C^{\alpha}_{[1,1,1]}(x_1, x_2, x_3)$$

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 α, must be the same in the two terms of z.

First, we express the two terms in monomial basis:

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

Their product thus becomes a linear combination of sums of products of two monomials, which are in turn converted in linear combinations 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,

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

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

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

We are now able to finish the work:

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

4 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. α, m, γ, 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 α, 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, hence one would expect that the running times for symbolic evaluation would 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.

4.1 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 $(x_1, \ldots, x_m)$, 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 notations given in Table 7.

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

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

and the inequalities

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

for asymptotical estimates.

1. ***jack***. The algorithm uses recurrence (6), together with the 'boundary conditions' $c_{\kappa,\lambda} = 0$ if $\kappa \not\geq \lambda$ in dominance ordering, and $c_{\kappa,\kappa} = \frac{\alpha^k k!}{c'(\kappa,\alpha)}$. The length of the

$k = \|\kappa\|$	size of partition κ
$s = \|\sigma\|$	size of partition σ
$l =$length(κ)	length of partition κ
n	number of variables used for computation
D_κ	number of partitions of k smaller in lexicographical ordering than κ
P_κ	number of partitions of k dominated by κ
$P_{[k]}$	number of partitions of the number k (each partition of k is dominated by $[k]$)
U_κ	number of subpartitions of κ
$U_{\kappa,\sigma}$	number of subpartitions of κ which are superpartitions for σ (this implies σ is a subpartition of κ)
A_κ	number of subpartitions of κ which sum to a number with the same parity with k

Table 7: Notations to be used throughout Section 4.

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 μ (at most $k_1\binom{k+1}{2}$) to see if they are dominated by κ and dominating λ (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 we can get is for the complexity of computing a Jack polynomial is thus $O(k^3 e^{\pi\sqrt{2k/3}})$, which is super-polynomial.

Below 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 [31], which has complexity $O(kP_k) \sim O(e^{\pi\sqrt{2k/3}})$.

Remark 4.1. *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 α. 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 $J^\alpha_{(59,1)}$ will take about 30 minutes.

2. ***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

k	κ	Running time, $\alpha = 1$	Running time, α 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

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

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 κ which is a superpartition of σ.

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

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 κ, as this yields all the binomial coefficients having κ 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 4.2. *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. *We believe that the generalized binomial coefficients are rational functions of α which can always be factored in small-degree factors, so that they are easy to store and operate with.*

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 $O(klU_{\kappa,\ [1^2]})$.

k	κ	Running time, $\alpha = 1$	Running time, α 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	$[8,6,4,3,2,2,1,1,1,1,1]$	11.87	89.61	2619
	$[6,6,6,6,6]$	0.91	3.95	462
	$[20,10]$	0.24	1.20	176

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

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 below are 0 and 1.

k	κ	Running time, $\alpha = 1,\ m = l$	Running time, m symbolic	Running time, α, m symbolic	U_κ
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

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

Remark 4.3. *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 once again $O(ke^{\pi\sqrt{2k/3}})$.

   The Laguerre parameter we chose for each of the computations below is 1.

k	κ	Running time, $\alpha = 1,\ m = l$	Running time, m symbolic	Running time, α, m symbolic	U_κ
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

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

   Remark 4.4. *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.*

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 σ of κ such that $|\sigma| \equiv k \pmod 2$. There are A_κ 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 4.5. *$A_\kappa = O(U_\kappa)$; $A_\kappa \sim U_\kappa/2$.*

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

k	κ	Running time, $\alpha = 1,\ m = l$	Running time, m symbolic	Running time, α, m symbolic	A_κ
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

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

Remark 4.6. *Note that when m is parametrized, but $\alpha = 1$, the computation is almost as fast as in the all-numerical case. 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 α 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.*

4.2 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 κ 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 κ.

   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 α	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 ratio sum-to-length is high, **m2m** performs much better, while on partitions with the ration sum-to-length 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 **m2m**	Runtime **tom**	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 **p2m**	Runtime **tom**	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 **m2p**	Runtime **top**	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 in 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

4.3 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_μ, where μ 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 4.7. *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 $O(nu^4 D_\mu^2) = O(nu^2 e^{3\pi\sqrt{2/3}\sqrt{u}})$.*

As explained in Section 3.4, 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 **expH**	Runtime **expL**	Runtime **expJ**
$m[6]$	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 the 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 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 **expHjacks**	Runtime **expLjacks**	Runtime **expJjacks**
$C[4,3,2,1]$	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

4.4 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 $C^3_{[3,2]}(2.53, -1.09, 7.33)$; 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 we have used to implement these options have been developed and analyzed by Koev and Demmel [6] 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.

5 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 α 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 good, but not top-of-the-line machine (see specifications in Section 4), we have been able to increase the size of the partition enough in order 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.

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 μ^{α}_{H}), the problem of computing the determinant is non-trivial. Closed form answers are known for the cases $\alpha = 1/2, 1$, and 2 (see [1], [5], [27]); however, the general α case does not have an explicit answer (except for some particular situations like in [7, chapter 8].

Since the kth moment of the determinant's distribution is given as the integral of $m_{[k^m]}(x_1, \ldots, x_m) = C^{\alpha}_{[k^m]}(x_1, \ldots, x_m)/C_{[k^m]}(I_m)$ 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

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

The duality principle between α and $1/\alpha$ proved in [7, Section 8.5.2] linking the expected value of the kth power of the determinant of a $n \times n$ matrix to the expected value of the nth 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

$$> \quad -\,\mathbf{a\,(a^4 + 10\,a^3 + 45\,a^2 + 80\,a + 89)}$$

2. *Expectations of powers of the trace.* Consider the problem of computing the expected value of the 6th power 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, n, m [6 ])), n);
   ```

   with answer

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

   It is perhaps worth mentioning that the time needed for the above computation on our test machine (see specifications in Section 4) was of 0.8 seconds.

   Integrals of powers of the trace are related to Catalan numbers and maps on surfaces of various genuses, and are of interest to (algebraic) combinatorialists ([9, 11]).

3. *Smallest eigenvalue distributions.* One of the quantities of interest in the study of Wishart matrices[6] is the distribution of the smallest eigenvalue. There is an extensive literature on the subject, starting with the work of James [16] and Constantine [4]. More recent references are Silverstein [30] and Edelman [8]. In [7], we find a closed-form answer for general α and integer values of γ, in terms of a hypergeometric ${}_2F_0$ function (see also (23)).

   We wrote a small script (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

[6]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).

ensemble, for comparison (see Figure 3). 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,z);
    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);
```

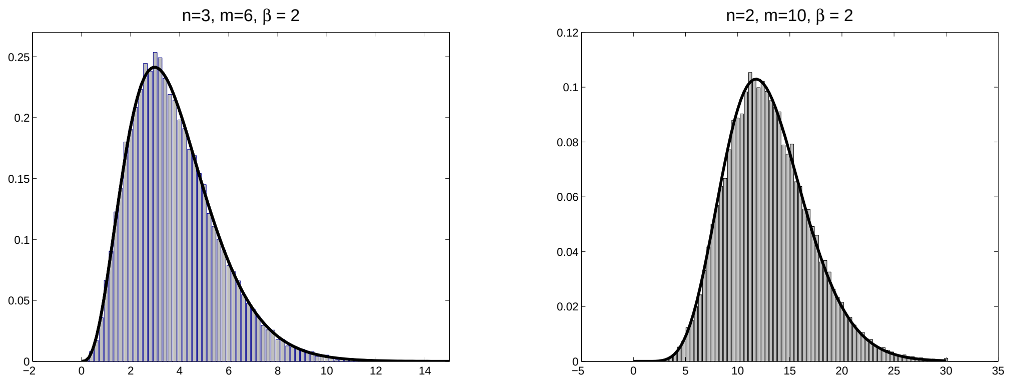


Figure 3: Histograms of the smallest eigenvalue distribution for the real Wishart ensembles of size $(3,6)$ and $(2,10)$ $(\alpha = 1)$, together with the exact distributions as given by (23).

4. *Level densities.* Level density formulas are well-known in terms of orthogonal polynomials for $\alpha = 1/2, 1, 2$. Forrester and Baker [2] have computed these densities in terms of a multivariate Hermite polynomial for $\beta = 2/\alpha$ an even integer (i.e. α 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 α is the inverse of an integer (equivalently, $\beta = 2/\alpha$ is an even integer). This formula is presented below:

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

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 scale both the variable and the density function by $\sqrt{2n\beta} \equiv \sqrt{4n/\alpha}$ (see the script).

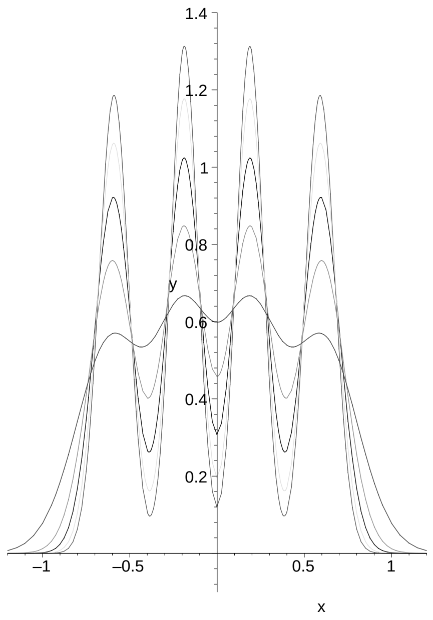
We have used the script below to produce Figure 4, which is an exact plot of the level densities for $n=4$, and $\beta = 2,4,6,8,10$ (equivalently, $\alpha = 1, 1/2, 1/3, 1/4, 1/5$).

```
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 $xI_n$, 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+m/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);
```

Figure 4: Level densities for $n = 4$, $\alpha = 1, 1/2, 1/3, 1/4, 1/5$; "bumps" increase as α decreases.



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

$$\frac{\sqrt{10}e^{-40\,x^2}}{50685458503680000\sqrt{\pi}} \times$$
$$(281474976710656000000000000000\,x^{32} - 281474976710656000000000000000\,x^{30} +$$
$$172051579514388480000000000000\,x^{28} - 69638668456820736000000000000\,x^{26} +$$
$$19434060435475660800000000000\,x^{24} - 3662524084534640640000000000\,x^{22} +$$
$$474005570177728512000000000\,x^{20} - 65812197267210240000000000\,x^{18} +$$
$$16226687345334681600000000\,x^{16} - 3108453312123371520000000\,x^{14} +$$
$$267390948612243456000000\,x^{12} - 13681920034131148800000\,x^{10} +$$
$$2934124875601920000000\,x^{8} - 113006045592760320000\,x^{6} +$$
$$674897998917542400000\,x^{4} - 2060099901411552000\,x^{2} + 32632929952848225).$$

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

Conjecture 5.1. *Let k be an integer, α 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 λ

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

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

6 Copyleft

References

[1] G.E. Andrews, David M. Jackson, and I.P. Goulden. Determinants of random matrices and Jack polynomials of rectangular shape. *Studies Applied Math.*, 2003. To appear.

[2] T. Baker and Peter Forrester. The Calogero-Sutherland model and generalized classical polynomials. *Commun.Math.Phys.*, 188:175–216, 1997.

[3] Yasuko Chikuse. Properties of Hermite and Laguerre polynomials in matrix argument and their applications. *Lin. Alg. Appl.*, 176:237–260, 1992.

[4] A.G. Constantine. Some noncentral distribution problems in multivariate analysis. *Ann. Math. Statist.*, 34:1270–1285, 1963.

[5] R. Delannay and G. Le Caër. Distribution of the determinant of a random real-symmetric matrix from the Gaussian orthogonal ensemble. *Phys. Rev. E*, 62:1526–1536, 2000.

[6] James Demmel and Plamen Koev. Efficient and accurate evaluation of Schur and Jack polynomials. 2003. Preprint.

[7] Ioana Dumitriu. *Eigenvalue Statistics for the Beta-Ensembles.* PhD thesis, Massachusetts Institute of Technology, 2003.

[8] Alan Edelman. The distribution and moments of the smallest eigenvalue of a random matrix of Wishart type. *Lin. Alg. Appl.*, 159:55–80, 1991.

[9] Peter Forrester. *Random Matrices.* 2001. Preprint.

[10] H. O. Foulkes. A survey of some combinatorial aspects of symmetric functions. In *Permutations.* Gauthier-Villars, Paris, 1974.

[11] I. Goulden and David M. Jackson. Maps in locally orientable surfaces and integrals over real symmetric matrices. *Canadian J. Math.*, 49:865–882, 1997.

[12] L.K. Hua. Harmonic Analysis of functions of several complex variables in the classical domains. *Transl. Math. Monogr. Am. Math. Soc.*, 6, 1963.

[13] Henry Jack. A class of symmetric polynomials with a parameter. *Proc. R. Soc. Edinburgh*, 69:1–18, 1970.

[14] David M. Jackson. Personal communication, April 2003.

[15] Alan T. James. The distribution of the latent roots of the covariance matrix. *Ann. Math. Stat.*, 31:151–158, 1960.

[16] Alan T. James. Distributions of matrix variates and latent roots derived from normal samples. *Ann. Math. Stat.*, 35:475–501, 1964.

[17] Alan T. James. Calculation of the zonal polynomial coefficients by use of the Laplace Beltrami operator. *Ann. Math. Stat.*, 39:1711–1718, 1968.

[18] Alan T. James. Special functions of matrix and single argument in Statistics. In Richard A. Askey, editor, *Theory and Application of Special Functions*, pages 497–520. Academic Press, New York, 1975.

[19] K. Kadell. The Selberg-Jack polynomials. *Advances in Mathematics*, 130:33–102, 1997.

[20] Joichi Kaneko. Selberg integrals and hypergeometric functions associated with Jack polynomials. *SIAM J. Math. Anal.*, 24:1086–1110, 1993.

[21] F. Knop and S. Sahi. A recursion and a combinatorial formula for the Jack polynomials. *Invent. Math.*, 128:9–22, 1997.

[22] P.R. Krishnaiah and T.C. Chang. On the exact distribution of the smallest root of the Wishart matrix using zonal polynomials. *Ann. I. Math. Stat.*, 23:293–295, 1971.

[23] M. Lasalle. Polynômes de hermite généralisés. *C.R. Acad. Sci. Paris, Séries I*, 313:579–582, 1991.

[24] M. Lasalle. Polynômes de jacobi généralisés. *C.R. Acad. Sci. Paris, Séries I*, 312:425–428, 1991.

[25] M. Lasalle. Polynômes de laguerre généralisés. *C.R. Acad. Sci. Paris, Séries I*, 312:725–728, 1991.

[26] I.G. Macdonald. *Symmetric Functions and Hall Polynomials.* Oxford University Press Inc, New York, 1995.

[27] Madan Lal Mehta and Jean-Marie Normand. Probability density of the determinant of a random hermitian matrix. *J. Phys. A*, 31:5377–5391, 1998.

[28] Robb J. Muirhead. *Aspects of Multivariate Statistical Theory.* John Wiley & Sons, New York, 1982.

[29] Andrei Okounkov and Grigori Olshanski. Shifted Jack polynomials, binomial formula, and applications. *Mathematical Research Letters*, 4:69–78, 1997.

[30] Jack W. Silverstein. On the weak limit of the largest eigenvalue of a large dimensional sample covariance matrix. *J. Multivariate Anal.*, 30:307–311, 1989.

[31] Richard P. Stanley. Some combinatorial properties of Jack symmetric functions. *Adv. in Math.*, 77:76–115, 1989.