Symbolic implementation, in the *Mathematica* language, for deriving closed formulas for connection coefficients between orthogonal polynomials

Zélia da Rocha*
Departamento de Matemática - CMUP
Faculdade de Ciências da Universidade do Porto
Rua do Campo Alegre n.687, 4169 - 007 Porto, Portugal
Email: mrdioh@fc.up.pt

2010

Abstract

We deal with the problem of obtaining closed formulas for the connection coefficients between orthogonal polynomial sequences and, also, the canonical sequence, using a methodology based on symbolic computations, verifications and demonstrations in the *Mathematica* language. We present, explain and exemplify in details the implementation that has produced, among others, the results contained in two preceding articles of the author referenced herein and we give new insights on automatic demonstrations.

Key words: Connection coefficients; orthogonal polynomials; symbolic computations; automatic demonstrations; *Mathematica* 7.

2010 Mathematics Subject Classification: 33C45, 33D45, 42C05, 33F10, 68W30, 68-01, 68-04.

Contents

*Corresponding author. Work partially supported by the Centro de Matemática da Universidade do Porto, financed by FCT (Portugal) through the programmes POCTI (Programa Operacional "Ciência Tecnologia e Inovação") and POSI (Programa Operacional Sociedade da Informação), with national and European Community structural funds.

1 Introduction

The aim of this work is to present, explain and exemplify in details the *Mathematica* [19, 20] implementation which produces the results showed in [15, 16] for obtaining closed formulas for the connection coefficients between two orthogonal polynomials sequences, or between the canonical sequence and an orthogonal one. This implementation is available on a *Mathematica* notebook attached to [17].

Essentially, we follow the same methodology employed in [15, 16], which is based on symbolic computations, verifications and demonstrations. Here, we explore a new topic on automatic proofs.

We proceed with the simplest method based only on the recurrence relation fulfilled by any orthogonal sequence, which leads to a general recurrence relation with two indexes satisfied by the connection coefficients. The implementation of this relation allows the recursive computation of the first connection coefficients up to a certain order. The capabilities of simplification and factorization of *Mathematica* are crucial to get these coefficients written in a convenient form that enable us to infer the corresponding closed

formula. This task is not always possible and could be more or less difficult depending on the complexity of the examples. In most families the recurrence coefficients are factorized polynomial or rational functions in n. So the *Mathematica* algebraic manipulation commands work pretty well. This software is mostly intended for that cases. After that, if we identify the model for the closed formula, then it can be translated in a *Mathematica* command and we can easily verify the first results obtained. The final goal of the work is to provide a demonstration of the fact that the closed formula is really true, that is, the model is a solution of the general recurrence relation. In principle, this demonstration can also be achieved uniquely by implementation depending on the success of the simplifications abilities of *Mathematica*. When this is not possible, we can always try the procedures employed in [15, 16] doing only a part of the demonstration with the help of *Mathematica*.

We would like to refer the NAVIMA software [1, 2, 3, 8, 18] which implements a similar recursive approach to connection problems in *Mathematica*. NAVIMA algorithm generates in a systematic way a linear recurrence relation in one index only, using some additional proprieties of the orthogonal families, like structure relations or differential equations among others. Then that recurrence relation is solved in the following way: computation of the first few CC from the recurrence in order to guess its general expression. Afterwards, NAVIMA verify by substitution that this guess satisfies the recurrence relation [2, p.768, p.773]. In the present work, any particular character of the sequences like classical, semi-classical or other are not directly employed, the general recurrence relation that we use is based only on the orthogonality of the sequences. Other recursive method is also considered in [10, 11] and recently in [6] with implementation in *Maple*.

We remark the fact that the mathematical literature on this subject is extremely vast and a wide variety of methods have been developed using several other techniques. The reader can find some of the main references in [16]. Furthermore, nowadays, there are symbolic implementations in the domain of orthogonal polynomials. We refer the CAOP [12], a package for calculating formulas for orthogonal polynomials belonging to the Askey scheme in *Maple*, one approach based on special functions available on internet.

Let us present the summary of the this work.

This article begins with a section that remembers the basic definitions and mathematic results needed in the sequel, namely the notions of connection coefficients, orthogonal polynomials and symmetry and, also, the recurrence relation of order two satisfied by any orthogonal sequence, the resultant general recurrence relation fulfilled by the connection coefficients and the corresponding relations in the symmetrical case [15, 16].

Afterwards, we present an extensive section dedicated to the symbolic computation of the connection coefficients. As this work is based on the implementation of recurrence relations, the first subsection is devoted to this topic taking as basic example the computation of the Fibonacci numbers. In the following subsection, we give explicitly the commands that implement the general recurrence relation that allows the recursive computation of the first connection coefficients up to a certain fixed order: we call them the CC commands. Moreover, we discuss several important details in this implementation

that is conceived in order to work for every example. We note that each case is determined by the recurrence coefficients (the coefficients of the recurrence relation of order 2) and the parameters of the two orthogonal polynomials sequences corresponding to the connection coefficients we want to compute. The names of the commands that translate these elements are arguments of CC and must be implemented in a certain manner. Therefore, in the subsection 3.3, we propose the implementation of the recurrence coefficients and the command MOP that computes the corresponding monic orthogonal polynomials using the above cited recurrence relation of order two. In the subsection 3.4, we show how to call the CC commands in order to get the first connection coefficients up to a fixed order. Also, we established a command named $verificationRCC$ based on the mathematical definition of the connection coefficients, in order to verify the results produced by CC. From these results, we try to infer a model to a direct closed formula for the connection coefficients. If we succeed, it is always possible to implement that formula in a command and compare the results produced by it with those computed recursively. This comparison is implemented in a command named $verificationDCC$ presented in subsection 3.5. In the following subsection, we treat the main task of proving that the formula is true through a symbolic implementation furnish by the command $demonstrationDCC$. At last, we present a list of the main commands refereeing, for each one, the name, brief description, arguments, *Mathematica* commands and other commands employed in the implementation and the result produced.

Section 4 is dedicated to the presentation of test examples. In the first subsection, we follow all the steps of the methodology, giving the implementation and the results, for the Charlier polynomials [7]. Next, we express the canonical polynomials in terms of the monic Laguerre ones. Our goal is to explain how the commands work with an example involving the canonical sequence. We present all the steps giving the corresponding results. We finish the section with a description list of the commands developed.

Section 5 concerns the symmetrical case that need a specific treatment, because, often, there are two different formulas for the connection coefficients corresponding to odd or even indexes. The commands CC, $verificationRCC$ and $verificationDCC$ work perfectly in this case, but the implementation of the closed formulas must be more complete and the command $demonstrationDCC$ must be replaced by another one named $demonstrationSymDCC$. Next, the main command description is furnish. In the subsection 5.3, we present as test example the generalize Hermite polynomials [7] and we finish this section with the corresponding description.

We insist on the fact that all the developed commands are designed to accept every example. The characteristic elements of each case to treat, that is, the names of the commands corresponding to the recurrence coefficients and the parameters pass as arguments of CC, $verficationRCC$ and MOP. The name of the command translating the closed formula passes as argument of $verficationDCC$ and either $demonstrationDCC$ or $demonstrationSymDCC$.

For the sake of clearness, we have preferred to exemplify the commands with simple test examples. Moreover, the methodology work perfectly in the cases of Laguerre and Bessel with the canonical sequence presented in [15]; can be used to treat partially other

examples as the Gegenbauer and the Jacobi families with the canonical sequence given in [16], and allows to explore new cases as the semi-classical ones treated in [16] and [17].

In the last section, we expose some conclusions and commentaries on the work presented here and in [15, 16, 17].

2 Connection coefficients for orthogonal polynomials

Let $\mathcal{P}$ be the vector space of polynomials with coefficients in $\mathbb{C}$ and let $\mathcal{P}'$ be its dual. We denote by $\langle u, p\rangle$ the effect of $u \in \mathcal{P}'$ on $p \in \mathcal{P}$. In particular, $\langle u, x^n\rangle := (u)_n, n \geqslant 0$ represent the moments of u.

Let $\{P_n\}_{n\geqslant 0}$ be a monic polynomial sequence (MPS) with $\deg P_n = n,\ n \geq 0$, that is, $P_n(x) = x^n + ...$ A form u is said regular [13, 14] if and only if there exists a MPS $\{P_n\}_{n\geqslant 0}$, such that:

$$\begin{aligned} \langle u, P_n P_m\rangle &= 0\ ,\ n \neq m\ ,\ n, m \geq 0\ , &(1)\\ \langle u, P_n^2\rangle &\neq 0\ ,\ n \geq 0\ . &(2)\end{aligned}$$

In this case, $\{P_n\}_{n\geqslant 0}$ is said regularly orthogonal with respect to u and is called a monic orthogonal polynomial sequence (MOPS). The orthogonality conditions are given by (1), and (2) corresponds to the regularity conditions.

The sequence $\{P_n\}_{n\geqslant 0}$ is regularly orthogonal with respect to u if and only if [13, 14] there exist two sequences of coefficients $\{\beta_n\}_{n\geqslant 0}$ and $\{\gamma_{n+1}\}_{n\geqslant 0}$, with $\gamma_{n+1} \neq 0,\ n \geqslant 0$, such that, $\{P_n\}_{n\geqslant 0}$ satisfies the following initial conditions and recurrence relation of order 2:

$$P_0(x) = 1, \quad P_1(x) = x - \beta_0, \tag{3}$$

$$P_{n+2}(x) = (x - \beta_{n+1})P_{n+1}(x) - \gamma_{n+1}P_n(x), \quad n \geqslant 2. \tag{4}$$

Furthermore, the recurrence coefficients $\{\beta_n\}_{n\geqslant 0}$ and $\{\gamma_{n+1}\}_{n\geqslant 0}$ satisfy:

$$\beta_n = \frac{\langle u, xP_n^2(x)\rangle}{\langle u, P_n^2(x)\rangle}, \quad n \geqslant 0, \tag{5}$$

$$\gamma_{n+1} = \frac{\langle u, P_{n+1}^2(x)\rangle}{\langle u, P_n^2(x)\rangle}, \quad n \geqslant 0. \tag{6}$$

We remark that, from (3) and (6), the regularity conditions (2) are equivalent to the conditions $\gamma_{n+1} \neq 0,\ n \geqslant 0$.

As usual, we suppose that

$$\beta_n = 0,\ \gamma_{n+1} = 0,\ P_n(x) = 0,\ n < 0. \tag{7}$$

We recall that the recurrence coefficients of the canonical sequence $\{X_n\}_{n\geqslant 0}$, $X_n(x) = x^n$, are

$$\beta_n = 0,\ \gamma_{n+1} = 0,\ n \geqslant 0; \tag{8}$$

it is a nonregular sequence.

Given two MPS $\{P_n\}_{n\geqslant 0}$ and $\{\tilde{P}_n\}_{n\geqslant 0}$ the coefficients that satisfy the equality

$$P_n(x) = \sum_{m=0}^{n} \lambda_{n,m}\tilde{P}_m(x),\ n \geqslant 0, \tag{9}$$

are called the connection coefficients: $\lambda_{n,m} := \lambda_{n,m}^{P\tilde{P}} := \lambda_{n,m}(P \leftarrow \tilde{P})$.

It is obvious that these coefficients exist and are unique, because the polynomials are linearly independent.

Let us suppose that the two monic polynomial sequences $\{P_n\}_{n\geqslant 0}$ and $\{\tilde{P}_n\}_{n\geqslant 0}$ are orthogonal and are given by their recurrence coefficients $\{\beta_n\}_{n\geqslant 0}$, $\{\gamma_{n+1}\}_{n\geqslant 0}$ and $\{\tilde{\beta}_n\}_{n\geqslant 0}$, $\{\tilde{\gamma}_{n+1}\}_{n\geqslant 0}$, respectively, let us consider the problem of compute and determine closed formulas for the connection coefficients.

As demonstrate in [15, 16], the connection coefficients fulfill the following initial conditions and general recurrence relation

$$\lambda_{n,m} = 0,\ n < 0 \text{ or } m < 0 \text{ or } m > n, \tag{10}$$

$$\lambda_{n,n} = 1,\ n \geqslant 0, \tag{11}$$

$$\lambda_{1,0} = \tilde{\beta}_0 - \beta_0, \tag{12}$$

$$\lambda_{n,m} = \left(\tilde{\beta}_m - \beta_{n-1}\right)\lambda_{n-1,m} - \gamma_{n-1}\lambda_{n-2,m} + \tilde{\gamma}_{m+1}\lambda_{n-1,m+1} + \lambda_{n-1,m-1} \tag{13}$$
$$,\ 0 \leqslant m < n,\ n \geqslant 2.$$

We recall that a MPS, $\{P_n\}_{n\geqslant 0}$, is symmetric if and only if $P_n(-x) = (-1)^n P_n(x)$, $n \geqslant 0$; if it is orthogonal the symmetry is equivalent to $\beta_n = 0,\ n \geqslant 0$ [7].

If $\{P_n\}_{n\geqslant 0}$ and $\{\tilde{P}_n\}_{n\geqslant 0}$ are two symmetrical orthogonal polynomial sequences, then the corresponding connection coefficients fulfil [15, 16], for $0 \leqslant m \leqslant n$ and $n \geqslant 0$,

$$\lambda_{2n,2m-1} = 0,\ \lambda_{2n+1,2m} = 0, \tag{14}$$

$$\lambda_{2n,2m} = -\gamma_{2n-1}\lambda_{2n-2,2m} + \tilde{\gamma}_{2m+1}\lambda_{2n-1,2m+1} + \lambda_{2n-1,2m-1}, \tag{15}$$

$$\lambda_{2n+1,2m+1} = -\gamma_{2n}\lambda_{2n-1,2m+1} + \tilde{\gamma}_{2m+2}\lambda_{2n,2m+2} + \lambda_{2n,2m}. \tag{16}$$

In the case of other polynomial normalizations, that is, $B_n(x) = k_n P_n(x)$, $k_n \neq 0$ and $\tilde{B}_n(x) = \tilde{k}_n\tilde{P}_n(x)$, $\tilde{k}_n \neq 0$, $n \geq 0$, the corresponding connection coefficients to consider are

$$\lambda_{n,m}^{B\tilde{B}} := k_n\lambda_{n,m}^{P\tilde{P}}\tilde{k}_m^{-1}\ \ ,\ \ \lambda_{n,m}^{BX} := k_n\lambda_{n,m}^{PX}\ \ ,\ \ \lambda_{n,m}^{XB} := \lambda_{n,m}^{XP}k_m^{-1}\ ,\ 0 \leq m \leq n,\ n \geq 0\ .$$

3 Symbolic computation of connection coefficients

3.1 How to implement recurrence relations in *Mathematica*

In this work, the computation of the connection coefficients essentially involves the implementation of recurrence relations. In the *Mathematica* language [19, 20], this can be done easily using two different types of function definitions, as follows.

1- $f[x\_] := rhs$, define a standard function.

In this case, the value of the function is computed every time we ask for it.

2 - $f[x\_] := f[x] = rhs$, define a function which remembers values that it finds.

In this case, *Mathematica* never recomputes a function value, because the first time a function value is computed it is automatically stored in memory.

Let us consider the typical example of Fibonacci numbers, given by the following initial conditions and recurrence relation

$$f(0) = f(1) = 1\ ,\ \ f(n) = f(n-1) + f(n-2). \tag{17}$$

The point is that if we calculate say $f(10)$ by just applying the recursion relation over and over again, we end up having to recalculate quantities like $f(5)$ many times. In a case like this, it is therefore better just to remember all the values of the Fibonacci numbers already computed [19, 20]. This can be done, using a function definition of type 2, as follows.

$f[0] = f[1] = 1;$
$f[n\_/; And[n \in Integer, n \geq 0]] := f[n] = f[n-1] + f[n-2];$

There is a trade-off involved in remembering values. It is faster to find a particular value, but it takes more memory space to store all of them. We should usually define functions to remember values only if the total number of different values that will be produced is comparatively small, or the expense of recomputing them is very great [19, 20]. Of course that we can manage other kinds of implementation, but the preceding ones are the most straightforward to use.

3.2 Implementation of the general recurrence relation for the connection coefficients

It is clear that, the recursive computation of the connection coefficients is more difficult than the computation of the Fibonacci numbers, because the $\lambda_{n,m}$ have two indexes, the relation (13) is more complicated than (17) and involves the computation of the recurrence coefficients β_n, γ_{n+1}, $\tilde{\beta}_n$ and $\tilde{\gamma}_{n+1}$, with the corresponding parameters. In spite of this, all the considerations cited above remain valid for the $\lambda_{n,m}$. In this work, we have used always functions that remember values. In practice, the maximal value of n on $\lambda_{n,m}$ and P_n is 20 or 25, so there is not a large amount of elements to store in memory. On the other hand, to recompute the connection coefficients could be quite expensive, if there are several parameters to consider, so the standard function definitions are not indicated to this problem. We have not needed to do a more careful and complicated implementation, because we have obtained all the results without any problems of time or space memory.

Let us explain exactly how the implementation is done.

The initial conditions and the general recurrence relation for the connection coeffi-

cients (10)-(13) are easily implemented as follows.

$$
\begin{array}{l}
CC[rc\_, rct\_][p\_\_\_][pt\_\_\_] \\
[n\_/; And[IntegerQ[n], n < 0], m\_/; IntegerQ[m]] := \\
CC[rc, rct][p][pt][n, m] = 0;
\end{array} \tag{18}
$$

$$
\begin{array}{l}
CC[rc\_, rct\_][p\_\_\_][pt\_\_\_] \\
[n\_/; IntegerQ[n], m\_/; And[IntegerQ[m], m < 0]] := \\
CC[rc, rct][p][pt][n, m] = 0;
\end{array} \tag{19}
$$

$$
\begin{array}{l}
CC[rc\_, rct\_][p\_\_\_][pt\_\_\_] \\
[n\_/; And[IntegerQ[n], n \geq 0], n\_/; And[IntegerQ[n], n \geq 0]] := \\
CC[rc, rct][p][pt][n, n] = 1;
\end{array} \tag{20}
$$

$$
\begin{array}{l}
CC[rc\_, rct\_][p\_\_\_][pt\_\_\_][1, 0] := CC[rc, rct][p][pt][1, 0] = \\
Factor[FullSimplify[\ rct[pt][0][[1]] - rc[p][0][[1]]\ ]];
\end{array} \tag{21}
$$

$$
\begin{array}{l}
CC[rc\_, rct\_][p\_\_\_][pt\_\_\_][n\_/; IntegerQ[n], m\_/; IntegerQ[m]] := \\
CC[rc, rct][p][pt][n, m] = \\
Module[\{\}, \\
\quad If[And[n >= 0, n <= m - 1], Return[0], \\
\quad\quad Return[\ Factor[FullSimplify[\\
\quad\quad\quad (rct[pt][m][[1]] - rc[p][n-1][[1]]) * CC[rc, rct][p][pt][n-1, m] - \\
\quad\quad\quad rc[p][n-1][[2]] * CC[rc, rct][p][pt][n-2, m] + \\
\quad\quad\quad rct[pt][m+1][[2]] * CC[rc, rct][p][pt][n-1, m+1] + \\
\quad\quad\quad CC[rc, rct][p][pt][n-1, m-1]\ ]] \\
\quad\quad\quad\quad];(* end\ of\ Return\ *) \\
\quad\quad];(* end\ of\ If\ *) \\
\quad];(* end\ of\ Module\ *)
\end{array} \tag{22}
$$

The arguments $[rc\_, rct\_]$ and $[p\_\_\_][pt\_\_\_]$ correspond to the names of the recurrence coefficient commands and the names of the parameters of the sequences P and $\tilde{P}$, in such a way that, $rc[p][n]$ and $rct[pt][n]$ are lists corresponding to $\{\beta_n, \gamma_n\}$ and $\{\tilde{\beta}_n, \tilde{\gamma}_n\}$, respectively. To access to the first or second element of a list, we join to its name [[1]] or [[2]]. The commands corresponding to rc and rct must be given before calling the CC commands (see next subsections). Note that the triple underscore next p and pt is a pattern object that can stand for any sequence of zero or more expressions [19, 20], which means that it matches for polynomial sequences without parameters as is the case of the canonical one.

After the above explanations, we think it is easy to accept that the definitions of the CC commands (18)-(19), (20), (21) and (22) correspond to the mathematical relations (10), (11), (12) and (13), respectively. Note that we put exclusion and initial conditions ahead of the general recurrence relation. This principle has been followed by *Mathematica* in order to avoid special rules be shadowed by more general ones [19, 20].

Remark that, the CC commands only accept integer values for the arguments n or m. This is done by the restrictions following the patterns arguments. The aim is to avoid acceptation of absurd values for n and m. In fact, if we could give, for example, $n = 5.5$ or n symbolic in a CC-calling statement, the recurrence relation would never match the initial conditions and would enter in an infinite recursion process.

We have added to the body of (21) and (22) the very useful *Mathematica* commands $Factor$ followed by $FullSimplify$ in order to get results written in the simplest factorized form [19, 20]. In examples for which the recurrence coefficients are rational functions $Factor$ should be replace by $Together$. We note that $Factor[expr]$ writes $expr$ as a product of minimal factors and $Together[expr]$ puts terms in a sum over a common denominator and cancels factors in the result [19, 20].

3.3 Implementation of recurrence coefficients and orthogonal polynomials

Let us consider two polynomial sequences identified, for example, by the names $Ex1P$ and $Ex2P$; the first one with two parameters $p1$ and $p2$ and the second one with one parameter p, for instance. We would like to compute the connection coefficients $\lambda_{n,m} := \lambda_{n,m}(Ex1P \leftarrow Ex2P)$. The CC-commands suppose that the corresponding recurrence coefficients are implemented in commands, named $Ex1C$ and $Ex2C$, for example, as follows.

$$Ex1C[p1\_, p2\_][n\_] := Ex1C[p1, p2][n] = \\ Module[\{...\},\ ...; Return[\{..., ...\}]\ ]; \tag{23}$$

$$Ex2C[p\_][n\_] := Ex2C[p][n] = Module[\{...\},\ ...; Return[\{..., ...\}]\ ]; \tag{24}$$

In that manner, these definitions return a list of two elements $\{..., ...\}$, thus $Ex1C[p1, p2][n][[1]]$ and $Ex1C[p1, p2][n][[2]]$ correspond to β_n and γ_n; and $Ex2C[p][n][[1]]$ and $Ex2C[p][n][[2]]$ correspond to $\tilde{\beta}_n$ and $\tilde{\gamma}_n$, respectively.

Note that there is no restriction on the argument n, because nearly always, we are able to give a closed formula for the recurrence coefficients valid for all n, so n can be a symbol argument or a numeric expression in the calling statement ($Ex1C[p1, p2][5]$; $Ex1C[p1, p2][n]$, for example). In spite of the fact that the computation of recurrence coefficients is not recursive, we have used functions that remember values, because each recurrence coefficient is needed many times.

The monic orthogonal polynomials are recursively computed using the identities (3)-

(4) and are implemented in the following MOP-commands.

$$MOP[rc\_][p\_\_\_][n\_/; And[IntegerQ[n], n < 0], x\_] := \tag{25}$$
$$MOP[rc][p][n, x] = 0;$$

$$MOP[rc\_][p\_\_\_][0, x\_] := MOP[rc][p][0, x] = 1; \tag{26}$$
$$MOP[rc\_][p\_\_\_][1, x\_] := MOP[rc][p][1, x] = \tag{27}$$
$$Simplify[\ x - rc[0][[1]]\ ];$$
$$MOP[rc\_][p\_\_\_][n\_/; And[IntegerQ[n], n \geq 2], x\_] := \tag{28}$$
$$MOP[rc][p][n, x] =$$
$$Collect[\ Simplify[\ (x - rc[p][n-1][[1]]) * MOP[rc][p][n-1, x] -$$
$$rc[p][n-1][[2]] * MOP[rc][p][n-2, x]\ ],$$
$$x,\ Factor\ ];$$

To the sake of implementation, it is necessary to consider the shift $n \leftarrow n - 2$ in (4) in order to have the index n isolated on the left hand side of the relation.

As set before, the arguments $[rc\_]$ and $[p\_]$ correspond to the name of the recurrence coefficients command and the name of the parameters of the sequence P. As usual n is the degree and x is the variable in $P_n(x)$.

It is easy to see that the MOP commands (25), (26)-(27) and (28) correspond to the mathematical relations (7), (3) and (4), respectively. Remark that we have used functions that remember values and in the calling statements of MOP n must be a fixed integer and x should be a symbol or a numeric expression.

In order to get the polynomials simplified and written in the canonical base with factorized coefficients we have used the $Simplify$ and $Collect$ *Mathematica* commands ($Collect[$ *expr*, x, *command* $]$ groups together powers of x in *expr* and applies the *command* to the corresponding coefficients; in rational cases $Factor$ shoud be replaced by $Together$ [19, 20]). If we are going to use the MOP command for no numeric values of x ($NumericQ[x] === False$), then $Collect$ should be omitted.

The commands that define the polynomials of the specified examples with which we treat can be easily implemented in one line calling the MOP definition as follows.

$$Ex1P[p1\_, p2\_][n\_, x\_] := Ex1P[p1, p2][n, x] = MOP[Ex1C][p1, p2][n, x]; \tag{29}$$
$$Ex2P[p\_][n\_, x\_] := Ex2P[p][n, x] = MOP[Ex2C][p][n, x]; \tag{30}$$

To get the polynomials of degree 5 for example, we furnish

$$Ex1P[p1, p2][5, x] \qquad\qquad Ex2P[p][5, x]$$

If we want to consider special values to the parameters, for example, $p1 = 0$, $p2 = 1$ and $p = -1$, we call

$$Ex1P[0, 1][5, x] \qquad\qquad Ex2P[-1][5, x]$$

To evaluate the preceding in $x = 0$, we give

$$Ex1P[0,1][5,0] \qquad\qquad Ex2P[-1][5,0]$$

3.4 Getting and verifying results for connection coefficients computed recursively

Supposing the structure of commands previously establish, the statement to get the connection coefficients $\lambda_{n,m} := \lambda_{n,m}(Ex1 \leftarrow Ex2)$, for $n = 0, ..., 10,\ m = 0, ..., n$, is

$$Table[\ CC[Ex1C, Ex2C][p1, p2][\tilde{p}][n, m],\ \{n, 0, 10\}, \{m, 0, n\}\ ]$$

, where $Table[expr, \{i, imin, imax\}, \{j, jmin, jmax\}]$ generates a double list of the values of $expr$ when i runs from $imin$ to $imax$ and j runs from $jmin$ to $jmax$. The list associated with i is outermost [19, 20].

We should make some kind of verifications in order to test our implementation. A crucial one is based on the definition of connection coefficients given by the equality (9), which can be reproduced with our commands fixing the upper value of $n = nmax$. This can be done by defining

$$\begin{aligned}
&verificationRCC[rc_, rct_][p___][pt___] \\
&\qquad\qquad [nmax_/; And[IntegerQ[nmax], nmax \geq 0] := \\
&Module[\{x, answer = Table[True, \{n, 0, nmax\}]\}, \\
&\quad Table[\ FullSimplify[\ MOP[rc][p][n, x] - \\
&\qquad\qquad \sum_{m=0}^{n} CC[rc, rct][p][pt][n, m] * MOP[rct][pt][m, x]\ ] === 0 \\
&\qquad\qquad , \{n, 0, nmax\}\ ] === answer\ ];
\end{aligned}$$

In *Mathematica*, in order to ask if two entities a and b are identical, we can give $a === b$ and have $True$ or $False$ as answer. Often, it is more efficient to verify if their difference is 0 doing $FullSimplify[a - b] === 0$ [19, 20].

In our example, the calling statement to get this verification up to 20, for example, is

$$verificationRCC[Ex1C, Ex2C][p1, p2][\tilde{p}][20]$$

And we should have $True$ as answer.

To get the connection coefficients corresponding to two different polynomial sequences concerning the same familiy, $Ex1P$, but with different parameters, $[p1, p2]$ and $[\tilde{p}1, \tilde{p}2]$, we call

$$Table[\ CC[Ex1C, Ex1C][p1, p2][\tilde{p}1, \tilde{p}2][n, m],\ \{n, 0, 10\}, \{m, 0, n\}\ ]$$

The corresponding verification up to 20 is

$$verificationRCC[Ex1C, Ex1C][p1, p2][\tilde{p}1, \tilde{p}2][20]$$

And, we should have $True$ as answer.

3.5 Getting and verifying results for connection coefficients computed by a direct closed formula

In [15, 16], it is shown that, in several important cases, we are able to infer from the table of results produced by the CC-definitions, what is the mathematical direct closed formula of the connection coefficients, for all n and m. Then, we can translate this model in a *Mathematica* command and compare the connection coefficients given by it with those produced by the recursive computations of the CC-commands, for the first values of n up to $nmax$. This comparison can be implemented in a command like

$$
\begin{aligned}
&verificationDCC[dcc\_][rc\_, rct\_][p\_\_\_][pt\_\_\_] \\
&\qquad\qquad [nmax\_/; And[nmax \in Integer, nmax \geq 0]] := \\
&Module[\{answer = Table[True, \{n, 0, nmax\}, \{m, 0, n\}]\}, \\
&\qquad Table[\ FullSimplify[\ dcc[p][pt][n, m] - CC[rc, rct][p][pt][n, m]\ ] === 0 \\
&\qquad\qquad , \{n, 0, nmax\}, \{m, 0, n\}\ ] === answer\ ];
\end{aligned}
$$

where the *dcc* argument is the name of the direct definition,

In the case of the examples $Ex1$ and $Ex2$, we establish a command of the following type for the closed formula.

$$
\begin{aligned}
&Ex1Ex2DCC[p1\_, p2\_][pt\_][n\_, m\_] := Ex1Ex2DCC[p1, p2][pt][n, m] = \\
&Module[\{...\}, Return[...]\ ];
\end{aligned}
$$

Note that the parameters of the two sequences must figure separately as arguments. Remark, also, that there are no restriction on n and m, because they can be fixed integers or symbols.

The comparison up to 20 can be executed as follows.

$$verificationDCC[Ex1Ex2DCC][Ex1C, Ex2C][p1, p2][\tilde{p}][20]$$

And, we should have a $True$ as answer. Of course, this does not constitute a proof of the direct formula.

3.6 *Mathematica* demonstration of closed formulas for the connection coefficients

The mathematic demonstration corresponds to show that the direct closed formula for the connection coefficients is a solution of the recurrence relation (13), for all non negative integers n and m such that $m \leq n$.

In principle, this proof can be done totally in *Mathematica* using the next command, if the formulas for the recurrence coefficients $\{\beta_n\}$, $\{\gamma_{n+1}\}$ and $\{\tilde{\beta}_n\}$, $\{\tilde{\gamma}_{n+1}\}$ are available, for all n, that is, for n symbolic, and the commands $FullSimplify$ or $FunctionExpand$

are able to make the necessary simplifications.

$$
\begin{aligned}
&demonstrationDCC[dcc\_][rc\_, rct\_][p\_\_\_][pt\_\_\_][n\_Symbol, m\_Symbol] := \\
&FullSimplify[\\
&\quad dcc[p][pt][n,m] - (rct[pt][m][[1]] - rc[p][n-1][[1]]) * dcc[p][pt][n-1][m] + \\
&\quad rc[p][n-1][[2]] * dcc[p][pt][n-2,m] - \\
&\quad rct[pt][m+1][[2]] * dcc[p][pt][n-1,m+1] - dcc[p][pt][n-1,m-1], \\
&\quad Assumptions \rightarrow \\
&\quad And[Element[n, Integers], Element[m, Integers], n >= 0, m >= 0, m <= n] \\
&\qquad\qquad] === 0;
\end{aligned}
$$

Remark that n and m must be symbols in the calling statement otherwise *demonstrationDCC* does nothing. Note that the assumptions inform that the symbols n and m represent integers such that $0 \leq m \leq n$. These *Assumptions* can not be necessary to accomplish the demonstrations. *demonstrationDCC* is , in fact, a symbolic translation of the recurrence relation (13).

In the case of the examples $Ex1$ and $Ex2$ the demonstration corresponds to get $True$ as answer to the following entry.

$$demonstrationDCC[Ex1Ex2CC][Ex1C, Ex2C][p1, p2][\tilde{p}][n, m]$$

3.7 Main commands description

In this section, we furnish a list with a description of the main commands implemented in the general case, they are: CC, MOP, $verificationRCC$, $verificationDCC$ and *demonstrationDCC*. For each command we refer the name, brief description, arguments, *Mathematica* commands and other commands employed in the implementation and the result produced.

- **Arguments** rc, rct, p and pt of the commands listed in the sequel.

  - rc and rct are the names of the commands that define the recurrence coefficients $\{\beta_n, \gamma_n\}$ and $\{\tilde{\beta}_n, \tilde{\gamma}_n\}$ of the two polynomials sequences P and $\tilde{P}$; they must be symbols.

  - rc and rct must be defined like:

  $rc[p\_\_\_][n\_] := ...[...; Return[\{\beta_n, \gamma_n\}]]$; $rct[pt\_\_\_][n\_] := ...[...; Return[\{\tilde{\beta}_n, \tilde{\gamma}_n\}]]$;

  - p and pt are the sequences of parameters of P and $\tilde{P}$.

- $CC[rc, rct][p][pt][n, m]$

  Description:

  CC computes recursively the connection coefficient, $\lambda_{n,m} := \lambda_{n,m}(P \leftarrow \tilde{P})$, defined in (9), using the initial conditions (10)-(12) and the recurrence relation (13).

Arguments:

- rc, rct, p and pt described before.

- n and m must be integers.

Mathematica commands used:

- $Factor$ or $Together$, $FullSimplify$.

Result:

- $\lambda_{n,m}$.

- $MOP[rc][p][n, x]$

  Description:

  MOP computes recursively the monic orthogonal polynomial, $P_n(x)$, of degree n in the variable x, using the initial conditions (2) and the recurrence relation (3).

  Arguments:

  - rc and p described before.

  - n must be an integer.

  - x should be a symbol or a numeric expression.

  Mathematica commands used:

  - $Collect$, $Factor$ or $Together$, $Simplify$ or $FullSimplify$.

  Result:

  - $P_n(x)$.

- **Argument** $nmax$ of the commands listed in the sequel.

  - $nmax$ must be a non negative integer.

- $verificationRCC[rc, rct][p][pt][nmax]$

  Description:

  $verificationRCC$ makes a verification of the first connection coefficients, $\lambda_{n,m} := \lambda_{n,m}(P \leftarrow \tilde{P})$, computed recursively by the CC command up to an index $n = nmax$. $verificationRCC$ is based on the definition (9) of the connection coefficients.

  Arguments:

  - rc, rct, p, pt and $nmax$ described before.

  Mathematica commands used:

  - $FullSimplify$.

  Other commands used:

  - CC and MOP.

Result:

- Returns $True$ if the the verification is correct, returns $False$ otherwise.

- **Argument** *dcc* of the commands listed in the sequel.

  - *dcc* is the name of the command that implements the direct closed formula; it must be a symbol.

  - *dcc* must be defined like $dcc[p\_\_\_][pt\_\_\_][n\_, m\_] := ...[...; Return[\lambda_{n,m}]]$;

- $verificationDCC[dcc][rc, rct][p][pt][nmax]$

  Description:

  $verificationDCC$ makes a comparison between the values of the connection coefficients, $\lambda_{n,m} := \lambda_{n,m}(P \leftarrow \tilde{P})$, computed by the CC command and the ones computed using a direct closed formula, this, up to the index $n = nmax$.

  Arguments:

  - *dcc*, *rc*, *rct*, *p*, *pt* and *nmax* described before.

  Mathematica commands used:

  - $FullSimplify$.

  Other commands used:

  - CC.

  Result:

  - Returns $True$ if the verification is correct, returns $False$ otherwise.

- $demonstrationDCC[dcc][rc, rct][p][pt][n, m]$

  Description:

  $demonstrationDCC$ tries to demonstrate the direct closed formula for the connection coefficients $\lambda_{n,m} := \lambda_{n,m}(P \leftarrow \tilde{P})$ for every n and m.

  Arguments:

  - *dcc*, *rc*, *rct*, *p* and *pt* described before.

  - n and m are symbols and represent non negative integers such that $0 \leq m \leq n$.

  Mathematica commands used:

  - $FullSimplify$ with $Assumptions$.

  Result:

  - Returns $True$ if *Mathematica* achieves the demonstration, returns $False$ otherwise.

4 Test examples

4.1 Charlier polynomials

Let us see how the commands we have developed work and what results they produce in the simple case of the classical discrete monic Charlier polynomials $\{P_n(\alpha,.)\}_{n\geq 0}$ with parameter α [7]. The Charlier recurrence coefficients

$$\beta_n(\alpha) = n+\alpha \ , \ n\geq 0 \ ; \ \gamma_n(\alpha) = n\alpha \ , \ n\geq 1 \ , \ \alpha \neq 0, \tag{31}$$

are implement in the following command.

$$\begin{aligned} &CharlierC[\alpha\_][n\_] := CharlierC[\alpha][n] = \\ &\quad If[\ And[NumericQ[n], n<0],\ Return[\{0,0\}],\ Return[\{n+\alpha, \alpha * n\}]\ ]; \end{aligned}$$

The monic Charlier polynomials are defined, using the MOP command, by

$$CharlierP[\alpha\_][n\_, x\_] := CharlierP[\alpha][n,x] = MOP[CharlierC][\alpha][n,x];$$

The connection coefficients $\lambda_{n,m} := \lambda_{n,m}(P(\alpha;-) \leftarrow P(\tilde{\alpha};-))$ are computed recursively up to $n=6$, for example, by the next calling statement of the CC command.

$$\begin{aligned} &In[\ ] := Table[\ CC[CharlierC, CharlierC][\alpha][\tilde{\alpha}][n,m], \{n,0,6\}, \{m,0,n\}\ ]// \\ &\qquad TableForm \\ &Out[\ ]TableForm = \end{aligned}$$

$$\begin{array}{llllll} 1 & & & & & \\ -\alpha+\tilde{\alpha} & 1 & & & & \\ (\alpha-\tilde{\alpha})^2 & -2(\alpha-\tilde{\alpha}) & 1 & & & \\ -(\alpha-\tilde{\alpha})^3 & 3(\alpha-\tilde{\alpha})^2 & -3(\alpha-\tilde{\alpha}) & 1 & & \\ (\alpha-\tilde{\alpha})^4 & -4(\alpha-\tilde{\alpha})^3 & 6(\alpha-\tilde{\alpha})^2 & -4(\alpha-\tilde{\alpha}) & 1 & \\ -(\alpha-\tilde{\alpha})^5 & 5(\alpha-\tilde{\alpha})^4 & -10(\alpha-\tilde{\alpha})^3 & 10(\alpha-\tilde{\alpha})^2 & -5(\alpha-\tilde{\alpha}) & 1 \\ (\alpha-\tilde{\alpha})^6 & -6(\alpha-\tilde{\alpha})^5 & 15(\alpha-\tilde{\alpha})^4 & -20(\alpha-\tilde{\alpha})^3 & 15(\alpha-\tilde{\alpha})^2 & -6(\alpha-\tilde{\alpha}) \quad 1 \end{array}$$

Now, we can verify these results and the next ones up to $nmax = 20$, for example, calling

$$In[\ ] := Timing[\ verificationRCC[CarlierC, CharlierC][\alpha,\tilde{\alpha}][20]\ ]$$

and we get the answer

$$Out[\ ] = \{278.7, True\}$$

Note that $Timing[expr]$ evaluates $expr$, and returns a list of the time in seconds used, together with the result obtained [19, 20].

The observation of the above results getting by the CC commands allows us to infer the following direct closed formula for the connection coefficients

$$\lambda_{n,m} = (-1)^{n-m}\binom{n}{m}(\alpha-\tilde{\alpha})^{n-m},\ 0\leq m\leq n,\ n\geq 0, \tag{32}$$

which can be implement in a command as follows

$$\begin{aligned} &CharlierDCC[\alpha\_][\alpha t\_][n\_,m\_] := \\ &\qquad (-1)\hat{}(n-m) * Binomial[n,m] * (\alpha-\alpha t)\hat{}(n-m); \end{aligned}$$

In order to compare the results given by this command with those produced by CC up to $nmax = 20$, for example, we do

$$\begin{aligned} &In[\,] := Timing[\, verificationDCC[CharlierDCC][CharlierC, CharlierC][\alpha][\tilde{\alpha}][20]\,] \\ &Out[\,] = \{0.02, True\} \end{aligned}$$

The mathematical demonstration of the formula (32) is achieved in *Mathematica* doing

$$\begin{aligned} &In[\,] := Timing[\\ &\qquad demonstrationDCC[CharlierDCC][CharlierC, CharlierC][\alpha][\tilde{\alpha}][n,m]\,] \\ &Out[\,] = \{0.5, True\} \end{aligned}$$

We remember that the formula (32) is well known and can be found in several references; see, among others, [1, 11].

4.2 Canonical and Laguerre polynomials

Let us express the canonical sequence in terms of the classical monic Laguerre polynomials. For that purpose, we need to recall the Laguerre recurrence coefficients [7],

$$\beta_n(\alpha) = 2n+\alpha+1\ ,\ \gamma_{n+1}(\alpha) = (n+1)(n+\alpha+1),\ \alpha\neq -n,\ n\geq 0, \tag{33}$$

which can be implemented in the following command.

$$\begin{aligned} &LaguerreC[\alpha\_][n\_] := LaguerreC[\alpha][n] = \\ &If[\, And[NumericQ[n], n<0],\ Return[\{0,0\}],\ Return[\{2*n+\alpha+1, n*(n+\alpha)\}]\,]; \end{aligned}$$

The recurrence coefficients of the canonical sequence given in (8) and the canonical polynomials are easily translated as follows

$$CanonicalC[\,][n\_] := CanonicalC[\,][n] = \{0,0\};$$

$$CanonicalP[\,][n\_, x\_] := CanonicalP[\,][n,x] = MOP[CanonicalC][][n,x];$$

The connection coefficients $\lambda_{n,m} := \lambda_{n,m}(X \leftarrow P(\tilde{\alpha}; -))$ are computed recursively up to $n = 4$, for example, by the next calling statement of the CC command.

$In[\ ] := Table[\ CC[CanonicalC, LaguerreC][\ ][\tilde{\alpha}][n, m], \{n, 0, 4\}, \{m, 0, n\}\ ]//$
$TableForm$

$Out[\ ]TableForm =$

1				
$(1+\tilde{\alpha})$	1			
$(1+\tilde{\alpha})(2+\tilde{\alpha})$	$2(2+\tilde{\alpha})$	1		
$(1+\tilde{\alpha})(2+\tilde{\alpha})(3+\tilde{\alpha})$	$3(2+\tilde{\alpha})(3+\tilde{\alpha})$	$3(3+\tilde{\alpha})$	1	
$(1+\tilde{\alpha})(2+\tilde{\alpha})(3+\tilde{\alpha})(4+\tilde{\alpha})$	$4(2+\tilde{\alpha})(3+\tilde{\alpha})(4+\tilde{\alpha})$	$6(3+\tilde{\alpha})(4+\tilde{\alpha})$	$4(4+\tilde{\alpha})$	1

Now, we can verify these results and the next ones up to $nmax = 20$, for example, calling

$In[\ ] := Timing[\ verificationRCC[CanonicalC, LaguerreC][\ ][\tilde{\alpha}][20]\ ]$

and we get the answer

$Out[\ ] = \{402.41, True\}$

The observation of the above table allows us to infer the following direct closed formula for the connection coefficients

$$\lambda_{n,m} = \binom{n}{m} \prod_{k=0}^{n-m-1} (\tilde{\alpha} + n - k),\ 0 \le m \le n,\ n \ge 0, \tag{34}$$

which can be implement in a command as follows

$$CanonicalLaguerreDCC[\ ][\alpha t\_][n\_, m\_] := Binomial[n, m] * \prod_{k=0}^{n-m-1} (\alpha t + m - k);$$

In order to compare the results given by this command with those produced by CC up to $nmax = 20$, for example, we do

$In[\ ] := Timing[$
$verificationDCC[\ CanonicalLaguerreDCC[CanonicalC, LaguerreC][\ ][\tilde{\alpha}][20]\ ]\ ]$
$Out[\ ] = \{0.02, True\}$

The mathematical demonstration of the formula (34) is achieved in *Mathematica* doing

$In[\ ] := Timing[$
$demonstrationDCC[\ CanonicalLaguerreDCC[CanonicalC, LaguerreC][\ ][\tilde{\alpha}][n, m]]]$
$Out[\ ] = \{0.32, True\}$

We remember that the formula (34) is well known and can be found in several references; see, among others, [4, 5, 2, 15]. About the table of recursive results, when $\tilde{\alpha} = 0$, see [9].

4.3 Commands description of test examples

In this section, we give a descriptive list, in the same terms as before, of the commands implemented in the section of test examples.

- **Commands for the Charlier family:** $CharlierC$, $CharlierP$ and $CharlierDCC$.

  Argument α of the commands listed in the sequel.

  - α is a parameter ($\alpha \neq 0$).

  - If $\alpha = 0$, then Charlier polynomials are not regular, $\gamma_n = 0, n >= 1$.

  - α should be a symbol or a numeric expression.

- $CharlierC[\alpha][n]$

  Description:

  $CharlierC[\alpha][n]$ is the n-th recurrence coefficients, $\{\beta_n, \gamma_n\}$, of the monic Charlier polynomials (31).

  Arguments:

  - n should be a symbol or an integer.

  Result:

  - If $n > 0$, $\{\beta_n, \gamma_n\}$. If n=0, $\{\beta_n, 0\}$. If $n < 0$, $\{0, 0\}$.

- $CharlierP[\alpha][n, x]$

  Description:

  $CharlierP[\alpha][n, x]$ is the monic Charlier polynomial of parameter α of degree n in the variable x: $P_n(x)$.

  Arguments:

  - n must be an integer.

  - x should be a symbol or a numeric expression.

  Commands used:

  - *CharlierC*, *MOP*.

  Result:

  - $P_n(x)$

- $CharlierDCC[\alpha][\tilde{\alpha}][n, m]$

  Description:

  $CharlierDCC[\alpha][\tilde{\alpha}][n, m]$ computes the connection coefficient $\lambda_{n,m} := \lambda_{n,m}(P(\alpha, -) \leftarrow \tilde{P}(\tilde{\alpha}, -))$, where P notes the monic Charlier polynomials, using the direct closed formula (32) inferred from the above results produced by the CC command.

  Arguments:

- α and $\tilde{\alpha}$ are the parameters of P and $\tilde{P}$.

- n and m should be symbols or integers.

Result:

- $\lambda_{n,m}$.

- **Commands for the canonical family:** *CanonicalC* and *CanonicalP*.

- *CanonicalC*[][n]

  Description:

  CanonicalC[][n] is the n-th recurrence coefficients $\{\beta_n, \gamma_n\}$ of the canonical polynomials (8).

  Arguments:

  - n should be a symbol or an integer.

  Result:

  - $\{0, 0\}$.

- *CanonicalP*[][n, x]

  Description:

  CanonicalP[][n, x] is the canonical polynomial of degree n in the variable x.

  Arguments:

  - n must be an integer.

  - x should be a symbol or a numeric expression.

  Commands used:

  - *CanonicalC, MOP.*

  Result:

  - If $n >= 0$, x^n. If $n < 0$, 0.

- **Commands for the Laguerre family:** *LaguerreC*, *LaguerreP* and *CanonicalLaguerreDCC*.

  Argument α of the commands listed in the sequel.

  - α is a parameter ($\alpha \neq -n, n > 0$).

  - If $\alpha = -n, n > 0$, then Laguerre polynomials are not regular, $\gamma_n = 0, n \geq 1$.

  - α should be a symbol or a numeric expression.

- *LaguerreC*[α][n]

  Description:

  LaguerreC[α][n] is the n-th recurrence coefficients, $\{\beta_n, \gamma_n\}$, of the monic Laguerre polynomials (33).

Arguments:

- n should be a symbol or an integer.

Result:

- If $n > 0$, $\{\beta_n, \gamma_n\}$. If $n = 0$, $\{\beta_n, 0\}$. If $n < 0$, $\{0, 0\}$.

- $LaguerreP[\alpha][n, x]$

Description:

$LaguerreP[\alpha][n, x]$ is the monic Laguerre polynomial of parameter α of degree n in the variable x: $P_n(x)$.

Arguments:

- n must be an integer.

- x should be a symbol or a numeric expression.

Commands used:

- *LaguerreC, MOP.*

Result:

- $P_n(x)$

- $CanonicalLaguerreDCC[][\tilde{\alpha}][n, m]$

Description:

$CanonicalLaguerreDCC[][\tilde{\alpha}][n, m]$ computes the connection coefficient $\lambda_{n,m} := \lambda_{n,m}(X \leftarrow P(\tilde{\alpha}), -))$, where X notes the canonical polynomials and P notes the monic Laguerre polynomials, using the direct closed formula (34) inferred from the above results produced by the CC command.

Arguments:

- $\tilde{\alpha}$ is the parameter of $\tilde{P}$.

- n and m should be symbols or integers.

Result:

- $\lambda_{n,m}$

5 Symbolic computation of connection coefficients in the symmetrical case

5.1 *Mathematica* demonstration of closed formulas for the connection coefficients

It is quite clear that the general commands CC, $verificationRCC$ and $verificationDCC$ work perfectly in the case of the two polynomials sequences are symmetric. Nevertheless, we could develop more efficient commands due to the simplifications introduced by

the equalities (14), creating a command named $SymCC$ for implement the recurrence relations (14)-(16) and the commands $verificationSymRCC$ and $verificationSymDCC$ for accomplish the corresponding verifications. We have not do that because the gain in second times execution is very small.

On the other hand, in the symmetrical case, the command $demonstrationDCC$ can not work, because, often, the direct closed formulas for the connection coefficients are given separately for even and odd integer indexes, that is, we have different formulas for $\lambda_{2n,2m}$ and $\lambda_{2n+1,2m+1}$. Therefore, the commands that implement these formulas must be defined for the arguments $[2*n\_, 2*m\_]$ and $[2*n\_+1, 2*m\_+1]$. In calling statements, the arguments corresponding to the indexes $2*n+2, 2*m-2$ and $2*n-1, 2*m-1$, for example, should be given as $[2*(n-1)+1, 2*(m-1)+1]$ and $[2*(n+1), 2*(m-1)]$ in order to match with the definitions.

We next translate the command $demonstrationSymDCC$, which allows to show that the direct closed formulas, which name enter as the argument $symdcc$, satisfy the recurrence relations (15) and (16), for every n and m.

$$
\begin{aligned}
&demonstrationSymDCC[symdcc\_][rc\_, rct\_][p\_\_\_][pt\_\_\_][n\_Symbol, m\_Symbol] := \\
&And[FullSimplify[\\
&\qquad symdcc[p][pt][2*n, 2*m] + \\
&\qquad rc[p][2*(n-1)+1][[2]] * symdcc[p][pt][2*(n-1), 2*m] - \\
&\qquad rct[pt][2*m+1][[2]] * symdcc[p][pt][2*(n-1)+1, 2*m+1] - \\
&\qquad symdcc[p][pt][2*(n-1)+1, 2*(m-1)+1], \\
&\qquad Assumptions \rightarrow \\
&\qquad And[Element[n, Integers], Element[m, Integers], n >= 0, \\
&\qquad m >= 0, m <= n] \\
&\qquad\quad] === 0, \\
\\
&\quad FullSimplify[\\
&\qquad\quad symdcc[p][pt][2*n+1, 2*m+1] + \\
&\qquad\quad rc[p][2*n][[2]] * symdcc[p][pt][2*(n-1)+1, 2*m+1] - \\
&\qquad\quad rct[pt][2*(m+1)][[2]] * symdcc[p][pt][2*n, 2*(m+1)] - \\
&\qquad\quad symdcc[p][pt][2*n, 2*m], \\
&\qquad\quad Assumptions \rightarrow \\
&\qquad\quad And[Element[n, Integers], Element[m, Integers], n >= 0, \\
&\qquad\quad m >= 0, m <= n] \\
&\qquad\qquad] === 0 \\
&]];
\end{aligned}
$$

These $Assumptions$ can not be necessary to accomplish the symplifications.

In the case of the symmetric examples $SymEx1$ and $SymEx2$ the demonstration corresponds to get $True$ as answer to the next entry, supposing that $SymEx1SymEx2DCC$ is the name of the command that translate the direct closed formulas of $\lambda_{2n,2m}$ and $\lambda_{2n+1,2m+1}$, and $SymEx1C$ and $SymEx2C$ are the names of the recurrence coefficients.

$demonstrationSymDCC[SymEx1SymEx2DCC][SymEx1C, SymEx2C][p1, p2][\tilde{p}][n, m]$

5.2 Main command description

As before, we give a description of the main command implemented in the symmetrical case: $demonstrationSymDCC$.

- $demonstrationSymDCC[symdcc][rc, rct][p][pt][n, m]$

  Description:

  $demonstrationSymDCC$ tries to demonstrate the direct closed formulas for the connection coefficients $\lambda_{n,m} := \lambda_{n,m}(P \leftarrow \tilde{P})$, when P and $\tilde{P}$ are symmetrical polynomials sequences implementing the recurrence relations (15) and (16).

  Arguments:

  - *symdcc* is the name of the command that implements the direct closed formulas, it must be a symbol.

  - *rc* and *rct* are the names of the commands that define the recurrence coefficients, $\{0, \gamma_n\}$ and $\{0, \tilde{\gamma}_n\}$, of the two symmetrical polynomials sequences P and $\tilde{P}$; they must be symbols.

  - *rc* and *rct* must be defined like:

  $rc[p\_\_\_][n\_] := ...[...; Return[\{0, \gamma_n\}]]$; $rct[pt\_\_\_][n\_] := ...[...; Return[\{0, \tilde{\gamma}_n\}]]$;

  - *p* and *pt* are the sequences of parameters of P and $\tilde{P}$.

  - n and m are symbols and represent non negative integers such that $0 <= m <= n$.

  Mathematica commands used:

  - $FullSimplify$ with $Assumptions$.

  Result:

  - Returns $True$ if *Mathematica* achieves the demonstration, returns $False$ otherwise.

5.3 Test example - generalized Hermite polynomials

Let us consider the semi-classical generalized monic Hermite sequence $\{P_n(\mu; .)\}_{n\geq 0}$ with parameter μ, which is symmetric. When $\mu = 0$, we recover the classical Hermite sequence.

The recurrence coefficients [7]

$$\beta_n = 0, n \geq 0 \ ; \ \gamma_n := \gamma_n(\mu) = \frac{1}{2}\left(n + \mu(1 + (-1)^{n-1})\right), \mu \neq -n - \frac{1}{2}, \ n \geq 1, \qquad (35)$$

are implemented in the following command.

$GenHermiteC[\mu\_][n\_] := GenHermiteC[\mu][n] =$
$If[\ And[NumericQ[n], n <= 0],\ Return[0, 0],\ Return[0, (n + \mu * (1 + (-1)^{(n-1)}))/2]\ ];$

We can compute recursively the connection coefficients $\lambda_{n,m} := \lambda_{n,m}(P(\mu;-) \leftarrow P(\tilde{\mu};-))$ up to $n = 5$, for example, by the next call of the CC command.

$In[\ ] := Table[CC[GenHermiteC, GenHermiteC][\mu][\tilde{\mu}][n, m], \{n, 0, 5\}, \{m, 0, n\}]//$
$TableForm$
$Out[\ ]//TableForm =$

1					
0	1				
$-\mu+\tilde{\mu}$	0	1			
0	$-\mu+\tilde{\mu}$	0	1		
$(-\mu+\tilde{\mu})(1+\mu-\tilde{\mu})$	0	$-2(\mu-\tilde{\mu})$	0	1	
0	$(-\mu+\tilde{\mu})(1+\mu-\tilde{\mu})$	0	$-2(\mu-\tilde{\mu})$	0	1

We can verify these results and the next ones up to $nmax = 20$, for example, calling

$In[\ ] := Timing[\ verificationRCC[GenHermiteC, GenHermiteC][\mu, \tilde{\mu}][20]\ ]$

and we get the answer

$Out[\ ] = \{90.22, True\}$

From the above table and some more results, we can easily infer the direct closed formulas for the connection coefficients; they are

$$\lambda_{2n,2m} = (-1)^{n+m}\binom{n}{m}\prod_{k=0}^{n-m-1}(\mu-\tilde{\mu}+k)\ , \tag{36}$$

$$\lambda_{2n+1,2m+1} = \lambda_{2n,2m}\ ; \tag{37}$$

which are implemented as follows.

$GenHermiteDCC[\mu\_][\mu t\_][n\_/; And[IntegerQ[n], n >= 0],$
$\qquad m\_/; And[IntegerQ[m], m >= 0]] :=$
$GenHermiteDCC[\mu][\mu t][n, m] =$
$Module[\{n1, m1\},$
$If[\ Or[\ And[EvenQ[n], OddQ[m]], And[OddQ[n], EvenQ[m]]\ ], Return[0]\ ];$
$n1 = Quotient[n, 2];\ m1 = Quotient[m, 2];$
$Return[$
$(-1)\hat{}(n1 - m1) * Binomial[n1, m1] * Product[\mu - \mu t + k, \{k, 0, n1 - m1 - 1\}]]\ ];$

We inform that $EvenQ[exp]$ gives $True$, if exp is an even integer, and $False$ otherwise; $OddQ$ works in a similar way; $Quotient[n, m]$ returns the integer quotient of n and m [19, 20]. We remark that these commands could not furnish the desired answer if n or m are symbols. In principle, $EvenQ[exp]$ and $OddQ[exp]$ return always $False$, if exp is a symbol and $Quotient[n, m]$ gives no answer if n or m are symbols. Therefore these commands should work only for non negative values of n and m as is insured by the restrictions following the corresponding pattern arguments.

In order to compare the results produced by $GenHermiteDCC$ with those obtained from CC up to $nmax = 20$, for example, we do

$$In[\,] := Timing[\, verificationDCC[GenHermiteDCC] \\ [GenHermiteC, GenHermiteC][\mu][\tilde{\mu}][20]\,] \\ Out[\,] = \{0.03, True\}$$

In order to demonstrate the formulas (36) and (37) for every n and m, through the command $demonstrationSymDCC$, the above definition of $GenHermiteDCC$ is not adequate and we should add the next definitions for $GenHermiteDCC$ that will be called in $demonstrationSymDCC$ for n and m symbolic. We alert that a $Symbol$ restriction following the patterns of the arguments n and m must not figure in these definitions and that they do not allow computations for fixed values of n and m.

$$GenHermiteDCC[\mu\_][\mu t\_][2 * n\_, 2 * m\_] := \\ (-1)\hat{}(n - m) * Binomial[n, m] * Product[\mu - \mu t + i, \{i, 0, n - m - 1\}]; \\ GenHermiteDCC[\mu\_][\mu t\_][2 * n\_ + 1, 2 * m\_ + 1] := \\ GenHermiteDCC[\mu][\mu t][2 * n, 2 * m];$$

Therefore the demonstration is achieved doing

$$In[\,] := Timing[\, demonstrationSymDCC[GenHermiteDCC] \\ [GenHermiteC, GenHermiteC][\mu][\tilde{\mu}][n, m]\,] \\ Out[\,] = \{0.66, True\}$$

We remember that the formulas (36) and (37) are well known and they are available in several references; see, among others, [15].

5.4 Commands description of test example

In this section, we give a descriptive list, in the same terms as before, of the commands implemented in order to test the symmetrical case of generalized Hermite family, they are: $GenHermiteC$, $GenHermiteP$ and $GenHermiteDCC$.

- **Argument** μ of the commands listed in the sequel.

  - μ is a parameter ($\mu \neq -n - \frac{1}{2}, n >= 0$).

- If $\mu = -n - \frac{1}{2}, n >= 0$, then generalized Hermite polynomials are not regular, $\gamma_{2n+1} = 0, n >= 1$.

- μ should be a symbol or a numeric expression.

- $GenHermiteC[\mu][n]$

  Description:

  $GenHermiteC[\mu][n]$ is the n-th recurrence coefficients, $\{0, \gamma_n\}$, of the monic generalized Hermite polynomials (35).

  - n should be a symbol or an integer.

  Result:

  - If $n > 0$, $\{0, \gamma_n\}$. If $n <= 0$, $\{0, 0\}$.

- $GenHermiteP[\mu][n, x]$

  Description:

  $GenHermiteP[\mu][n, x]$ is the monic generalized Hermite polynomial of parameter μ of degree n in the variable x: $P_n(x)$.

  Arguments:

  - n must be an integer.

  - x should be a symbol or a numeric expression.

  Commands used:

  - *GenHermiteC, MOP.*

  Result:

  - $P_n(x)$

- $GenHermiteDCC[\mu][\tilde{\mu}][n, m]$

  Description:

  $GenHermiteDCC[\mu][\tilde{\mu}][n, m]$ computes the connection coefficients $\lambda_{n,m} := \lambda_{n,m}(P(\mu, -) \leftarrow P(\tilde{\mu}, -))$, where P notes the monic generalized Hermite polynomials, using the direct closed formulas (36) and (37) inferred from the above results produced by the CC command.

  Arguments:

  - μ and $\tilde{\mu}$ are the parameters of P and $\tilde{P} = P(\tilde{\mu}, -)$.

  - n and m must be **non negative integers.**

  Results:

  - $\lambda_{2n,2m}$, $\lambda_{2n+1,2m+1}$.

- $GenHermiteDCC[\mu][\tilde{\mu}][n,m]$

  Description:

  $GenHermiteDCC[\mu][\tilde{\mu}][n,m]$ translates the definitions (36) and (37) of the connection coefficients $\lambda_{n,m} := \lambda_{n,m}(P(\mu,-) \leftarrow P(\tilde{\mu},-))$, where P notes the monic generalized Hermite polynomials, for n and m symbolic, in order to be used in calling statements of the $demonstrationSymCC$ command.

  Arguments:

  - μ and $\tilde{\mu}$ are the parameters of P and $\tilde{P} = P(\tilde{\mu},-)$.

  - n and m should be **symbols**.

  Results:

  - $\lambda_{2n,2m}$, $\lambda_{2n+1,2m+1}$.

6 Conclusions

There are two difficult steps in the methodology exposed here and in [15, 16, 17]. The first one consists to infer the closed formula for the CC from enough data produced and treated by a symbolic language like *Mathematica*. The second one is to accomplish the demonstration of the model, which can be done completely via *Mathematica* or only in part with the help of *Mathematica*.

In spite of some limitations, we know that this method can be useful to treat several other examples of CC. Furthermore, the implementation principles studied here can be applied in a similar way to other situations in the branch of orthogonal polynomials and in mathematics.

References

[1] I. Area, E. Godoy, A. Ronveaux, A. Zarzo, Minimal recurrence relations for connection coefficients between classical orthogonal polynomials: Discrete case, J. Comput. Appl. Math., 89 (1998) 309-325.

[2] I. Area, E. Godoy, A. Ronveaux, A. Zarzo, Inversion Problems in the q-Hahn Tableau, J. Symbolic Computation, 28 (1999) 767-776.

[3] I. Area, E. Godoy, A. Ronveaux, A. Zargo, Solving connection and linearization problems within the Askey scheme and its $q-$analogue via inversion formulas, J. Comput. Appl. Math., 133, 151-162, 2001.

[4] R. Askey, Orthogonal Polynomials and Special Functions. CBMS-NSF Regional Conference Series, Appl. Math., 21, SIAM, Philadelphia, PA, 1975.

[5] G. E. Andrews, R. Askey, R. Roy, Special Functions, Cambridge University Press, 71, 1999.

[6] H. Chagarra, W. Koepf, On linearization and connection coefficients for generalized Hermite polynomials, J. Comput. Appl. Math., in press.

[7] T.S. Chihara, An Introduction to Orthogonal Polynomials, Gordon and Breach, New York,1978.

[8] E. Godoy, I. Area, A. Ronveaux, A. Zarzo, Minimal recurrence relations for connection coefficients between classical orthogonal polynomials: Continuous case, J. Comput. Appl. Math., 84(2) (1997) 257-275.

[9] U. W. Hochstrasser, Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables, In M. Abramowitz, I. A. Stegun, eds., N.Y., 1970.

[10] S. Lewanowicz, Quick construction of recurrence relations for the Jacobi coefficients, J. Comput. Appl. Math., 43 (1992) 355-372.

[11] S. Lewanowicz, Recurrence relations for the connection coefficients of orthogonal polynomials of a discrete variable, J. Comput. Appl. Math., 76 (1996) 213-229.

[12] Koepf, Wolfram and Swarttouw, Ren: "CAOP: Computer Algebra and Orthogonal Polynomials". http://pool-serv1.mathematik.uni-kassel.de/CAOP

[13] P. Maroni, Variations around classical orthogonal polynomials. Connected problems, J. Comput. Appl. Math., 48, 133-155, 1993.

[14] P. Maroni, Fonctions eulériennes. Polynômes orthogonaux classiques. Techniques de l'Ingénieur, traité Généralités (Sciences Fondamentales), 1994.

[15] P. Maroni, Z. da Rocha, Connection coefficients for orthogonal polynomials and the canonical sequence, Preprints CMUP, Centro de Matemática da Universidade do Porto, 29, 1-18, 2007. http://cmup.fc.up.pt/cmup/v2/frames/publications.htm

[16] P. Maroni, Z. da Rocha, Connection coefficients between orthogonal polynomials and the canonical sequence: an approach based on symbolic computation, Numerical Algorithms, 47-3 (2008) 291-314.

[17] P. Maroni, Z. da Rocha, Connection coefficients for orthogonal polynomials: symbolic computations, verifications and demonstrations in the *Mathematica* language, submitted for publication, 2010.

[18] A. Ronveaux, A. Zarzo, E. Godoy, Recurrence relation for connection coefficients between two families of orthogonal polynomials, J. Comput. Appl. Math. 62 (1995) 67-73.

[19] Stephen Wolfram, The Mathematica Book, 4th ed., Wolfram Media/Cambridge University Press, 1999.

[20] Wolfram *Mathematica* Virtual Book, 2010. www.wolfram.com