Computing nonsymmetric and interpolation Macdonald polynomials

W. Baratta
Department of Mathematics, University of Melbourne

December 21, 2013

Abstract

In this paper we present a Mathematica notebook for computing nonsymmetric and interpolation Macdonald polynomials. We present the new recursive generation algorithm employed within the notebook and the theory required for its development. We detail the contents of the notebook and conclude with a couple of applications of the notebook.

1 Introduction

The nonsymmetric Macdonald polynomial \( E_q(z; q, t) \) and its generalisation - the interpolation Macdonald polynomial \( \tilde{E}_q(z; q, t) \) - have found applications in mathematical physics, combinatorics and representation theory \[12, 11, 6, 13\].

A feature of the aforementioned polynomials is that they allow for explicit computation. Having this knowledge provides an opportunity to experimentally seek new properties and to check analytic work. Within this paper a computer software program that was developed for these purposes is detailed.

The nonsymmetric and interpolation Macdonald polynomials are generated via recursive generation formulas. This paper begins with the theory required to introduce these formulas (Section 2) and then later shows how these formulas can be used to construct a recursive generation algorithm for the generation of the nonsymmetric and interpolation polynomials (Proposition 2). Section 3.4 details the functions within the Mathematica notebook.
containing this algorithm and the paper concludes with two possible applications of the
notebook.

2 Foundation Theory

2.1 Compositions

The polynomials discussed in this paper are labelled by compositions. We define a composition to be an $n$-tuple $\eta := (\eta_1, \ldots, \eta_n)$ of non-negative integers. Each $\eta_i$ is called a component, the length of a composition is the number of components it contains and the sum of the components is called the modulus and denoted by $|\eta|$. Each composition $\eta$ corresponds to a unique partition $\eta^+$ obtained by rearranging the components of $\eta$ so that they are nondecreasing.

Two important operators that act on compositions are the switching and raising operators. We have the switching operator $s_i$ acting on compositions according to

$$s_i(\eta_1, \ldots, \eta_i, \eta_{i+1}, \ldots, \eta_n) := (\eta_1, \ldots, \eta_{i+1}, \eta_i, \ldots, \eta_n), \quad i = 1, \ldots, n - 1$$

and the raising operator $\Phi$ which has the action

$$\Phi \eta := (\eta_2, \ldots, \eta_n, \eta_1 + 1). \quad (1)$$

It can be seen that every composition $\eta$ can be recursively generated from the all zero composition $(0, \ldots, 0)$. In Section 3.2 we construct an algorithm to recursively generate any composition from $\eta$ from $(0, \ldots, 0)$ using the least number of operators. It is this algorithm that allows us to obtain our main result, the recursive generation algorithm for the nonsymmetric and interpolation polynomials (Proposition 2).

We now proceed to the required polynomial theory.

2.2 Polynomial theory

The nonsymmetric Macdonald polynomial \cite{7} and interpolation Macdonald polynomials \cite{14} are most commonly defined by their eigenfunction and vanishing properties, respectively. Here we take an alternative approach and define them via their respective recursive generation operators.
In the previous section we noted that every composition \( \eta \) can be recursively generated from \((0, \ldots, 0)\) using the operators \( s_i \) and \( \Phi \). In a similar way the nonsymmetric Macdonald polynomial \( E_\eta(z; q, t) := E_\eta(z) \) and the interpolation Macdonald polynomial \( E_\eta^*(z; q, t) := E_\eta^*(z) \) can be recursively generated from \( E_{(0, \ldots, 0)}(z; q, t) = 1 \) and \( E_{(0, \ldots, 0)}^*(z; q, t) = 1 \), respectively, using two elementary operators. We begin by introducing the switching-type operators for the two polynomials; namely the Demazure-Lustig operator \( T_i \) for the nonsymmetric polynomials and the Hecke operator \( H_i \) for the interpolation polynomials. These operators relate \( E_\eta(z) \) and \( E_{s_\eta}(z) \), and \( E_\eta^*(z) \) and \( E_{s_\eta}^*(z) \), respectively.

Each switching-type operator is a realisation of the type-\( A \) Hecke algebra, an associative unitaal algebra over \( \mathbb{Q}(t) \) generated by elements \( h_1, \ldots, h_{n-1} \) and subject to the relations

\[
\begin{align*}
  h_i h_{i+1} h_i &= h_{i+1} h_i h_{i+1}, & 1 \leq i \leq n - 2 \\
  h_i h_j &= h_j h_i, & |i - j| > 1 \\
  (h_i + 1)(h_i - t) &= 0.
\end{align*}
\]

The switching type operators are defined by

\[
T_i := t + \frac{t z_i - z_{i+1}}{z_i - z_{i+1}} (s_i - 1) \quad \text{and} \quad H_i := t + \frac{z_i - t z_{i+1}}{z_i - z_{i+1}} (s_i - 1),
\]

where here, the switching operator \( s_i \) is defined to act on functions according to

\[
(s_i f)(z_1, \ldots, z_i, z_{i+1}, \ldots, z_n) := f(z_1, \ldots, z_{i+1}, z_i, \ldots, z_n).
\]

These operators act on their respective polynomials according to \( [18] \)

\[
T_i E_\eta(z) = \begin{cases} 
  -t \frac{1 - \delta_{\eta, (q, t)}}{1 - \delta_{\eta, (q, t)}} E_\eta(z) + t E_{s_\eta}(z), & \eta_i < \eta_{i+1} \\
  t E_\eta(z), & \eta_i = \eta_{i+1} \\
  t \frac{1 - \delta_{\eta, (q, t)}}{1 - \delta_{\eta, (q, t)}} E_\eta(z) + \frac{(1 - t \delta_{\eta, (q, t)})(1 - t^{-1} \delta_{\eta, (q, t)})}{(1 - \delta_{\eta, (q, t)})^2} E_{s_\eta}(z), & \eta_i > \eta_{i+1},
\end{cases}
\]

and \( [21] \ [13] \)

\[
H_i E_\eta^*(z) = \begin{cases} 
  -t \frac{1 - \delta_{\eta, (q, t)}}{1 - \delta_{\eta, (q, t)}} E_\eta^*(z) + E_{s_\eta}^*(z), & \eta_i < \eta_{i+1} \\
  t E_\eta^*(z), & \eta_i = \eta_{i+1} \\
  t \frac{1 - \delta_{\eta, (q, t)}}{1 - \delta_{\eta, (q, t)}} E_\eta^*(z) + \frac{(1 - t \delta_{\eta, (q, t)})(1 - t^{-1} \delta_{\eta, (q, t)})}{(1 - \delta_{\eta, (q, t)})^2} E_{s_\eta}^*(z), & \eta_i > \eta_{i+1}.
\end{cases}
\]

\[3\]
In (4) and (5) \( \delta_{i,\eta}(q,t) := \eta_i / \eta_{i+1} \), with
\[
\eta_i := q^n t^{-l'_{\eta}(i)},
\]
where
\[
l'_\eta(i) := \# \{ j < i; \eta_j \geq \eta_i \} + \# \{ j > i; \eta_j > \eta_i \}.
\]
The raising type-operators \( \Phi_q \) and \( \Phi_q^* \) of the nonsymmetric and interpolation polynomials, respectively, are given by [1]
\[
\begin{align*}
\Phi_q &:= z_n T_{n-1} \cdots T_i^{-1} = t_{i-n} T_{n-1} \cdots T_i z_i T_{i-1}^{-1} \cdots T_1^{-1} \\
\Phi_q^* &:= (z_n - t^{-n+1}) \Delta.
\end{align*}
\]
In (6) the operator \( T_i^{-1} \) is related to \( T_i \) by the quadratic relation in (2) and given explicitly by
\[
T_i^{-1} := t^{-1} - 1 + t^{-1} T_i,
\]
and in (7) \( \Delta f(z_1, \ldots, z_n) := f(z_n / q, z_1, \ldots, z_{n-1}) \).

The raising type operators act on the Macdonald polynomials according to [1]
\[
\begin{align*}
\Phi_q E_\eta(z) &:= t^{-\# \{ i > 1; \eta_i \geq m \} } E_\eta \Phi_q(z) \\
\Phi_q^* E_\eta^*(z) &:= q^{-m} E_\eta \Phi_q^*(z).
\end{align*}
\]
In Section 3.3 we provide details of the algorithm that implements these formulas to recursively generate any nonsymmetric or interpolation Macdonald polynomial.

Below are some examples on nonsymmetric and interpolation polynomials, for further details on the polynomials we refer the reader to [4].

\[
\begin{align*}
E_{(0,3)}(z) &:= z_3^2 + \frac{t-1}{qt-1} z_1 z_2 + \frac{(q+1)(t-1)}{qt-1} z_1 z_2^2 \\
E_{(2,1)}(z) &:= z_1^2 z_2 + \frac{q(t-1)}{qt-1} z_1 z_2^2 \\
E_{(1,2)}(z) &:= z_1^2 z_2^2 \\
E_{(1,1)}^*(z) &:= z_1 z_2 - \frac{1}{t} z_1 - \frac{1}{t} z_2 + \frac{1}{t} \\
E_{(1,0)}^*(z) &:= z_1 + \frac{t-1}{qt-1} z_2 - \frac{qt^2-1}{t(qt-1)} \\
E_{(0,1)}^*(z) &:= z_2 - \frac{1}{t} \\
\end{align*}
\]
Before considering the recursive generation algorithm we discuss the limiting properties of the nonsymmetric and interpolation Macdonald polynomials.

2.3 Limiting properties

The following diagram indicates that the nonsymmetric Macdonald polynomials can be obtained from the interpolation Macdonald polynomials through a process of homogenisation, that is taking only the terms of top degree from the interpolation polynomial extracts the nonsymmetric polynomial. Furthermore, the diagram indicates that there are many additional families of polynomials that can be obtained from the nonsymmetric and interpolation Macdonald polynomials. It is this feature that allows the computational work discussed in this paper to immediately yield algorithms for the generation of these other families of polynomials.

Through symmetrisation the nonsymmetric Macdonald polynomial $E_\eta(z; q, t)$ can be reduced to symmetric Macdonald polynomial $P_\eta^+(z; q, t)$. Additionally, the symmetric, nonsymmetric and interpolation Macdonald polynomials reduce to the corresponding Jack polynomials, $P_\eta^+(z; \alpha)$, $E_\eta(z; \alpha)$, $E_\eta^*(z; \alpha)$, in the limit $t = q^{1/\alpha}$, $q \to 1$, which themselves reduce to the Schur and zonal polynomials, $s_\eta^+(z)$, $Z_\eta^+(z)$, by setting $\alpha$ to 1 and 2, respectively. Lastly, we have that the symmetric Macdonald polynomials reduce to the Hall-Littlewood polynomials $P_\eta^+(z; t)$ in the limit $q = 0$. 
The definitive reference for the symmetric Macdonald polynomials is [15]. This book also contains details on the Schur and Hall-Littlewood polynomials. An accessible reference for the Jack polynomials and zonal polynomials are [10] and [19], respectively. For details on how the symmetric Macdonald polynomials can be obtained from the nonsymmetric polynomials we refer the reader to [17].

3 Computer Programming

3.1 A brief history

Computer software that generates the polynomial families listed above is quite limited. One of the few programs available is designed to compute symmetric Jack polynomials [9]. The software, written in Maple, uses recurrence formulas to evaluate the coefficients $K'_{\kappa\mu}(\alpha) \in \mathbb{Q}(\alpha)$ in the expansion

$$P_{\eta^+}(z; \alpha) = \sum_{\lambda^+ \leq \eta^+} K'_{\lambda^+\eta^+}(\alpha)m_{\lambda^+}(z),$$

where the $m_{\lambda^+}(z)$ are symmetric monomial functions (see for example [15]). The study in [9] discusses the implementation of the software and the run
times of particular functions. They also provide details for the generation of the generalised classical Hermite, Laguerre and Jacobi polynomials.

A further study in the generation of symmetric Jack polynomials is by Demmel and Koev [8]. They use the expansion formula for a Jack polynomial in one of its variables to obtain a more efficient evaluation for sets of Jack polynomials than was known previously.

Although there are known methods for generating the nonsymmetric and interpolation Macdonald polynomials, for example the Rodrigues formulas [20], software that generates nonsymmetric and interpolation polynomials appears to be nonexistent in the literature. It is our aim to initiate momentum in this area.

We note that the methods presented here would be dramatically improved in efficiency if the algorithms introduced by Demmel and Koev were generalisable to nonsymmetric theory. Although progress has been made using computer-generated coefficients, confirming the ability to generalise the algorithms, the formulas cannot be run efficiently until the nonsymmetric equivalent of equation (6.2) of [8] — a type of dual Pieri formula — is known.

We now present the algorithms implemented in our Mathematica notebook.

3.2 Recursively generating compositions

As stated earlier every composition $\eta$ can be recursively generated from the all zero composition $(0, \ldots, 0)$ using a (non-unique) sequence of raising and switching operators. For example, $\eta = (0, 2, 1)$ can be generated from $(0, 0, 0)$ by $s_1 s_2 s_1 s_2 \Phi s_2 \Phi s_2 \Phi (0, 0, 0)$ or more efficiently by $s_2 \Phi s_1 \Phi (0, 0, 0)$. In polynomial classes for which switching and raising operators exist analogous methods can be used to generate polynomials. Consequently once we construct an algorithm that generates any composition recursively we automatically obtain an algorithm for the polynomials.

Our aim is to construct an algorithm to recursively generate any composition $\eta$ from $(0, \ldots, 0)$ using the least number of operators. We first observe that the raising operator $\Phi$ must be used $|\eta|$ times. Since the raising operator acts on a composition by increasing the value of the component in the first position by one, appending it to the end of the composition and shifting each other component back one position, to minimise the number of operators we must always increase the value of the leftmost component requiring raising.
A systematic way of doing this is to apply the raising operator to build all components greater or equal to a specific size only using the switchings to move the leftmost component needing raising to the first position. We note that this method is quite similar to the Rodrigues formulas construction [20]. Using this method we naturally construct the composition \((\eta^+)^R\), where 
\[
\eta^R := (\eta_n, \eta_{n-1}, \ldots, \eta_1).
\]

Due to the nature of \(\Phi\) there is no possible way to construct a composition containing each component of \(\eta\) using fewer operators.

To reorder \((\eta^+)^R\) minimally we switch each component into its correct position beginning with either \(\eta_1, \eta_2, \ldots\) or \(\eta_n, \eta_{n-1}, \ldots\). We choose to start with repositioning \(\eta_n\). By always choosing the closest component of the unordered composition we ensure that the number of switches is minimal.

**Proposition 1** Define 
\[
l_{\eta,i} := \#\{\eta_j < i\}, 
g_{\eta,i} := \#\{\eta_j \geq i\}, 
\]
and 
\[
\sigma(\eta, i) := (l_{\eta,i-1}, 1, l_{\eta,i-1} + 1, \ldots, 2, \ldots, l_{\eta,i-1} + g_{\eta,i} - 1, \ldots, g_{\eta,i}).
\]

Define 
\[
r_{\eta,i} := \begin{cases} 
\Phi^{g_{\eta,i}}, & i = 1 \\
\Phi^{g_{\eta,i}} s_{\sigma(\eta, i)}, & i > 1,
\end{cases}
\]

where \(s_{(i_1, \ldots, i_l)} := s_i \ldots s_1\). Define 
\[
p_{\eta,i} := \max \left\{ j \leq i : \prod_{k=1}^{i-1} s_{\sigma'(\eta, k)}(\eta^+)^R_j = \eta_i \right\}
\]

and 
\[
s_{\sigma'(\eta, i)} := \begin{cases} 
s_{(p_{\eta,i}, \ldots, i-1)}, & p_{\eta,i} < i \\
1, & p_{\eta,i} = i.
\end{cases}
\]

The minimal length sequence of operators that transforms \((0, \ldots, 0)\) to \(\eta\) is
\[
s_{\sigma'(\eta, 2)} \ldots s_{\sigma'(\eta, n)} r_{\eta, \max(\eta)} \ldots r_{\eta, 1}.
\]

That is
\[
s_{\sigma'(\eta, 2)} \ldots s_{\sigma'(\eta, n)} r_{\eta, \max(\eta)} \ldots r_{\eta, 1}(0, \ldots, 0) = \eta. \quad (8)
\]
Proof We prove (8) in two steps. We first show by induction that
\[ r_{\eta, \max(\eta)} \cdots r_{\eta, 1}(0, \ldots, 0) = (\eta^+)^R. \] (9)
By the definition of \( r_{\eta, 1} \) it is clear that \( r_{\eta, 1}(0, \ldots, 0) \) produces a composition of the form \((0, \ldots, 1, \ldots, 1)\) where the number of 1’s is equal to the number of components of \( \eta \) that are greater or equal to 1. Suppose before applying \( r_{\eta, k+1} \) we have generated the correct number of components with value \( 0, 1, \ldots, k-1 \), that is we have constructed a composition of the form \((\eta^+)_{i_1}^R, \ldots, (\eta^+)_{i_j}^R, k, \ldots, k)\) where the number of \( k \)'s is equal to the number of components of \( \eta \) that are greater or equal to \( k \). Quite obviously \( r_{\eta, k+1}((\eta^+)_{i_1}^R, \ldots, (\eta^+)_{i_j}^R, k, \ldots, k) = ((\eta^+)_{i_1}^R, \ldots, (\eta^+)_{i_j}^R, k, \ldots, k, k+1, \ldots, k+1) \) where the number of \( k \)'s equals the number of components of \( \eta \) equal to \( k \) and the number of \((k+1)\)'s equals the number of components greater than or equal to \( k+1 \). By induction (9) holds. The final task is to show that
\[ s_{\sigma'(\eta, 2)} \cdots s_{\sigma'(\eta, n)}(\eta^+)^R = \eta. \] (10)
This result follows immediately from the definition of \( p_{\eta,i} \) as quite clearly \( \sigma'(\eta, i) \) successively permutes each \( \eta_i \) into the correct position. The fact that the total sequence of operators is of minimal length follows from the action of \( \Phi \) and the inability to generate a composition with components \( \eta_1, \ldots, \eta_n \) more economically than what is specified by (9), the permutation that places each component into its correct position can not be improved either. □

We note that further evidence showing the permutation in (10) is minimal is the comparison of its length to the minimal permutation \( \omega_{\eta, \omega_{\eta}^{-1}(\eta)}^R \), where \( \omega_{\eta} \) is the minimal length permutation such that \( \omega_{\eta}^{-1}(\eta) = \eta^+ \). Due to the different structures of the permutations we use the computational evidence to support our claim.

To provide additional clarity to the algorithm we show how \((4, 1, 2, 1)\) is generated using the above operators.

Example 1 We construct the composition \((4, 1, 2, 1)\) recursively from \((0, 0, 0, 0)\).
We first construct \((1, 1, 2, 4)\) using the operators \(r_{\eta,i}\).

\[
\begin{align*}
  r_{\eta,4}r_{\eta,3}r_{\eta,2}r_{\eta,1}(0, 0, 0, 0) &= r_{\eta,4}r_{\eta,3}r_{\eta,2}\Phi^4(0, 0, 0, 0) \\
  &= r_{\eta,4}r_{\eta,3}r_{\eta,2}(1, 1, 1, 1) \\
  &= r_{\eta,4}r_{\eta,3}\Phi^2(1, 1, 1, 1) \\
  &= r_{\eta,4}r_{\eta,3}(1, 1, 2, 2) \\
  &= r_{\eta,4}\Phi s_1s_2(1, 1, 2, 2) \\
  &= r_{\eta,4}(1, 1, 2, 3) \\
  &= \Phi s_1s_2s_3(1, 1, 2, 3) \\
  &= (1, 1, 2, 4).
\end{align*}
\]

We complete the generation by permuting each component into its correct position

\[
\begin{align*}
  s_{\sigma'(\eta,2)}s_{\sigma'(\eta,3)}s_{\sigma'(\eta,4)}(1, 1, 2, 4) &= s_{\sigma'(\eta,2)}s_{\sigma'(\eta,3)}s_3s_2(1, 1, 2, 4) \\
  &= s_{\sigma'(\eta,2)}s_{\sigma'(\eta,3)}(1, 2, 4, 1) \\
  &= s_{\sigma'(\eta,2)}s_2(1, 2, 4, 1) \\
  &= s_{\sigma'(\eta,2)}(1, 4, 2, 1) \\
  &= s_1(1, 4, 2, 1) \\
  &= (4, 1, 2, 1).
\end{align*}
\]

We now move onto the major goal of the paper, developing the recursive generation algorithm for the nonsymmetric and interpolation Macdonald polynomials.

### 3.3 Recursively generating polynomials

Beyond the nonsymmetric and interpolation Macdonald polynomials there are many other families of polynomials that can be recursively generated via switching and raising type operators. Some examples include nonsymmetric and interpolation Jack polynomials (see e.g. [10, Chap. 12]) and the generalised nonsymmetric Hermite and Laguerre polynomials (see e.g. [10, Chap. 13]). In this section we show how the algorithm developed in Proposition 1 can be employed to recursively generate any of the polynomials in these families.

For simplicity in this section we use \(F_{\eta}(z)\) to denote any of the aforementioned families of polynomials. We show how the recursive generation
algorithm works in the general setting and then provide a specific example for the nonsymmetric Macdonald polynomials.

To most simply express the sequence of operators required to recursively generate a composition according to Proposition 1 we use the numbers $1, \ldots, n - 1$ to represent the allowable switching operators, 0 to represent the raising operator and denote the required sequence by $R(\eta)$. For Example 1 in the previous section we observe that

$$R((4, 1, 2, 1)) = \{0, 0, 0, 0, 0, 2, 1, 0, 3, 2, 1, 0, 2, 3, 2, 1\}.$$  

**Proposition 2** Define

$$RG_j(F_\eta(z), \eta(j), R(\eta)_j) := \left\{
\begin{array}{ll}
\{F_{s_1\eta(j)}(z), s_1\eta(j)\}, & i = R(\eta)_j = 1, \ldots, n - 1 \\
\{F_{\Phi\eta(j)}(z), \Phi\eta(j)\}, & i = R(\eta)_j = 0,
\end{array}
\right.$$  

(11)

where $\eta(j)$ represents the composition obtained after $j$ transformations from $(0, \ldots, 0)$ to $\eta$ specified by $R(\eta)$ and $F_{s_1\eta}(z)$ and $F_{\Phi\eta}(z)$ are obtained from $F_\eta(z)$ using known formulas. With initial input $RG_1(1, (0, \ldots, 0), R(\eta)_1)$ and each subsequent polynomial derived from the previous by entering the newly obtained polynomial and composition along with the next number in $R(\eta)$ in $RG$ the polynomial $F_\eta(z)$ will be obtained after $|R(\eta)|$ steps.

**Proof** By Proposition 1 we know that the sequence specified by $R(\eta)$ will recursively generate the composition $\eta$ from $(0, \ldots, 0)$. Consequently $RG_j$ will recursively generate $F_\eta(z)$ from $F_{(0, \ldots, 0)}(z)$. \qed

We note that we keep track of the composition labelling the polynomial at each stage due to requirements of the formulas transforming $F_\eta(z)$ to $F_{s_1\eta}(z)$ and $F_{\Phi\eta}(z)$.

**Example 2** To recursively generate a nonsymmetric Macdonald polynomial using the methods in the previous proposition we begin by rewriting the recursive generation formulas for the nonsymmetric Macdonald polynomials as

$$E_{s_1\eta}(z) = \left\{
\begin{array}{ll}
t^{-1}T_iE_\eta(z) - \frac{t-1}{t(1-\delta_{i,\eta}(q,t))}E_\eta(z), & \eta_i < \eta_{i+1} \\
E_\eta(z), & \eta_i = \eta_{i+1} \\
\frac{(1-\delta_{\eta,\eta}(q,t))^2}{(1-\delta_{s_1\eta,\eta}(q,t))(1-t^{-1}\delta_{s_1\eta,\eta}(q,t))}(T_iE_\eta(z) - \frac{t-1}{1-\delta_{s_1\eta,\eta}(q,t)}E_\eta(z)), & \eta_i > \eta_{i+1},
\end{array}
\right.$$  

$$E_{\Phi\eta}(z) = t^{\#\{i > 1: \eta_i \leq \eta_1\}}\Phi E_\eta(z).$$
To generate the polynomial $E_{(2,1)}(z; q, t)$ from $E_{(0,0)}(z; q, t) = 1$ using RG we first compute $R((2,1))$ using Proposition 7 to be $R(2,1) = \{0,0,0,1\}$ we then proceed recursively

\[
\begin{align*}
RG_1(1, (0,0), 0) &= \{E_{(0,1)}(z), (0,1)\} \\
RG_2(E_{(0,1)}(z), (0,1), 0) &= \{E_{(1,1)}(z), (1,1)\} \\
RG_3(E_{(1,1)}(z), (1,1), 0) &= \{E_{(1,2)}(z), (1,2)\} \\
RG_4(E_{(1,2)}(z), (1,2), 1) &= \{E_{(2,1)}(z), (2,1)\}.
\end{align*}
\]

In the cases where $F_\eta(z)$ is homogeneous, for example the nonsymmetric Macdonald and Jack polynomials, we can greatly reduce the number of operators required to generate polynomials labelled by compositions with no zero components by making use of the relationship [16]

\[F_{\eta+(k\alpha)}(z) = (z_1 \ldots z_n)^k F_\eta(z). \quad (12)\]

This result allows us to omit the first $n \times \min\{\eta\}$ zeros from $R(\eta)$, forming a new set $R'(\eta)$, and begin our recursive process with

\[\{(z_1 \ldots z_n)^{\min\{\eta\}}, (\min\{\eta\}, \ldots, \min\{\eta\}), R'(\eta)_1\}\]

rather than \{1, (0, \ldots, 0), R(\eta)_1\}.

**Example 3** Using (12) we recursively generate $E_{(2,1)}(z; q, t)$ from $E_{(1,1)}(z; q, t)$. With $R'((2,1)) = \{0,1\}$ we obtain

\[
\begin{align*}
RG_1(z_1z_2, (1,1), 0) &= \{E_{(1,2)}(z), (1,2)\} \\
RG_2(E_{(1,2)}(z), (1,2), 1) &= \{E_{(2,1)}(z), (2,1)\}.
\end{align*}
\]

### 3.4 Runtimes and Software

#### 3.4.1 Algorithm runtimes

In this section we analyse the performance of the algorithm constructed in Proposition 7 when used to generate the nonsymmetric and interpolation Macdonald polynomials.
To provide a thorough analysis we must select polynomials of varying degrees of complexity. That is, varying the number of variables, the maximum degree and of these polynomials selecting those requiring the least and most number of operators to generate. Of the polynomials with $n$ variables and maximum degree $k$ those that take the least and most number of operators to generate using the algorithm of Proposition 2 are labelled by compositions $(q^{n-r}, (q + 1)^r)$ and $(k, 0, \ldots, 0)$, where $k = qn + r$, and require $k$ and $nk$ operators to generate, respectively. We note that we have employed (12) into our algorithm for the nonsymmetric Macdonald polynomials and polynomials labelled by compositions with no zero components will be generated more efficiently than what is specified above, for example the nonsymmetric Macdonald polynomials labelled by $\eta = (k^n)$ would be generated almost instantly.

Table 1 shows the runtimes of the generation of different nonsymmetric Macdonald polynomials using the recursive generation algorithm of Proposition 2. The computer these runtimes were observed on was an iMac 2.4GHz Intel core 2 duo processor in version 7.01.0 of Mathematica.

| $|\eta|$ | $\eta$ | $E_\eta(z)$ time | $E^*_\eta(z)$ time | $E_\eta(z)$ # operators | $E^*_\eta(z)$ # operators |
|---|---|---|---|---|---|
| 4 | (0, 4) | 0.0308 | 0.7792 | 7 | 7 |
| | (1, 3) | 0.0064 | 0.0759 | 3 | 5 |
| | (2, 2) | 0.0011 | 0.0009 | 0 | 4 |
| | (3, 1) | 0.0177 | 0.6081 | 4 | 6 |
| | (4, 0) | 0.0674 | 12.3216 | 8 | 8 |
| | (0, 0, 4) | 0.48 | 4.54 | 10 | 10 |
| | (1, 1, 2) | 0.0014 | 0.0010 | 1 | 4 |
| | (2, 1, 1) | 0.0105 | 0.3108 | 3 | 6 |
| | (4, 0, 0) | 0.6337 | 65.7861 | 12 | 12 |
| | (1, 3, 0) | 0.3171 | 30.3733 | 9 | 9 |
| 7 | (0, 7) | 0.2677 | 34.4630 | 13 | 13 |
| | (3, 4) | 0.0019 | 0.001648 | 1 | 7 |
| | (4, 3) | 0.0070 | 0.1471 | 2 | 8 |
| | (7, 0) | 0.4744 | 68.2642 | 14 | 14 |

Table 1: Runtimes in seconds and number of operators required for the recursive generation algorithm
We note that it is the inhomogeniety of the interpolation polynomials that makes their construction time longer than the nonsymmetric polynomials. Furthermore, we observe occasions where fewer operators result in longer runtimes, for example $E_{(0,0,4)}(z)$ compared with $E_{(1,3,0)}(z)$. This can be explained by the complexity of the polynomials generated in the recursive generation process.
3.4.2 Mathematica notebook

We now provide the details of the Mathematica notebook containing the polynomials and operators contained within this paper, and more generally the author’s PhD thesis [4]. The Mathematica notebook, titled SpecialFunctions.nb, can be found at www.ms.unimelb.edu.au/wbaratta/index.html and ran on Mathematica 7. The original purpose of the Mathematica notebook was to efficiently generate nonsymmetric and nonsymmetric interpolation Macdonald polynomials to develop an understanding of their known theory. Surpassing this motivation it was extensively used throughout the author’s PhD candidature to assist with conjecture formulation and testing. We now present a table containing the key functions defined in the Mathematica notebook. The notations used in the table are consistent with those used throughout the paper. For those extending beyond the current paper (denoted by a *), we refer the reader to [4]. Note, some functions are not introduced until the following section.

| Syntax                | Description                                                                 |
|-----------------------|-----------------------------------------------------------------------------|
| CompositionModulus[η] | Computes |η|                                                               |
| SwitchComposition[η, i] | Computes s_i η                                                               |
| RaiseComposition[η]   | Computes Φ η                                                                |
| *cI[η, I]             | Computes c_I(η)                                                              |
| *Dominance[η, λ]      | Determines whether η ≤ λ or λ ≤ η                                           |
| *PartialOrder[η, λ]   | Determines whether η ≤ λ or λ ≤ η                                           |
| *PartialOrder2[η, λ]  | Determines whether η < λ or λ < η                                           |
| *Successor[η, λ]      | Determines whether η ≤' λ or λ ≤' η                                         |
| *ArmLength[η, i, j]   | Computes a_η(i, j), the armlength of a composition at a square               |
| *ArmCoLength[η, i, j] | Computes a_η'(i, j), the coarmlength of a composition at a square            |
| *LegLength[η, i, j]   | Computes l_η(i, j), the leglength of a composition at a square               |
| LegCoLength[η, i]     | Computes l_η'(i), the coleglength of a composition at a square               |
| *Md[η]                | Computes d_η(q, t)                                                          |
| *MdDash[η]            | Computes d_η'(q, t)                                                         |
| *Me[η]                | Computes e_η(q, t)                                                          |
| *MeDash[η]            | Computes e_η'(q, t)                                                         |
| R[η]                  | Computes the set R(η), specifying the sequence of operators required to generate η from (0, . . . , 0) |

Table 2: Mathematica functions relating to compositions
Syntax Description

**PermutationOnComposition**[$\sigma, \eta$] Computes $\sigma(\eta)$

**DecompositionOnComposition**[$\{s_{i_1}, \ldots, s_{i_l}\}, \eta$] Computes $s_{i_l} \ldots s_{i_1} \eta$

**SwitchingOperator**[$f, i, j$] Computes $s_{ij} f(\ldots, z_i, \ldots, z_j, \ldots)$

**PermutationOnPolynomial**[$\sigma, f$] Computes $\sigma f(z_1, \ldots, z_n)$

**ShortestPermutation**[$\eta$] Computes $\omega_\eta$, the shortest permutation such that $\omega_\eta^{-1}(\eta) = \eta^+$

**RequiredPermutation**[$\eta, \lambda$] Computes the permutation $\sigma$ such that $\sigma(\eta) = \lambda$

Table 3: Mathematica functions relating to permutations

Syntax Description

* **Monomial**[$\eta$] Computes $z^\eta$

**ElementarySymmetricFunction**[$r, n$] Computes $e_r(z)$ in $n$ variables

* **CompleteSymmetricFunction**[$r, n$] Computes $h_r(z)$ in $n$ variables

**SymmetricMonomialFunction**[$\kappa$] Computes $m_\kappa(z)$

* **Vandermonde**[$n$] Computes $\Delta(z)$

* **VandermondeJ**[$J$] Computes $\Delta_J(z)$

**tVandermonde**[$n$] Computes $\Delta_t(z)$

* **tVandermondeJ**[$J$] Computes $\Delta_{tJ}(z)$

**Schur**[$\kappa$] Computes $s_\kappa(z)$

**Zonal**[$\kappa$] Computes $Z_\kappa(z)$

**HallLittlewood**[$\kappa$] Computes $P_\kappa(z; t)$

Table 4: Mathematica functions relating to miscellaneous polynomials
### Syntax Description

| Syntax              | Description                                           |
|---------------------|-------------------------------------------------------|
| SymJack[κ]          | Computes $P_κ(z; α)$                                  |
| NSJack[η]           | Computes $E_η(z; α)$                                  |
| IntJack[η]          | Computes $E^*_η(z; α)$                                |
| *JEval[η]           | Computes $\overline{η}^i$, where $\overline{η}_i := αη_i - Δ'_η(i)$ |
| *EOpSymJack[f, n]   | Computes $D_2(α)f$                                    |
| *EOpNSJack[f, n, i] | Computes $ξ_i f$                                     |
| *EOpIntJack[f, n, i]| Computes $Ξ_i f$                                     |
| *JInnerProduct[f, g, n, k] | Computes $(f, g)_{1/k}$ for polynomials $f, g$ of $n$ variables |

Table 5: Mathematica functions relating to Jack polynomials

| Syntax              | Description                                           |
|---------------------|-------------------------------------------------------|
| NSMac[η]            | Computes $E_η(z; q, t)$                                |
| SymMac[κ]           | Computes $P_κ(z; q, t)$                                |
| ASymMac[κ]          | Computes $S_κ(z; q, t)$                                |
| IntMac[η]           | Computes $E^*_η(z; q, t)$                              |
| SymIntMac[κ]        | Computes $P^*_κ(z; q, t)$                              |
| ASymIntMac[κ]       | Computes $S^*_κ(z; q, t)$                              |
| *MEval[η]           | Computes $\overline{η}_i$, where $\overline{η}_i := q^η t^η_i$ |
| *EOpSymMac[f, n]    | Computes $D^*_1(q, t)f$                               |
| *EOpNSMac[f, i]     | Computes $Y_i f$                                     |
| *EOpIntMac[f, i]    | Computes $Ξ_i f$                                     |
| Ti[f, i]            | Computes $T_i f$                                     |
| TiInv[f, i]         | Computes $T_i^{-1} f$                                 |
| Phiq[f, n]          | Computes $Φ_η f$                                     |
| Hi[f, i]            | Computes $H_i f$                                     |
| PhiqInt[f, n]       | Computes $Φ_η^i f$                                    |
| *MInnerProduct[f, g, n, k] | Computes $(f, g)_{q,q^k}$ for polynomials $f, g$ of $n$ variables |

Table 6: Mathematica functions relating to Macdonald polynomials
**Syntax Description**

- **UPlus[f, n]** Computes $U^+f$, where $f$ is a function of $n$ variables
- **UMinus[f, n]** Computes $U^-f$, where $f$ is a function of $n$ variables
- **UPlusInt[f, n]** Computes $U^{+}\ast f$, where $f$ is a function of $n$ variables
- **UMinusInt[f, n]** Computes $U^{-}\ast f$, where $f$ is a function of $n$ variables
- **OIJ[f, n, I, J]** Computes $O_{I,J}f$, where $f$ is a function of $n$ variables
- **OIJInt[f, n, I, J]** Computes $O^{*}_{I,J}f$, where $f$ is a function of $n$ variables
- **PreSymMac[η*, I, J]** Computes $S^{(I,J)}_{\eta*}(z; q, t)$
- **PreSymIntMac[η*, I, J]** Computes $S^{*(I,J)}_{\eta*}(z; q, t)$
- **aEta[η, I, J]** Computes the normalisation $a^{(I,J)}_{\eta}(q, t)$

*Table 7: Mathematica functions relating to prescribed symmetry polynomials*

| Syntax                  | Description                                                                 |
|-------------------------|-----------------------------------------------------------------------------|
| *JEta[η, r]** and **Pieri[η, λ, r]** | Produces the set of $\lambda$ such that $|\lambda| = |\eta| + r$ and $\eta \preceq \lambda \preceq \eta + (1^n)$ Computes the Pieri-type coefficient $A^{(r)}_{\eta, \lambda}(q, t)$ |
| *GeneralisedBinomialCoefficient[λ, η]** | Computes $(\lambda)_{\eta, q, t}$ |

*Table 8: Mathematica functions relating to Pieri-type coefficients*

**Copyleft**

Copyleft 2011 Wendy Baratta

Permission is granted to anyone to use, modify and redistribute SpecialFunction.nb freely subject to the following.

- We make no guarantees that the software is free of defects.
- We accept no responsibility for the consequences of using this software.
- All explicit use of this notebook must be explicitly represented.
- No form of this software may be included or redistributed in a library to be sold for profit without our consent.
4 Further Work

It is hoped that the notebook SpecialFunctions.nb will continue to assist researchers with conjecture formulation and testing in areas relating to the polynomial families discussed within this paper. In fact this has already shown itself to be the case in a study of special vanishing properties of Jack polynomials for $\alpha = -(r - 1)/(k + 1)$ and Macdonald polynomials with $t^{k+1}q^{r-1} = 1$ [3]. Below are some examples of computations that aim to provide motivation for two research problems stemming from the author’s previous studies; Macdonald polynomials with prescribed symmetry [2] and Pieri-type formulas for nonsymmetric Macdonald polynomials [3].

4.1 Interpolation polynomials with prescribed symmetry

In [2] the author investigated properties of Macdonald polynomials with prescribed symmetry. Macdonald polynomials with prescribed symmetry, denoted $S_{\eta^+}(z; q, t)$, are generalisations of the symmetric Macdonald polynomials and the antisymmetric Macdonald polynomials, denoted $S_{\eta^+}(z; q, t)$ where $\delta = (n - 1, \ldots, 1, 0)$, and are symmetric with respect to some variables and antisymmetric with respect to others. They are obtained from the nonsymmetric Macdonald polynomials via symmetrisation by $O_{I,J}$ and normalisation. The subscript $I$, $J$ indicates the sets of variables for which $O_{I,J}$ symmetrises and antisymmetrises with respect to. The operator $O_{I,J}$ is defined by

$$O_{I,J} := \sum_{\omega \in W_{I\cup J}} \left( -\frac{1}{t} \right)^{l(\omega)} T_\omega,$$

where $l(\omega)$ is the length of the permutation $\omega$,

$$T_\omega := T_{i_1} \ldots T_{i_l},$$

(13)

where $s_{i_1} \ldots s_{i_l}$ is a reduced decomposition of $\omega$, and $W_{I\cup J} := \{s_k; k \in I \cup J\}$, a subset of $S_n$ where each $\omega \in W_{I\cup J}$ can be decomposed as

$$\omega = \omega_I \omega_J, \quad \text{with } \omega_I \in W_I \text{ and } \omega_J \in W_J.$$

The prescribed symmetric Macdonald polynomials is obtained from the nonsymmetric Macdonald polynomials according to

$$O_{I,J}E_\eta(z; q, t) = a^{(I,J)}_\eta S^{(I,J)}_{\eta^+}(z; q, t).$$
where $\eta^*$ is a composition satisfying

$$\eta^*_i \geq \eta^*_{i+1} \text{ for all } i \in I \text{ and } \eta^*_j > \eta^*_{j+1} \text{ for all } j \in J$$

and $a^{(I,J)}_{\eta}$ is a normalisation that ensures the coefficient of $z^{\eta^*} := z_1^{\eta^*_1} \cdots z_n^{\eta^*_n}$ is unity.

The theory of Macdonald polynomials with prescribed symmetry leads most naturally to an investigation of interpolation polynomials with prescribed symmetry. Since the operator $H_i$ [3] satisfies the Hecke algebra [2] and plays the same role in interpolation theory as $T_i$ does in nonsymmetric theory it seems natural to define the prescribed symmetry operator for the interpolation polynomials as

$$O^*_{I,J} := \sum_{\omega \in \mathcal{W}_{I \cup J}} (-1)^{(\omega)} H_{\omega},$$

where $H_{\omega}$ is as in [13].

Trial computation suggest that it may be possible to extend many results obtained in [2] to their interpolation polynomial analogues (see for example, Propositions 3.2.1, 3.2.2 and 3.3.1).

Specifically, Proposition 3.3.1 in [2] specifies the relationship between the antisymmetric and symmetric Macdonald polynomial

$$S_{\eta^*+\delta}(z; q, t) = \Delta_t(z) P_{\eta^*}(z; q, qt),$$

where $\Delta_t(z)$ is the $t$-Vandermonde and specified by

$$\Delta_t(z) := \prod_{1 \leq i < j \leq n} (z_i - t^{-1} z_j).$$

This property has extensions in the theory of Macdonald polynomials with prescribed symmetry and is therefore a suitable starting point for the theory of interpolation Macdonald polynomials with prescribed symmetry.

Here we provide some explicit computations obtained using the Mathematica notebook that may assist with the identification of the relationship between the antisymmetric and symmetric interpolation polynomial.
In[1] SymIntMac[{1, 0}]
Out[1] \( \frac{t(x_1 + x_2) - t - 1}{t} \)

In[2] ASymIntMac[{2, 0}]
Out[2] \( \frac{(x_1 - tx_2)(t(x_1 + x_2) - qt - 1)}{t} \)

In[1] SymIntMac[{1, 1}]
Out[1] \( \frac{(tx_1 - 1)(tx_2 - 1)}{t^2} \)

In[2] ASymIntMac[{2, 1}]
Out[2] \( \frac{(x_1 - tx_2)(tx_1 - 1)(tx_2 - 1)}{t^2} \)

In[1] SymIntMac[{1, 0, 0}]
Out[1] \( \frac{t^2(x_1 + x_2 + x_3) - t^2 - t - 1}{t^2} \)

In[2] ASymIntMac[{3, 1, 0}]
Out[2] \( \frac{(x_1 - tx_2)(x_1 - tx_3)(x_2 - tx_3)(t^2(x_1 + x_2 + x_3) - qt(qt + 1) - 1)}{t^2} \)

These computations show a clear relationship between \( S_{\kappa+\delta}^*(z) \) and \( P_{\kappa}^*(z) \), highlighting that the difficulty in identifying the relationship lies in the transformations of the parameters. It also appears that the \( t \)-Vandermonde product that relates the polynomials is not of standard form.

### 4.2 Pieri-type formulas for nonsymmetric Macdonald polynomials

In [3] the formulas for the coefficients in the expansion of a nonsymmetric Macdonald polynomial and an elementary symmetric function were given.
Explicitly, the formulas for $A^{(r)}_{\eta \lambda}$ in
\[ e_r(z)E_\eta(z; q^{-1}, t^{-1}) = \sum_{\lambda:|\lambda|=|\eta|+r} A^{(r)}_{\eta \lambda} E_\lambda(z; q^{-1}, t^{-1}), \]
where
\[ e_r(z) := \sum_{1 \leq i_1 < \ldots < i_r \leq n} z_{i_1} \ldots z_{i_n}. \]

The formulas of $A^{(r)}_{\eta \lambda}$ given in [3] are given in summation form, though computational evidence suggests that almost all of the coefficients display a product structure. Here we present explicit formulas for the Pieri-type coefficients for the case $r = 2$ showing which cases display a product structure and which cases do not.
In[9]  Pieri[\{1,0,1\},\{0,2,2\},2]  In[10]  Pieri[\{1,0,1\},\{2,0,2\},2]

Out[9] \(\frac{qt^2(q-1)(t-1)}{(qt-1)(qt+1)(qt^2-1)}\)

Out[10] \(\frac{qt^2(q-1)(t-1)}{(qt-1)(qt+1)(qt^2-1)}\)

In[11]  Pieri[\{1,0,1\},\{1,0,2\},2]

Out[11] \(\frac{(q-1)t^2(q(t((q-2)qt^5+(3q-1)t^4+(2-3q)t^3-2t+1)+3(t-1))+2)-1}{(qt-1)(qt^2-1)^2(qt^3-1)}\)

In the above formulas for the Pieri-type coefficients \(A_{(1,0,1),\lambda}^{(2)}(q,t)\) a simple product structure in \(q\) and \(t\), with roots in \(t\) being simple fractional powers of \(q\) for example, is exhibited in all cases except \(\lambda = (1,0,2,1)\). Additional trial computations suggest that it is always the successor of the form \(\Phi\Phi\eta\) that cannot be expressed as a simple product.

Acknowledgement

This work was supported by an Australian Mathematical Society Lift-off Fellowship. I would like to thank AustMS for providing the funding that allowed me to write this paper and further the work from my PhD. I would also like to thank my PhD supervisor who guided me through the research contained in this paper, and kindly agreed to proof read the work after his days as my supervisor were completed. Additional thanks goes to my friend Sam Blake who taught me how to use mathematica.

References

[1] T. H. Baker and P. J. Forrester. A \(q\)-analogue of the type \(A\) Dunkl operator and integral kernel. International Mathematics Research Notices, (14):667–686, 1997.

[2] W. Baratta. Some properties of Macdonald polynomials with prescribed symmetry. Kyusha Journal of Mathematics, (64):323–343, 2010.

[3] W. Baratta. Further Pieri-type formulas for the nonsymmetric Macdonald polynomial. 2011. [arXiv:1008.0892v1].
[4] W. Baratta. *Special function aspects of Macdonald polynomial theory*. PhD thesis, The University of Melbourne, 2011.

[5] W. Baratta and P. J. Forrester. Jack polynomials fractional quantum Hall states and their generalizations. *Nuclear Physics B*, (1):362–381, 2011.

[6] B. A. Bernevig and F. D. M. Haldane. Model fractional quantum Hall states and Jack polynomials. *Phys. Rev. Lett.*, (100):246802, 2008.

[7] I. Cherednik. Nonsymmetric Macdonald polynomials. *International Mathematics Research Notices*, (10):483–515, 1995.

[8] J. Demmel and P. Koev. Accurate and efficient evaluation of Schur and Jack functions. *Mathematics of Computation*, 75(253):223–239 (electronic), 2006.

[9] I. Dumitriu, A. Edelman, and G. Shuman. MOPS: multivariate orthogonal polynomials (symbolically). *Journal of Symbolic Computation*, 42(6):587–620, 2007.

[10] P. J. Forrester. *Log-gases and random matrices*. Princeton University Press, Princeton, NJ, 2010.

[11] J. Haglund, M. Haiman, and N. Loer. A combinatorial formula for nonsymmetric Macdonald polynomials. *American Journal of Mathematics*, (103):359–383, 2008.

[12] M. Haiman. Hilbert schemes, polygraphs and the Macdonald positivity conjecture. *Journal of the American Mathematical Society*, 14(4):941–1006 (electronic), 2001.

[13] S. Kakei, M. Nishizawa, Y. Saito, and Y. Takeyama. The Rational qKZ equation and shifted non-symmetric Jack polynomials. *SIGMA*, 5:10–21, 2009.

[14] F. Knop. Symmetric and non-symmetric quantum Capelli polynomials. *Commentarii Mathematici Helvetici*, 72(1):84–100, 1997.

[15] I. G. Macdonald. *Symmetric functions and Hall polynomials*. The Clarendon Press Oxford University Press, New York, second edition, 1995.
[16] D. Marshall. *Macdonald polynomials*. Masters thesis, The University of Melbourne, 1999.

[17] D. Marshall. Symmetric and nonsymmetric Macdonald polynomials. *Annals of Combinatorics*, 3(2-4):385–415, 1999.

[18] K. Mimachi and M. Noumi. A reproducing kernel for nonsymmetric Macdonald polynomials. *Duke Mathematical Journal*, 91(3):621–634, 1998.

[19] R. J. Muirhead. *Aspects of multivariate statistical theory*. John Wiley & Sons Inc., New York, 1982. Wiley Series in Probability and Mathematical Statistics.

[20] A. Nishino, H. Ujino, and M. Wadati. Rodrigues formula for the nonsymmetric Macdonald polynomial. *Journal of the Physical Society of Japan*, 68(3):701–704, 1999.

[21] S. Sahi. Interpolation, integrality, and a generalization of Macdonald’s polynomials. *International Mathematics Research Notices*, (10):457–471, 1996.