FACTA UNIVERSITATIS (NIŠ)
SER. MATH. INFORM. 19 (2004), 17–36

THE MATHEMATICA PACKAGE "OrthogonalPolynomials" *

Aleksandar S. Cvetković and Gradimir V. Milovanović

Dedicated to Prof. G. Mastroianni for his 65th birthday

Abstract. In this paper we give basic concepts of the *Mathematica* package "OrthogonalPolynomials". The package "OrthogonalPolynomials" emerged from the need to implement basic algorithms from the theory of orthogonal polynomials to the *Mathematica* platform which offers, in our opinion the highest computing possibilities. Package performs construction of orthogonal polynomials and quadrature formulas. Also, the package has implemented almost all the classes of orthogonal polynomials studied up to date. For the detailed exposition of the material presented in this paper we refer to [3], here we present only basic characteristics of the package.

1. Introduction

In the first section we present possibilities of the programming language *Mathematica*. In the second section we present the basic theory of orthogonal polynomials and quadrature formulas. We present only the basic theory which is needed for the understanding of construction algorithms. In the third section we present symbolic implementation of the algorithms from the orthogonal polynomial theory. In the fourth section we present numerical implementation of algorithms. In the fifth section we present supported classes of orthogonal polynomials studied up to date.

Received April 10, 2004.
2000 *Mathematics Subject Classification.* Primary 33C45; Secondary 65D30, 65D32.
*The authors were supported in part by the Serbian Ministry of Science and Environmental Protection (Project #2002: Applied Orthogonal Systems, Constructive Approximation and Numerical Methods).

2. The Package Mathematica

Mathematica package [19] is very suitable for the implementation of the algorithms we are studying for two reasons. The first reason is orientation of the *Mathematica* in supporting symbolical computing, which is offered by very few other packages. The second reason is possibility of working with numbers of theoretically infinite precision.

Possibility of symbolic computation is very valuable for any scientific discipline, and those are especially important for the theory of orthogonal polynomials. All computations in the theory of orthogonal polynomials are quite complicated and usually several free variables are included so that, if performed by hand, computations are messy. Almost all algorithms which are used are nonlinear and rational, i.e., if some nonlinearity is included it is of polynomial type, therefore, the most valuable functions for symbolic computations are those operating on rational functions. Complete information about those functions can be found in [19]. We mention only functions which are the most important for our application

- *Together* function which adds rational expressions,
- *Cancel* function which is used for cancellation of the common factors in the rational expression,
- *Factor* function which is used to factorize expressions in the ring of integers.

There are still more functions which can be used to manipulate with rational expressions. Some of them are also presented in [3].

For the manipulation with expressions containing different special functions we can use functions like *Simplify* and *FullSimplify*. These two functions transform the starting expression into the expression which should be the 'simplest'. The term being simplest is measured in the context of the number of special functions employed, and it is really the weighted sum of the symbols appearing in the expression. Applying all identities, which are known to the *Mathematica*, *Mathematica* keeps measuring simplicity of the expressions it produces. Functions simply return the expression with the smallest simplicity, i.e., having the smallest weighted sum of the symbols appearing. Weights of the symbols appearing are not suitable for all the possible applications, but there is a possibility left to the user to specify the transformation rules and the algorithms most suitable for the simplification.

Mathematica also has possibility of implementation of the transformation rules for the symbols which are built-in or are defined by the user. We mention that it is needed to define new functions performing transformations, if we want to apply the new set of transformation rules using functions *Simplify* and *FullSimplify*. Of course, we need to specify the names of the transformation functions as arguments (options) for the functions *FullSimplify* and *Simplify*. It should be also mentioned that in some cases symbolic computation can lead into the expression containing an error. One of the simplest examples is the integration of the function x^n. *Mathematica* returns result which is exact for all values of n except for $n = -1$.

All the algorithms dealt with in the theory of orthogonal polynomials have rational dependence of the result of the input data. Since, there are very few rational connections between the special functions, applicability of the functions *Simplify* and *FullSimplify* is rather small. Actually no real problem arose with some special function as input data for which some serious simplification can be done using functions *Simplify* and *FullSimplify*.

It should be mentioned that expressions involving special functions, which are not known to *Mathematica*, can not be used in the process of simplification. For the new relations between special functions we need to program new transformation functions. Therefore, possibility of simplification of expressions involving special functions is based on the simplifications already being done. The most important application of the possibility of symbolic computing in the package is based on the possibility of producing analytic results which were too complicated to be obtained by hand. Symbolic computing enables constructions which can be useful in testing some hypothesis, but also it can be used to impose some.

Arbitrary precision number format offers a totally new perspective for the numerical computation. *Mathematica* can represent real number with almost infinite precision, i.e., with almost infinite length mantissa. We mention that arbitrary length mantissa gives possibility of performing calculations in the arithmetics of arbitrary precision. Arithmetics of arbitrary precision is exactly what is needed by anyone performing numerical calculations. [1]. It is known that there exist numerical algorithms with bad conditioning, resulting in the significant loss of precision in the result. We adopt the following meaning of the term bad conditioning: Algorithm is bad conditioned provided we need much higher precision of the input data to

[1] Although, numerical algorithms are designed to return the result of the precision which is comparable with the precision of the input data, possibility of the extended mantissa is valuable at least for the possibility of checking the conditioning of the computation.

get output data with a specified precision. Bad conditioning usually makes the difference between applicable and non-applicable numerical algorithms. If bad conditioned algorithm is executed on the machine with standard 16-digits mantissa it may happen that all digits in the result are wrong, which is clearly an argument to call such an algorithm non-applicable. The same algorithm in the arbitrary precision arithmetics is applicable; provided we know the input data with sufficient precision, we can always get some significant digits in the result.

Of course, arbitrary precision number format can not really operate with numbers with infinite mantissa, but mantissa can safely be a couple of thousands digits long and still execution time can be acceptable.

3. Orthogonal Polynomials

The sequence of polynomials $\{P_k\}_{k=0}^{+\infty}$ is a sequence of orthogonal polynomials with respect to the measure μ if and only if the following conditions are fulfilled

- $\deg P_k = k, \quad k \in \mathbb{N}_0,$
- $\int P_k P_\ell \, d\mu = C_k \delta_{k\ell}, \quad k, \ell \in \mathbb{N}_0, \quad C_k \neq 0, \ k \in \mathbb{N}_0.$

The case when there exist some k such that constant C_k is equal to zero belongs to the cases of degenerate orthogonality and we say that sequence of orthogonal polynomials is not regular, or that measure μ is not regular.

There exist a lot of sequences of orthogonal polynomials studied to these days. A famous one is the sequence of Legendre polynomials, orthogonal with respect to the Legendre measure (see [13]).

$$\int_{-1}^{1} P_k(x) P_\ell(x) \, dx = \frac{2\delta_{k\ell}}{2k+1}. \tag{3.1}$$

The characteristic property of orthogonal polynomials is the three term recurrence relation they satisfy. It can be proven that monic sequences of orthogonal polynomials satisfy the following relation

$$P_{k+1}(x) = (x - \alpha_k) P_k(x) - \beta_k^2 P_{k-1}(x), \tag{3.2}$$

where sequences α_k and β_k^2, $k \in \mathbb{N}_0$, are called coefficients of the three term recurrence relation. The condition $C_k \neq 0$ is equivalent to the condition

$\beta_k^2 \neq 0$, $k \in \mathbb{N}_0$. The case when there is some β_k equal to zero belongs to non-regular sequences of orthogonal polynomials.

Construction of orthogonal polynomials means calculation of the three term recurrence relation coefficients, i.e., the sequences α_k and β_k, $k \in \mathbb{N}_0$. If we know the three term recurrence coefficients we are able, applying three term recurrence relation (3.2), to construct all the members of the sequence of orthogonal polynomials. If we do not know the three term recurrence relation coefficients then it is useful to calculate them since they are also needed for the calculation of zeros of orthogonal polynomials, and for the construction of the related Gaussian quadrature rules.

It is known (see [13]) that zeros of orthogonal polynomials can be determined as eigenvalues of the tridiagonal matrix, known as the Jacobi matrix. The Jacobi matrix is a matrix created from the sequences of the three term recurrence relation coefficients and it has the following form

$$J_n = \begin{bmatrix} \alpha_0 & \beta_1 & 0 & \dots & 0 \\ \beta_1 & \alpha_1 & \beta_2 & \dots & 0 \\ 0 & \beta_2 & \alpha_2 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \alpha_{n-1} \end{bmatrix}. \tag{3.3}$$

As it can be seen, instead of the quantities β_k^2, $k \in \mathbb{N}$, quantities β_k, $k \in \mathbb{N}$, appear in the Jacobi matrix. So if some β_k^2 is negative (or complex) respective entry in the Jacobi matrix is complex, we can expect that eigenvalues are also complex. If all quantities β_k^2, $k \in \mathbb{N}$, are positive, then the Jacobi matrix is real and symmetric. Therefore, it has only real eigenvalues with simple algebraic multiplicity (see [13]). In the case all β_k^2, $k \in \mathbb{N}$, are positive we also say that we are dealing with a positive definite case; when at least one β_k^2, $k \in \mathbb{N}$, is negative (or complex) we say that we are dealing with a quasi-definite case. This difference is significant for the determination of zeros of orthogonal polynomials. For the positive definite case, Jacobi matrix is real and symmetric, for calculation of zeros of orthogonal polynomials it is enough to use QR-algorithm, (see [13]). Determination of zeros is more complicated for the quasi-definite case, since QR-algorithm can become bad conditioned, or orthogonal polynomial may have zeros of higher algebraic multiplicity.

One of the most important applications of orthogonal polynomials is approximation of the integral with respect to the orthogonality measure. Approximation of the integral is performed with the weighted sum of the

following form

$$\int f\,d\mu \approx \sum_{k=1}^{n} w_k f(x_k). \tag{3.4}$$

Quantities w_k, $k = 1, \ldots, n$ are called weights of the quadrature formula, and quantities x_k, $k = 1, \ldots, n$, are called nodes of the quadrature formula. Usual requirement of the quadrature formulae is maximal algebraic degree of exactness. A formula has algebraic degree of exactness n provided it can integrate exactly all polynomials with degree not exceeding n. A quadrature formula for which weights are designed to fulfill this criterion is called the interpolatory quadrature formula. The special case of the interpolatory quadrature formulae are quadrature formulae for which nodes are fixed and chosen equidistant inside of the supporting interval of the measure of orthogonality μ. The other way of construction is to choose nodes and weighs so that we have maximal algebraic degree of exactness. This idea originated in the work of Gauss. Quadrature formulae for which nodes and weights are chosen so that formula has the maximal algebraic degree of exactness are called Gaussian quadrature formulae.

It can be shown that Gaussian quadrature formulae have zeros of orthogonal polynomials as their nodes (see [13], [2]). The formula with n terms has as its nodes zeros of the n-th orthogonal polynomial and has algebraic degree of exactness $2n - 1$. The main benefit introduced by the Gaussian quadrature formula is that we need only n computations of the integrand to get algebraic degree of exactness $2n - 1$, which is only half of what we need with the interpolatory quadrature rule with, for example, fixed and equidistant nodes for the same algebraic degree of exactness. A problem connected with Gaussian quadrature rules is connected with the measures which are not positive, also known as signed or complex measures. Zeros of polynomials orthogonal with respect to such measures need not be inside of the supporting set of the measure, even more zeros of such orthogonal polynomials need not be simple. It is well-known that for the positive measure zeros of orthogonal polynomials are inside the supporting set and that they are simple. In the latter case, the Gaussian quadrature rule has full meaning, since if nodes are not contained in the supporting set of the measure we can not apply the quadrature rule to the integrands for which we do not know the values outside the supporting set of the measure[2].

[2]Unless integrand is not an analytic function, we can not extend it uniquely to some neighborhood of the supporting set of the measure in the complex plane, so that we can not calculate its values in nodes of the quadrature rule which do not belong to the supporting set of the measure.

There exist measures which do not have sequences of orthogonal polynomials at all. Such measure is, for example, $d\mu(x) = \chi_{[-a,a]}xdx$, $a \in \mathbb{R}$. Such measures are called non-definite or non-regular, and sequence of orthogonal polynomials can not be constructed at all. It is possible in some such cases to create a sequence of orthogonal polynomials with respect to the 'slightly' modified measure. We also mention that there can be constructed a measure for which integrals of all polynomials are equal to zero. If support of such measure is bounded integral of every continuous function, continuous on the supporting set is also zero. If support is not bounded integrals of continuous functions need not be zero.

If measure is positive the sequence of orthogonal polynomials always exists, therefore, the Gaussian quadrature rule always exists. It can be shown that provided support of the measure is bounded sequence of the Gaussian quadrature formulae converge for the continuous integrand. In the case support is not bounded convergence question is not that simple. It might happen that two different positive measures, with unbounded support, may have the same sequences of orthogonal polynomials. If the sequence of orthogonal polynomials uniquely determines the orthogonality measure, we say that measure and sequence are determinate, if opposite they are non-determinate. A historically important example of the measure which is not determinate is the Stiletjes family of measures connected with the Stieltjes orthogonal polynomials (see [2]).

For the complex measures term determinate is not important since there exist complex measures which have integral zero on the set of polynomials, hence, no sequence of orthogonal polynomials can not determine the complex measure uniquely. What can be determined by the sequence of orthogonal polynomials is the class of measures having the same orthogonal polynomial sequence. If we are limited to the measures of the bounded support if integrals of polynomials can not be distinguished integrals of continuous functions can not be distinguished either, provided integrand is continuous on the union on the supporting sets. However, if the support is not bounded, the previous is not valid, which is a direct consequence of the fact that the Weierstrass theorem (see [13]) about approximation of continuous function by polynomials is not valid on the unbounded domains.

We can pose the following question, provided we have quasi-definite measure: what can we say about the convergence of the Gaussian quadrature formulae? Specially, for the case of the positive measure with the bounded supporting set it will converge to the value of the integral, for every continuous function. In the case support is not bounded the previous need not

be true. A case when measure is not positive has been studied only partially. There are no general results about the convergence of the Gaussian quadrature formulae. There is the result connected with the polynomials orthogonal on the semicircle (see [9]). Also, some convergence results can be given in the case sequences of the three term recurrence coefficients are uniformly bounded (see [4]). As it has already been mentioned, the Gaussian quadrature formula does not have any meaning for the continuous functions unless measure is positive, since zeros of orthogonal polynomials need not be contained in the supporting set of the measure. In the case of quasi-definite measure a Gaussian quadrature formula has full meaning only if the integrand is an analytic function in the some domain of the complex plain, which is the neighborhood of the supporting set of the orthogonality measure and provided all zeros of orthogonal polynomials are simple. A case with non-simple zeros can also be useful (see [4]). Whatsoever, such a neighborhood of the supporting set of the measure has not been characterized yet, neither it is simple to characterize the case when all zeros of orthogonal polynomials are simple. In the case of already mentioned polynomials orthogonal on the semicircle it was shown that zeros of orthogonal polynomials are localized in the certain domain of the complex plain and that zeros are simple (see [9]). The experimental results are quite interesting and show that in the case of quasi-definite measure we may also have localized and simple zeros very often, actually it is hard to find a case in which it will be known that zeros are not simple.

For the class of polynomials orthogonal on the semicircle we know that all zeros are simple; the same holds for the class of the generalized Bessel polynomials. If all zeros are not simple, the Gaussian quadrature formula is modified (see [4]), and has the following form

$$\int f(x)\,d\mu \approx \sum_{k=1}^{M}\sum_{\nu=1}^{M_k} w_{k,\nu} f^{(\nu-1)}(x_k), \tag{3.5}$$

where x_k, $k = 1, \ldots, M$, are different zeros of the n-th orthogonal polynomial with respect to μ and M_k, $k = 1, \ldots, M$, are respective multiplicities, of course, $\sum_{k=1}^{n} M_k = n$.

Regardless of the problems we mentioned, the leading idea during the construction of the package was to include all known classes of orthogonal polynomials and to prove tools for the construction of the Gaussian quadrature rule for the all possible polynomial classes. Construction of the Gaussian quadrature rule is important at least, as it provides the way to calculate zeros of the orthogonal polynomials.

Besides the basic Gaussian quadrature formula there are other types of the quadrature rules which can be used for the approximation of an integral and which are also connected with some classes of orthogonal polynomials. Usually, numerical integration is performed by the Gauss-Kronrod quadrature formula. It can be given in the following form

$$\int f\,d\mu \approx \sum_{k=1}^{n} w_k f(x_k^G) + \sum_{k=1}^{n+1} w_{n+k} f(x_k^K) \tag{3.6}$$

The idea for the Gauss-Kronrod quadrature formula belongs to Kronrod, and it can be summarized as follows. If we already have calculated function values in the Gaussian quadrature formula for some $n \in \mathbb{N}$, can we construct the formula with algebraic exactness higher then $2n-1$ which will include already calculated function values. In formula (3.6), quantities x_k^G, $k = 1, \ldots, n$, are nodes of the Gaussian quadrature formula of exactness $2n-1$ quantities w_k, $k = 1, \ldots, 2n+1$, are the weights of Gauss-Kronrod quadrature formula which are to be constructed, as we need to construct additional nodes x_k^K, $k = 1, \ldots, n+1$. The algebraic degree of exactness of this formula, according to the number of free terms it has, is $3n+1$. More information on the Gauss-Kronrod quadrature rules can be found in [11], [8], [1].

A problem with the Gauss-Kronrod quadrature formula is position of the additional nodes x_k^K, $k = 1, \ldots, n+1$, with respect to the supporting set of the measure μ. Namely, it is proven that in the case of the Legendre measure additional nodes x_k^K, $k = 1, \ldots, n+1$, are contained in the supporting set of the Legendre measure, even more additional nodes are interlaced with Gaussian nodes, i.e., $x_k^K < x_k^G < x_{k+1}^K$, $k = 1, \ldots, n$. This is particularly of great importance for the applications, since, almost every function written to perform numerical integration relies on the previous fact. On the other hand it is proven that in the case of the Laguerre measure additional nodes x_k^K, $k = 1, \ldots, n+1$ in the Gauss-Kronrod quadrature formula are outside of the supporting set of the measure.

There exist also quadrature formulae which use derivatives of the integrand to approximate an integral. Those are formulae of the following form

$$\int f\,d\mu \approx \sum_{k=1}^{n} \sum_{j=0}^{2s_k-1} w_{kj} f^{(j)}(x_k) \tag{3.7}$$

These formulas have been studied only for the positive measures, and in that case it can be shown that nodes of these formulae are contained inside the supporting set of the measure. In this case construction of the quadrature

formulae can be performed using orthogonal polynomials, but those polynomials are not polynomials orthogonal with respect to the measure μ any more, rather, those polynomials are orthogonal with respect to the measure

$$d\mu^{\sigma}(x) = d\mu(x) \prod_{k=1}^{n} (x - x_k)^{2s_k}. \tag{3.8}$$

Where x_k, $k = 1, \ldots, n$, are zeros of the polynomial of the n-th degree with respect to the measure μ^{σ}. Usually, with σ we denote the vector $(s_1, s_2, s_3, \ldots, s_n)$. It is possible to give the construction when all s are different or when all s are the same.

4. Implementation of Some Symbolic Algorithms

As it is already mentioned in the previous section 2, all algorithms which are important in the theory of orthogonal polynomials have nonlinear dependence of output data as functions of input data, but that nonlinearity can be expressed using rational functions. The most important algorithms are Chebyshev algorithm, modified Chebyshev algorithm, Laurie algorithm and algorithms which perform Christoffel modifications of the measure.

Chebyshev algorithm can be represented as the mapping of the sequence of moments of the measure μ

$$\mu_k = \int_R x^k \, d\mu(x),$$

into the coefficients of the three term recurrence relation α_k and β_k^2, $k \in \mathbb{N}_0$. Algorithm is rational and nonlinear and it can be represented using recurrence relation which uses only addition and multiplication of the operations (see [7], [13], [3]).

Modified Chebyshev algorithm can be expressed as the mapping of the sequence of modified moments of some measure μ

$$\mu_k = \int_R T_k(x) \, d\mu(x),$$

and coefficients of the three term recurrence relation for the sequence of the polynomials T_k, $\hat{\alpha}_k$ i $\hat{\beta}_k^2$, $k \in \mathbb{N}_0$, into the coefficients of the three term recurrence relation α_k i β_k^2, $k \in \mathbb{N}_0$, for the measure μ. Algorithm is, also, nonlinear and has rational data dependence, only operations involved are addition and multiplication (see [7], [13], [3]).

Laurie's algorithm can be expressed as the mapping between the three term recurrence relation coefficients of the measure μ into the three term recurrence relation coefficients from which it is possible to get the Gauss-Kronrod quadrature formula nodes and weights using QR-algorithm. Algorithm is also nonlinear and rational, the only operations involved are addition and multiplication (see [3], [11]).

The Christoffel modification algorithms are algorithms which give answer to the following problems. Suppose we are given three term recurrence relation coefficients α_k and β_k^2, $k \in \mathbb{N}_0$, for the measure μ, what are three term recurrence coefficients for the measures

$$\frac{d\mu(x)}{z-x}, \quad (z-x)d\mu(x). \tag{4.1}$$

Algorithms which describe the solutions of the problems are known as the Christoffel modification algorithms (see [7]). The Christoffel modification algorithms are nonlinear and rational, the only operations involved are addition and multiplication (see [7], [3], [6]).

Because of the rational dependence of output and input data the most optimal implementation should use built-in power of the functions which operate on the rational expressions. Implementation should involve the following calculations

- On every operation of addition, we should apply the function *Together*, which performs addition of two (or more) rational expressions into the unique rational expression, also function *Together* performs possible cancellation to the rational expression obtained, such that returned rational expression has mutually simple numerator and denominator. Additional information on the function *Together* can be found in [19], [3].

- On every operation of the multiplication of the two rational expressions function *Cancel* should be applied, since *Mathematica* does not cancel common factors using simple multiplication, although it writes the result as the single rational expression. Only after function *Cancel* is employed numerator and denominator are mutually simple. For additional information about the function *Cancel* refer to [19], [3].

- On every expression which represents final result of the calculation function *Factor* should be applied in order to get the factored result, which is much more readable then the non-factored one. We repeat

that function *Factor* factors over ring of integers, however, it can be set to factor over any ring of Gaussian integers. For additional information on function *Factor* one should see [19], [3].

It is possible also to perform factorization and cancellation over some extensions of the ring of integers, ring known as Gaussian integers. More about that can be found in [19], [3]. We only mention that such extended functionality is implemented in the package.

As an example of symbolic calculation we present construction and verification of the Layman hypothesis (see [12]). The Layman hypothesis can be expressed in the following way sequence of the Hankel transforms of the sequence

$$C_n + C_{n+1} = \frac{(2n)!(5n+4)}{n!(n+1)!}, \quad n \in \mathbb{N}_0,$$

i.e., that Hankel transform of the sum of two consecutive Catalan numbers are the Fibonacci numbers with odd indices. This hypothesis can be reformulated in the following form (see [5]) β coefficients of the three term recurrence relation satisfied by the orthogonal polynomials with the moment sequence

$$C_n + C_{n+1}, \quad n \in \mathbb{N},$$

are given by

$$\beta_k^2 = \frac{F_{2k-1}F_{2k+3}}{F_{2k+1}^2}, \quad k \in \mathbb{N},$$

where F_k, $k \in \mathbb{N}_0$, are Fibonacci numbers. This fact is proven in [5].

It can be verified directly using functions implemented in the package, using for example the following code

```
In[1]:=<<OrthogonalPolynomials‘
In[2]:=mom=Table[Binomial[2n,n]/(n+1)+
   Binomial[2(n+1),n+1]/(n+2),{n,0,20}];
   aChebyshevAlgorithm[mom,Algorithm->Symbolic]
Out[3]={{3/2, 19/10, 129/65, 883/442, 6051/3026,
   41473/20737, 284259/142130, 1948339/974170, 13354113/6677057,
   91530451/45765226}, {2, 5/4, 26/25, 170/169, 1157/1156,
   7922/7921, 54290/54289, 372101/372100, 2550410/2550409,
   17480762/17480761}}
```

Even more hypothesis can be found in [5].

5. Numerical Algorithms

For the implementation of the numerical algorithms the most important thing is to provide for the user the simplest control of the precision and accuracy of the computation being performed. *Mathematica* has two quantities to control quality of the returned numerical quantities. Those are *Precision* i *Accuracy*. It is known that quantity of the exact digits in some number can be measured as its relative error with respect to the exact number (see [19], [13], [3]). This characteristic of the number is known as the *Precision* of the number. Precision is perfect for measuring quality of the approximation of all numbers except the number zero, since if we are approximating zero precision does not have any meaning (see [3]). For determining the quality of the approximation of number zero we need different criterion, and that is *Accuracy*. Accuracy is really the absolute error of the approximation. The number has accuracy n if n decimal digits behind decimal point are zeros. These two terms *Precision* i *Accuracy* determine quality of the returned numerical quantities. Every numerical function implemented has as options a possibility of specifying these two quantities.

Beside these two terms which are crucial for determining the request quality of the result, it is also crucial to determine the number of decimal digits we are using to work with, i.e., we need to specify the length of the mantissa in the intermediate results. Every implemented numerical function has as an option also *WorkingPrecision*, which determines the length of mantissa in the intermediate results. Of course, we can not get higher precision then the precision we are working with.

The biggest effort in the implementation of the numerical algorithms is made for the construction of quadrature formulas. For the construction of the Gaussian quadrature formula for the positive measure QR-algorithm is implemented, for the computation of the eigenvalues of the real symmetric Jacobi matrices, this algorithm also is implemented for the calculation of the zeros of orthogonal polynomials (see, [10], [7], [13]). For the computation of weights in the Gaussian quadrature rule, QR-algorithm can be used; as it is well-known, however, there are the cases when QR-algorithm exhibits bad conditioning, such that an alternative approach is used for the construction of weights in the Gaussian quadrature formulae (see [15]).

For the signed (or complex) measures the nodes of the Gaussian quadrature formulae can be calculated using QR-algorithm, but one should be cautious since it is well-known that for the complex Jacobi matrices QR-algorithm is bad conditioned even for the computation of the eigenvalues of

the matrix. Therefore, one should increase *WorkingPrecision* if eigenvalues of the complex Jacobi matrix are constructed. There are also some alternative approaches. For example, for the computation of zeros of generalized Bessel polynomials there is an algorithm developed in [18], which can be applied in the modified form in all semiclassical cases. For the detailed discussion see [3]. In any case, in almost all practical applications it is enough to use QR-algorithm with *WorkingPrecision* which is significantly higher then the requested precision of the result. For example, it is usual to use couple of hounder digits in mantissa to calculate, for example, zeros of the polynomial of degree two or three hounders.

For the construction of weights in the Gaussian quadrature formula it is usual to use QR-algorithm as it is proposed by [10], [7], but precision of the constructed weights is not controlled. Because of this fact, there exist some constructions where returned weights are quite wrong, with very high loss of precision. Therefore, as it was already mentioned in [15], we developed an approach which can control the precision of the returned weights. However, this method slightly increases the time of computation and also it requires slightly more memory, but these problems are nothing compered to the fact that we have exact information of the precision of constructed weights.

For the construction of the Gauss-Kronrod quadrature formulae two algorithms are implemented. One developed in [1] and the second developed in [11]. The first algorithm can be used strictly for the construction of the Gauss-Kronrod quadrature formulae for the positive measure for which additional nodes in (3.6) are in addition contained in the support of the measure. The second algorithm is much more flexible and it can be used for the construction of the Gauss-Kronrod quadrature formulae for the arbitrary measure. It should be noted that second algorithm performs construction of the Gauss-Kronrod quadrature formula in such a way that it really constructs Jacobi matrix whose eigenvalues are nodes in the Gauss-Kronrod quadrature formula, this means that in order to construct the Gauss-Kronrod quadrature formula we compute again already known Gaussian nodes in (3.6). The first algorithm does not have such a redundancy, but it can only be applied for the special measures.

For the construction of nodes in quadrature formulae of type (3.7) two algorithms are implemented. One is given in [14], [16]. The mentioned algorithm uses construction over increasing parameter s in the quadrature formula (3.7), and it can hardly be applied in the case for the measures with unbounded supporting set. The second algorithm is investigated in [17] and it can be applied with equal success for the measures with bounded and

unbounded support. The latter algorithm is also given in many details in [3].

Still another group of algorithms is implemented and those are algorithms for performing Christoffel modifications of the measure (see [6]). In this class we should mention a very well conditioned algorithm which can be used for the modification of the measure with the factor $(x-t)^2$. Almost all algorithms for the modification of the measure can be found in [7].

We are also going to mention Chebyshev algorithm, which can be used for the construction of the three term recurrence relation coefficients for the sequence of moments. As it is well-known this algorithm is bad conditioned. For some measures, for example Laguerre measure, we need moments calculated with much higher precision in order to obtain the requested precision in three term recurrence coefficients. Since *Mathematica* can easily handle few hundreds of precision in the input data and in intermediate calculation, Chebyshev algorithm is applicable. We are going to give the example of determination of the three term recurrence coefficients for the measure

$$d\mu(x) = \chi_{[0,+\infty)}(x) e^{-x^4} dx. \tag{5.1}$$

It is known that moments can be expressed in the form

$$\mu_k = \int_{-\infty}^{\infty} e^{-x^4} x^k \, dx = \frac{1+(-1)^k}{4} \Gamma\left(\frac{k+1}{4}\right). \tag{5.2}$$

Coefficients of three term recurrence relation can be calculated using the following program

```
In[2]:=<<OrthogonalPolynomials‘
In[4]:=SetPrecision[aChebyshevAlgorithm[Table[(1 + (-1)^k)/4
    Gamma[(k + 1)/4],{k,0, 40}],WorkingPrecision->100],16]
Out[4]={{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {1.812804954110954, 0.3379891200336424, 0.4016796597635174,
    0.5051042323448223, 0.5780581503317113, 0.6467673820472450,
    0.7078631509051525, 0.7644231260520773, 0.8170217520109819,
    0.8664703641900223, 0.913249899440007, 0.957756084884179,
    1.000288746559780, 1.041089178928227, 1.080352725238558,
    1.118240764497883, 1.154888263975016, 1.190409501421220,
    1.224902232945476, 1.258450855794495}}
```

6. Supported Classes of Orthogonal Polynomials and Implemented Functions

All classes of orthogonal polynomials are divided into three classes. Those are polynomials orthogonal with respect to the absolutely continuous measure,i.e., where measure can be represented using weight function $d\mu = w(x)dx$, completely singular measure and polynomials orthogonal with respect to the signed (complex) measure. The first class of the polynomials is termed continuous class, second class is termed discrete class and the third class is termed complex.

All supported classes of polynomials are the following:

- Legendre, Chebyshev of the first, second, third and fourth kind, Gegenbaur, generalized Gegenbauer, Jacobi, associated Legendre, Polaczek the first, second and third kind, generalized Laguerre, generalized Hermite, Abel, Lindelof, logistic, Hano, dual Han, Wilson, Stieltjes-Wigert, Meixner of the second kind,
- Charlier, Lommel, Meixner of the first kind, Tricomi-Carlitz
- Bessel, generalized Bessel, Gautschi-Milovanović

There are also other classes of orthogonal polynomials which are to be implemented. We are referencing the class of orthogonal polynomials by the term which is formed from the name of the polynomials class with prefix "a". For example, Chebyshev polynomials of the first kind are referenced with "aChebyshevI".

For all supported classes of the polynomials package provides basic information. Function operating on the classes are the following

- *aThreeTermRecurrence*-function returns three term recurrence coefficients of the referenced polynomial class. It is implemented in the format of the *pure function*. It can return coefficients of the three term recurrence relation in the closed analytic form.
- *aNorm*-function which returns the norm of the monic polynomials of the referenced class. It is implemented in the format of the*pure function*. It is able to to return closed analytic expression of the norm of the referenced polynomial class.
- *aNumerator*-returns numerator polynomials of the given order for the referenced polynomial class (see [2]). It is also implemented in the format of the *pure function*.

- *aKernel*-returns kernel polynomial of the referenced polynomial class (see [2]). It is implemented in the format of the *pure function*.

Function which are specific for the continuous class of polynomials are the following

- *aWeight*-returns the weight function with respect to which referenced class is orthogonal, it is also *pure function*.
- *aGetInterval*-returns the interval of the orthogonality, i.e., the support of the measure.

Functions specific for the discrete polynomial class are the following

- *aDistribution*-represent the distribution function with respect to which referenced polynomials are orthogonal to, where distribution function is given by
$$\psi(x) = \int_{-\infty}^{x} d\mu.$$
- *aSupport*-returns the supporting set of the measure, however it is clear that support can be an infinite set, that is why function produces a message about the supporting set and also returns few points of the supporting set. The number of returned points is given as the parameter of the function.

There are no functions specific only for the complex class of orthogonal polynomials according to the supporting set. Complex class uses already presented functions. If complex polynomials are orthogonal with respect to the absolutely continuous measure, the weight function is returned using *aWeight*. If polynomials are orthogonal on the interval of the real line, the interval is returned by the function *aGetInterval*. If polynomials are orthogonal over some curve in the complex plain, information about the curve is contained in the function *aSupport*.

Every polynomial class has, as its representative, function which may return value of the specific polynomial in the sequence. The name of the representative function is given by adding prefix "a" to the English transcription of the name of the polynomial class. The function is able to return the numerical value of the given member of the polynomial sequence at some specific point, or it can return the analytic value of the polynomial.

For the construction of the quadrature formulae interface is unified through the function *aNodesWeights*. Choice of the different quadrature formulae is

dictated using the keyword for the given quadrature formula and referencing to some polynomial class. Keywords for the quadrature formulae are the following

- *aGaussian*-construction of the Gaussian quadrature formulae (3.4). It is possible to perform construction for all supported classes of the orthogonal polynomials. In the case function *aNodesWeights* is called for the construction of this type of quadrature rule function *aGaussianNodesWeights* is called which performs computation. Function *aGaussianNodesWeights* has different calling formats. It is possible to call function *aGaussianNodesWeights* for the class of polynomials which is not supported and for which coefficients of the three term recurrence relation are known, if the construction with QR-algorithm is wanted, or for which we know good starting values for the Pasquini algorithm (see [18]). Once functions *aNodesWeights* or *aGaussianNodesWeights* are called it is possible to choose the way in which weights are constructed.

- *aRadau*-performs construction of the Gauss-Radau quadrature formula. It is possible to construct Gauss-Radau quadrature formula calling directly function *aRadauNodesWeights* or calling the function *aNodesWeights*. It is possible to construct Gauss-Radau quadrature formula for all supported polynomial classes for which this formula has meaning.

- *aLobatto*-performs the construction of the Gauss-Lobatto quadrature formulae. It is possible to perform the construction directly calling the function *aLobattoNodesWei-ghts* or calling the function *aNodesWeights*. It is possible to construct Gauss-Lobatto quadrature formula for all supported classes of the polynomials for which this formula has meaning

- *aKronrod*- performs the construction of Gauss-Kronrod quadrature formula (3.6). It is possible to construct Gauss-Kronrod quadrature formula directly calling the function *aKronrodNodesWeights* or calling the function *aNodesWeights* with the keyword *aKronrod*. Construction can be performed for all supported polynomial classes for which Gauss-Kronrod quadrature formula exists, i.e., for which additional nodes of the Gauss-Kronrod formula are inside the supporting set of the measure. If the additional nodes of The Gauss-Kronrod quadrature formula are not inside the supporting set construction can be performed using Laurie algorithm.

- *aTuran*-performs the construction of the Gauss-Turan quadrature formula (3.7), with all $s_\nu = s$, $\nu = 1, \ldots, n$. It is possible to construct Gauss-Turan quadrature formula calling the function *aTuranNodesWeights* or function *aGaussianNodesWeights* using the keyword *aTuran*. Construction is possible for all supported classes of the polynomials with positive orthogonality measure.

- *aSigma*-performs the construction of the Gaussian quadrature formula (3.7). It is possible to perform the construction using function *aSigmaNodes-Weights* or the function *aNodesWeights* using the keyword *aSigma*. Construction is possible for all supported classes of polynomials with the positive orthogonality measure.

Using function *aThreeTermRecurrence* it is possible to call Chebyshev and modified Chebyshev algorithm for the given set of moments, i.e., modified moments. There is also implemented function *aChebyshevAlgorithm* which performs construction of the three term recurrence relation using Chebyshev and modified Chebyshev algorithm.

We also mention possibilities of performing Christofell modifications (4.1) for the given class of orthogonal polynomials. Using function *aChristoffelAlgorithm* and choosing among options it is possible to construct whatever modification with the rational function. Function *aModify* unifies interface i gives the possibility of performing modifications directly on the implemented polynomial classes.

REFERENCES

1. D. Calvetti, G.H. Golub, W.B. Gragg and L. Reichel: *Computation of Gauss-Kronrod quadrature rules.* Math. Comp. **69** (2000), 1035–1052.
2. T. S. Chihara: *An Introduction to Orthogonal Polynomials.* Gordon and Breach, New York, 1978.
3. A. S. Cvetković: *Program Package for Symbolic and Numerical Construction of Orthogonal Polynomials and Quadrature Formulas.* Master Thesis, University of Niš, Niš, 2002 (Serbian).
4. A. S. Cvetković: *Nonstandard Orthogonality and Quadrature Formulae.* Ph.D. Theses, University of Niš, Niš, 2004 (Serbian).
5. A. Cvetklović, P. Rajković, M. Ivković: *Catalan numbers, and Hankel transform, and Fibonacci numbers..* J. Integer Seq. **5** (2002), no. 1, Article 02.1.3, 8 pp. (electronic).

6. W. Gautschi: *An algorithmic implementation of the generalized Christoffel theorem.* In: Numerical Integration (G. Hämmerlin, ed.) ISNM, vol. 57, Birkhäuser, Basel, 1982, pp. 89–106.
7. W. Gautschi: *Algorithm 726: ORTHPOL – A package of routines for generating orthogonal polynomials and Gauss-type quadrature rules.* ACM Trans. Math. Software **10** (1994), 21–62.
8. W. Gautschi: *Orthogonal polynomials and quadrature.* Electron. Trans. Numer. Anal. **9** (1999), 65–76.
9. W. Gautschi and G. V. Milovanović: *Polynomials orthogonal on the semicircle.* J. Approx. Theory **46** (1986), 230–250.
10. G.H. Golub and J.H. Welsch: *Calculation of Gauss quadrature rule.* Math. Comput. **23** (1986), 221–230.
11. D. P. Laurie: *Calculation of Gauss-Kronrod quadrature rules.* Math. Comp. **66** (1997), 1133–1145.
12. J. W. Layman: *The Hankel transform and some of its properties.* J. Integer Seq. **4** (2001), no. 1, Article 01.1.5, 11 pp. (electronic).
13. G. V. Milovanović: *Numerical Analysis, Part I.* Third Edition, Naučna Knjiga, Belgrade, 1991 (Serbian).
14. G. V. Milovanović: *Quadratures with multiple nodes, power orthogonality, and moment-preserving spline approximation.* J. Comput. Appl. Math. **127** (2001), 267–286.
15. G. V. Milovanović and A. S. Cvetković: *Note on a construction of weights in Gauss-type quadrature rule.* Facta Univ. Ser. Math. Inform. **15** (2000), 69–83.
16. G. V. Milovanović and M. M. Spalević: *Quadrature formulae connected to σ-orthogonal polynomials.* J. Comput. Appl. Math. **140** (2002), 619–637.
17. G. V. Milovanović, M. M. Spalević, and A. Cvetković: *Calculation of Gaussian type quadratures with multiple nodes.* Math. Comput. Modelling **39** (2004), 325-347.
18. L. Pasquini: *Accurate computation of the zeroes of the generalized Bessel polynomials.* Numer. Math. **86** (2000), 507–538.
19. S. Wolfram: *The Mathematica Book.* Third edition, Wolfram Media, Inc., Champaign, IL; Cambridge University Press, Cambridge, 1996.

Faculty of Electronic Engineering
Department of Mathematics
P.O. Box 73
18000 Niš, Serbia
e-mails: aca@elfak.ni.ac.yu
grade@ni.ac.yu