E5-2004-69

H. Fukuda[1], M. Katuya[1], E. O. Alt[2], A. V. Matveenko

NONCLASSICAL ORTHOGONAL POLYNOMIALS AND CORRESPONDING QUADRATURES

Submitted to «Computer Physics Communications»

[1]School of Administration and Informatics, University of Shizuoka, Shizuoka 422-8526, Japan
[2]Institut für Physik, Universität Mainz, D-55099 Mainz, Germany

Фукуда Х. и др. E5-2004-69
Неклассические полиномы и соответствующие им квадратуры

Представлена программа для вычисления абсцисс и весов квадратуры Гаусса для произвольного веса и интервала. Программа написана на языке *Mathematica* и работает в том случае, если интегралы моментов вычисляются в аналитическом виде. Результатом работы программы является ФОРТРАН-подпрограмма, готовая для вычисления определенного интеграла.

Работа выполнена в Лаборатории теоретической физики им. Н. Н. Боголюбова ОИЯИ.

Препринт Объединенного института ядерных исследований. Дубна, 2004

Fukuda H. et al. E5-2004-69
Nonclassical Orthogonal Polynomials
and Corresponding Quadratures

We construct nonclassical orthogonal polynomials and calculate abscissas and weights of Gaussian quadrature for arbitrary weight and interval. The program is written by *Mathematica* and it works if moment integrals are given analytically. The result is a FORTRAN subroutine ready to utilize the quadrature.

The investigation has been performed at the Bogoliubov Laboratory of Theoretical Physics, JINR.

Preprint of the Joint Institute for Nuclear Research. Dubna, 2004

INTRODUCTION

The integrals of elementary functions could not, in general, be computed analytically, while derivatives could be. A lot of numerical analysis of the quadrature has been worked out [1]. Generally, the integral of a function is approximated by the sum of its functional values at a set of points, multiplied by certain weighting coefficients. The Gaussian quadrature gives the freedom to choose both the weights and the location of the abscissas at which the function is to be evaluated. The number of the function evaluation can be reduced twice.

The classical Gaussian quadratures are related to the corresponding classical polynomials which all are either hypergeometric or confluent hypergeometric series, i.e., trivial cases; if we are limited to use them only, the power of the method, which allows one to arrange the choice of weights and abscissas to make the integral exact for «polynomials times some known function $W(x)$», is wasted.

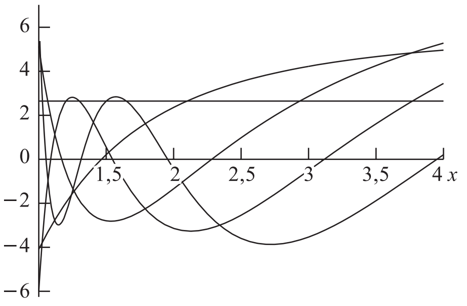

Fig. 1. Up to $n = 4$ normalized orhogonal polynomials $p_n(\mathbf{z} = z(x))$ with $W(x) = 1/(1+x^2)^2$, $z(x) = x/\sqrt{1+x^2}$, and interval $[a, b] = [1, \infty]$

Here we produced the code for generating Gauss quadrature with arbitrary weight and interval. It can also be used for classical cases if needed. We have found the code to be dramatically effective used in our 2D quadrature [2].

To finish the Introduction we present the first four polynomials useful (closely related) to our application.

The corresponding $n = 1, 2, 3, 4$ monic polynomials are:

$$\mathbf{z} - \frac{2}{3}\,\frac{\sqrt{2}}{\pi - 2};$$

$$\mathbf{z}^2 - \frac{6}{5}\,\frac{\sqrt{2}\,(9\,\pi - 28)\,\mathbf{z}}{9\,\pi^2 - 18\,\pi - 32} - \frac{1}{20}\,\frac{45\,\pi^2 - 448}{9\,\pi^2 - 18\,\pi - 32};$$

$$\mathbf{z}^3 - \frac{24}{7}\,\frac{\sqrt{2}\left(225\,\pi^2 - 1290\,\pi + 1832\right)\mathbf{z}^2}{675\,\pi^3 - 900\,\pi^2 - 13776\,\pi + 31232} -$$
$$-\frac{1}{14}\,\frac{\left(-137472\,\pi + 4725\,\pi^3 + 285376\right)\mathbf{z}}{675\,\pi^3 - 900\,\pi^2 - 13776\,\pi + 31232} +$$
$$+\frac{16}{35}\,\frac{\sqrt{2}\left(1078 + 225\,\pi^2 - 1050\,\pi\right)}{675\,\pi^3 - 900\,\pi^2 - 13776\,\pi + 31232};$$

$$\mathbf{z}^4 - \frac{40}{3}\,\frac{\sqrt{2}\left(-948150\,\pi^2 + 1061880\,\pi + 165375\,\pi^3 + 894208\right)\mathbf{z}^3}{2480625\,\pi^4 - 3307500\,\pi^3 - 99831600\,\pi^2 + 417542400\,\pi - 465531904} -$$
$$-\frac{1}{4}\,\frac{\left(1386729600\pi - 4961250\pi^3 + 7441875\pi^4 - 344224800\pi^2 - 1530253312\right)\mathbf{z}^2}{2480625\pi^4 - 3307500\pi^3 - 99831600\pi^2 + 417542400\pi - 465531904} +$$
$$+\frac{5}{42}\,\frac{\sqrt{2}\left(197011456 - 58557120\,\pi - 22623300\,\pi^2 + 6780375\,\pi^3\right)\mathbf{z}}{2480625\,\pi^4 - 3307500\,\pi^3 - 99831600\,\pi^2 + 417542400\,\pi - 465531904} +$$
$$+\frac{1}{336}\,\frac{52093125\,\pi^4 - 9650745344 - 2520705600\,\pi^2 + 9375744000\,\pi}{2480625\,\pi^4 - 3307500\,\pi^3 - 99831600\,\pi^2 + 417542400\,\pi - 465531904}.$$

1. METHOD

We consider a general one-dimensional integral with a given weight $W(x)$

$$I = \int_a^b f(z(x))W(x)dx, \tag{1}$$

where infinity is allowed for both a and b. The variable $z(x)$ is a monotonic function of x and is introduced for convenience. It is well known that the Gaussian quadrature (GQ, hereafter) is a quite effective numerical method if the integrand $f(z(x))$ is smooth. The integral I is approximated by GQ of degree n as

$$I \approx \sum_{j=1}^{n} w_j f(z_j). \tag{2}$$

Abscissas z_j and weights w_j are determined by demanding that Eq. (2) is exact if $f(z)$ is a polynomial of degree $2n-1$ or less.

As is well known [1], the abscissas z_j are $j = 1, 2, 3, \ldots, n$ distinct zeros of the polynomial $p_n(z)$ of degree n defined by the recurrence relation

$$p_{j+1}(z) = (z - a_j)p_j(z) - b_j p_{j-1}(z) \tag{3}$$

with $p_{-1}(z) \equiv 0,\ p_0(z) \equiv 1$, where

$$a_j = \frac{\langle zp_j|p_j\rangle}{\langle p_j|p_j\rangle},\ b_j = \frac{\langle p_j|p_j\rangle}{\langle p_{j-1}|p_{j-1}\rangle}, \tag{4}$$

and

$$\langle f|g\rangle \equiv \int_a^b f(z)g(z)W(x)dx. \tag{5}$$

The polynomials defined as above are *monic*, i.e., the coefficient of their leading term [z^j for $p_j(z)$] is unity. Note that the coefficients in the recurrence relation depend on the adopted normalization.

In its turn, the weights $w_j,\ j = 1, 2, 3, \dots, n$ are given by the expression

$$w_j = \frac{\langle p_{n-1}|p_{n-1}\rangle}{p_{n-1}(u_j)p'_n(u_j)}, \tag{6}$$

where $p'_n(z)$ is the derivative of $p_n(z)$ with respect to z.

If we know the first $2n$ moments of a weight function $W(x)$,

$$\mu_k = \int_a^b z^k W(x)dx,\ k = 0, 1, \dots, 2n-1, \tag{7}$$

it is formally possible to calculate $p_n(z)$. However, it is well known that the solution of the set of algebraic equations for the coefficients a_j and b_j in terms of the moment μ_k is extremely ill-conditioned: «Even in double precision it is not unusual to lose all accuracy by the time $n = 12$» [1]. The effective and stable methods to calculate GQ abscissas z_k and weights μ_k are known only for classical polynomials [1].

On the other hand, if we use symbolic calculations, there is formally no problem with accuracy since we can obtain accurate results by converting the analytic output into numbers. However, in symbolic calculations there is also a limitation originated in the capacity of a computer memory. For example, we have found it to be difficult to obtain the abscissas in the case of $n = 12$ for

$$W(x) = \frac{1}{(1+cx^2)^2},\ z(x) = \frac{x}{\sqrt{1+cx^2}},\ [a, b] = [1, \infty], \tag{8}$$

with $c = 1$ by a direct unsophisticated run.

In order to get abscissas and weights for an arbitrary weight function and large enough n we suggest to combine numerical and symbolic calculations as follows:

1. We calculate the moments (7) symbolically and convert the results into numbers having N_m digits since if we use the moment expressed in symbols the polynomial $p_n(z)$ might be too complex to be used by the usual computer.

2. We obtain the polynomial $p_n(z)$ from the moment accurate in N_m digits and, accordingly, a_i and b_i from Eq. (4).

3. Since accurate abscissas are necessary to calculate weights w_j by Eq. (6), instead of seeking zeros of $p_n(z)$, we follow the suggestion from [1] and use the so-called Jacobi matrix,

$$J = \begin{pmatrix} a_0 & \sqrt{b_1} & & & \\ \sqrt{b_1} & a_1 & \sqrt{b_2} & & \\ & & \cdots & & \\ & & \sqrt{b_{n-2}} & a_{n-2} & \sqrt{b_{n-1}} \\ & & & \sqrt{b_{n-1}} & a_{n-1} \end{pmatrix}. \tag{9}$$

In this case, the eigenvalues of J are the abscissas z_j and the weights w_j are given by the corresponding eigenvector $\mathbf{v}_j$ as

$$w_j = \frac{\mu_0(\mathbf{v}_j \cdot \mathbf{e}_1)^2}{\mathbf{v}_j \cdot \mathbf{v}_j}, \tag{10}$$

where $\mathbf{e}_1 = (1, 0, 0, \ldots, 0)$. We solve this eigenvalue problem numerically in N_z digits precision.

This procedure is quite elementary and can be programmed easily using any symbolic computation system, such as *Mathematica* or *Maple*. But, if we want to have the final accuracy N_a, the internal accuracies N_m for the moments μ_j strongly depend on the problem: the weight function $W(x)$, the argument $z(x)$, the interval $[a, b]$ and the degree n of GQ formula. That is why we check the final accuracy N_a by comparing the exact moments μ_k with those calculated by using the obtained GQ

$$\hat{\mu}_k = \sum_{j=1}^{n} \hat{w}_j \hat{z}_j^k, \tag{11}$$

for $k = 0, 1, \ldots, 2n-1$. Here $\hat{z}_j$ and $\hat{w}_j$ are the calculated abscissas and weights, respectively. Naturally, if they coincide in N_a digits, we are sure that we have the same accuracy for approximation (2). The required accuracy N_z for the eigenvalue problem of the Jacobi matrix also depends on the problem.

2. DESCRIPTION OF THE PROGRAM

The program was written by using *Mathematica 4.0* and should work for its later versions. It produces the abscissas and weights for the weight function $W(x)$, the argument $z(x)$ and the interval $[a, b]$. It works if *Mathematica* can calculate the moments (7) analytically. The result is the FORTRAN subroutine that contains the arrays of calculated abscissas and weights (see the end of this section).

Integral (1) is defined at the beginning of the *Mathematica* code as follows.

The integral region $[a, b]$ where a and b are allowed to be infinity. For example,

```
(* integral region a, b *)
a=1;
b=Infinity;
```

The weight function $W(x)$. For example,

```
(* weight *)
c = 1;
W[x_]:=1/(1+ c x^2)^2;
```

The argument $z(x)$. For example,

```
(* argument *)
z=x/(Sqrt[1+ c x^2]);
```

The inverse of the argument $z^{-1}(u)$, which is defined by $z(z^{-1}(u)) = u$. For example,

```
(* inverse relation *)
zi=u/(Sqrt[1- c u^2]);
```

The degree of the GQ, n. For example,

```
(* degree of formula *)
n=4;
```

The internal precision in number of digits is: N_m for the moment μ_k and N_z for the eigenvalue problem of the Jacobi matrix. For example,

```
(* Internal precision *)
(* in number of digits *)
Nm = 50; (* moments *)
Nz = 25; (* Jacobi matrix *)
```

The final precision N_a of calculated abscissas and weights in number of digits. For example,

```
(* final precision *)
Na = 17;
```

FORTRAN output has two options. If `outputww` is set to 0, the FORTRAN subroutine produced by *Mathematica* provides weights itself, otherwise the FORTRAN subroutine provides weights divided by the weight function $W(x)$. This option is useful for very small or large weights. It can be important, for example, in the case of Gauss–Laguerre quadrature:

$$W(x) = e^{-x},\ z(x) = x,\ [a,b] = [0,\infty]. \tag{12}$$

```
(* FORTRAN output *)
outputww = 0;
```

The name of the produced FORTRAN file. For example,

```
(* fortran filename *)
fortranfname="gqxw.f";
```

After checking the above parameters, we execute our *Mathematica* code by pushing

```
[Shift]+[Enter]
```

keys. Then a message

```
Evaluating moment z^i W[x]...
```

will appear. If all moments are obtained analytically, evaluation of the Jacobi matrix begins. Otherwise the calculation will stop. Next, the abscissas and weights are calculated as described in the previous section. When the calculation is finished, the moments μ_k are compared with the GQ-approximation $\hat{\mu}_k$ (11). The two are displayed in N_a digits for $k = 0, 1, 2, \ldots, 2n-1$ together with the relative error

$$|\frac{\hat{\mu}_k - \mu_k}{\mu_k}|.$$

If the corresponding pairs coincide, we accept the results and the FORTRAN subroutine is being produced.

In the following example, the abscissas are stored in the array `x` when you call this subroutine by setting the input `n`, the degree of the GQ. Here, abscissas are given by x_j instead of z_j with $z_j = z(x_j)$. In the array `w`, the weights w_j or weights divided by the weight function $w_j/W(x_j)$ are stored depending on the value of the option `outputww`.

```
c      Abscissas and weights of Gaussian Quadrature
c      produced by Mathematica code: AWGQ
cccccc
c      for int_a^b{f(z(x))W(x)dx}
```

```
c      argument z(x) = x/Sqrt(1 + x**2)
c      weight   W(x) = (1 + x**2)**(-2)
c      region   a = 1
c      region   b = DirectedInfinity(1)
c      GQ order n <= 200
cccccc
       subroutine gqxw(x,w,n)
c      x: abscissas
c      w: weights
       implicit real*8(a-h,o-z)
       dimension x(200),w(200)
c
       if(n.eq.4) then
         x(1)=1.0545042737116109D0
         w(1)=3.1956375209299262D-2
         x(2)=1.3141812952767702D0
         w(2)=5.3744870692213551D-2
         x(3)=1.9558594860602826D0
         w(3)=4.2533155301151633D-2
         x(4)=3.9506935616438789D0
         w(4)=1.4464680496059708D-2
         return
       end if
c
       stop 'gqxw.f, n=', n
       end
```

3. TEST RUN

The test run presents integral (1) for the weight function, the argument and interval (8). The corresponding output list is that displayed in *Mathematica 2.2* for $n = 4$. Since the output from *Mathematica 4.0* is graphical, we have chosen a simpler output from *Mathematica 2.2*. Here we note that though the program can work in *Mathematica 2.2*, it does not control that the analytic moments μ_k are really calculated. The FORTRAN file produced was already shown in the previous section. In Table 1, we tabulate the computation time at Pentium IV 1.7 GHz and the two internal accuracies N_m and N_z necessary to get the final accuracy $N_a = 17$ for several values of n. In this case, internal accuracy N_m strongly depends on n but the internal accuracy N_z does not depend on n.

In Table 2, the internal accuracies N_m and N_z necessary to get the final accuracy $N_a = 17$ for fixed $n = 64$ are tabulated for two classical quadratures

(*Legendre* and *Laguerre*) and our sample GQ. Now, we can see that the internal accuracy N_z can also depend on the problems.

Table 1. **Computation time and internal accuracies for integral (1) with the weight function, the argument and the interval (8). n is the degree of GQ; N_m and N_z are the internal accuracies. The final output accuracy N_a is 17 digits. CPU is the computation time at Pentium IV 1.7 GHz**

n	N_m	N_z	CPU [s]
4	50	25	1
8	100	25	1
16	400	25	4
32	1300	25	21
64	6000	25	3135
96	12000	25	32323

Table 2. **Internal accuracy N_m and N_z necessary for three different GQ. The degree of GQ is $n = 64$. The final output accuracy N_a is 17 digits. CPU is the computation time at Pentium IV 1.7 GHz**

	$W(x)$	$[a, b]$	$z(x)$	N_m	N_z	CPU [s]
Legendre	1	$[-1, 1]$	x	1300	25	29
Laguerre	e^{-x}	$[0, \infty]$	x	1700	55	67
Our example	$\dfrac{1}{(1+x^2)^2}$	$[1, \infty]$	$\dfrac{x}{\sqrt{1+x^2}}$	6000	25	3135

4. TEST RUN OUTPUT

```
                    2 -2
weight W(x)=(1 + x )
                     x
argument z(x)=------------
                          2
              Sqrt[1 + x ]
integral region {a,b}={1, Infinity}
Degree of formula
 n=4
Precision
```

```
 moment Nm=50
 Jacobi Nz=25
 final  Na=17
 MachinePrecision=16
FORTRAN file
 gqxw.f
Evaluating moment z^i W[x]...
Orthogonal polynomial and Jacobi matrix ...
...50%
...75%
...100%
abscissas
{0.7256104344253013423139944, 0.7958055094055824274365386,
 0.8903722295270473536795167, 0.9694266243792582481606508}
weights
{0.03195637520929926237416762, 0.05374487069221355129581688,
 0.04253315530115163329036328, 0.014464680496059707847482635}
Check results
      exact moment,         moment by GQ,    (relative error)
  m0: 0.142699081698724155, 0.142699081698724155, (0.e-18)
  m1: 0.117851130197757921, 0.117851130197757921, (0.e-18)
  m2: 0.098174770424681039, 0.098174770424681039, (0.e-18)
  m3: 0.082495791138430545, 0.082495791138430545, (0.e-18)
  m4: 0.069920718545673853, 0.069920718545673853, (0.e-18)
  m5: 0.059767358886005803, 0.059767358886005803, (0.e-18)
  m6: 0.051512949091046158, 0.051512949091046158, (0.e-18)
  m7: 0.044755369682243782, 0.044755369682243782, (0.e-18)
*** computation time=1[sec]
```

5. FINAL REMARKS

We were calculating rotational three-body resonances in a new adiabatic approach, and the main numerical job to be done was a 2D quadrature in the (x, y)-plane [2]. In the hyperspheroidal coordinates we have $1 \leqslant x \leqslant \infty$ and $-1 \leqslant y \leqslant 1$ [3]. For the case of the molecular hydrogen ion H_2^+, the integration over y can be accurately carried out using the Gauss–Legendre scheme while for the semi-infinite x-region we have tried more than ten different methods but failed to find something even approximately reasonable. All subroutines with an automatic control of the accuracy provided finally inaccurate results. Thus, we were forced to use the fact that our integrands are approximately of the polynomial form

$$P(\frac{x}{\rho(x,y)}, \frac{y}{\rho(x,y)}),$$

with the volume element $(x^2 - y^2)/\rho^3(x,y)$, and $\rho(x,y) = 1 + c(x^2 + y^2 - 1)$ (c being the constant defined by the particle masses to be approximately 10^{-3} for the H_2^+-system). We have got the dramatic success. With $n_x = 64$ (Gauss quadrature as given here) and $n_y = 32$ (Gauss–Legendre case) all our 2D integrations using molecular-like orbitals could be done both fast and accurately.

That is why we believe that the multidimensional quadrature is to be the prime object of our paper.

REFERENCES

1. *Press W. H., Teukolsky S. A., Vetterling W. T., Flannery B. P.* Numerical Recipes in Fortran 77: The Art of Scientific Computing. Cambridge University Press, 1992.

2. *Matveenko A. V., Alt E. O., Fukuda H.* Rotational Three-Body Resonances: A New Adiabatic Approach // Few-Body Systems Suppl. 2001. V. 13. P. 140.

3. *Matveenko A. V., Abe Y.* // Few-Body Systems. 1987. V. 2. P. 127.

Received on May 13, 2004.

Корректор *Т. Е. Попеко*

Подписано в печать 10.06.2004.
Формат 60 × 90/16. Бумага офсетная. Печать офсетная.
Усл. печ. л. 0,93. Уч.-изд. л. 1,33. Тираж 315 экз. Заказ № 54474.

Издательский отдел Объединенного института ядерных исследований
141980, г. Дубна, Московская обл., ул. Жолио-Кюри, 6.
E-mail: publish@pds.jinr.ru
www.jinr.ru/publish/