Numer Algor (2009) 52:409–418
DOI 10.1007/s11075-009-9283-2

ORIGINAL RESEARCH

Variable-precision recurrence coefficients for nonstandard orthogonal polynomials

Walter Gautschi

Received: 6 January 2009 / Accepted: 26 February 2009 /
Published online: 18 March 2009

Abstract A symbolic/variable-precision procedure is described (and implemented in Matlab) that generates an arbitrary number `N` of recurrence coefficients for orthogonal polynomials to any given precision `nofdig`. The only requirement is the availability of a variable-precision routine for computing the first 2 `N` moments of the underlying weight function to any precision `dig > nofdig`. The procedure is applied to Freud, Bose–Einstein, and Fermi–Dirac orthogonal polynomials.

Keywords Variable-precision recurrence coefficients · Symbolic Chebyshev algorithm · Freud orthogonal polynomials · Bose–Einstein orthogonal polynomials · Fermi–Dirac oerthogonal polynomials

Mathematics Subject Classifications (2000) 3304 · 33C47

1 Introduction

The availability of symbolic/variable-precision software for orthogonal polynomials (for software in *Mathematica*, see the package `OrthogonalPolynomials` in [1]; for software in *Matlab*, the package `SOPQ` at http://www.cs.purdue.edu/archives/2002/wxg/codes) makes it possible to generate the respective recurrence coefficients to arbitrary precision also in nonstandard cases where they are not known explicitly. The basic vehicle is the Chebyshev

W. Gautschi (✉)
Department of Computer Sciences, Purdue University, West Lafayette, IN 47907-2066, USA
e-mail: wxg@cs.purdue.edu

algorithm, which allows us to compute the recurrence coefficients from the moments of the underlying weight function. Thus, all that is required is a procedure for evaluating the moments in variable-precision arithmetic. Since, as is well known, the problem of computing recurrence coefficients from moments is highly unstable, it will be necessary to employ high-precision computation to overcome the instability.

We illustrate this capability for a variety of weight functions, thereby, in part, extending existing software and tables to an arbitrary number of recurrence coefficients and arbitrary precision.

In Section 2 the basic algorithm is described. It uses the symbolic Matlab program[1] `schebyshev.m` implementing the Chebyshev algorithm and requires a symbolic routine `mom`*name*`.m` that generates the necessary moments. In the subsequent sections, the algorithm is applied to a variety of orthogonal polynomials, including Freud polynomials, Bose–Einstein polynomials, and Fermi–Dirac polynomials, corresponding, respectively, to weight functions $w(x) = |x|^\alpha \exp(-|x|^\beta)$ on $\mathbb{R}$ $(\alpha > -1,\ \beta > 0)$, $w(x) = [x/(e^x - 1)]^r$, $w(x) = [1/(e^x + 1)]^r$ on $\mathbb{R}_+$ $(r = 1, 2, 3, \dots)$.

2 Basic algorithm

Suppose we are given a (nonnegative) weight function w on some interval (a, b), $-\infty \le a < b \le \infty$, defining a set of (monic) orthogonal polynomials π_k, $k = 0, 1, 2, \dots$, where

$$\begin{aligned} \pi_{k+1}(x) &= (x - \alpha_k)\pi_k(x) - \beta_k \pi_{k-1}(x), \\ k &= 0, 1, 2, \dots, \\ \pi_0(x) &= 1, \quad \pi_{-1}(x) = 0 \end{aligned} \tag{1}$$

is the three-term recurrence relation satisfied by the polynomials π_k. Here, $\alpha_k = \alpha_k(w)$, $\beta_k = \beta_k(w)$ are certain constants depending on the weight function w, where $\alpha_k \in \mathbb{R}$, $\beta_k > 0$, and β_0, though arbitrary, is defined by $\beta_0 = \int_a^b w(x)\mathrm{d}x$. Our objective is to compute the first $\mathtt{N}$ of these coefficients to an accuracy of `nofdig` decimal places. To do so in Matlab, we store these coefficients in an $\mathtt{N} \times 2$ array `ab`, where the first column of `ab` contains $\alpha_0, \alpha_1, \dots, \alpha_{\mathtt{N}-1}$, and the second column $\beta_0, \beta_1, \dots, \beta_{\mathtt{N}-1}$ (cf. [3, §2.1]).

In principle, these coefficients can be computed from the first $2\,\mathtt{N}$ moments

$$\mu_k = \int_a^b x^k w(x)\mathrm{d}x, \quad k = 0, 1, \dots 2\,\mathtt{N} - 1, \tag{2}$$

[1] All Matlab routines referred to in this paper can be downloaded from the Purdue web site mentioned at the beginning of this section.

of w by means of the Chebyshev algorithm (cf. [3, §2.2], where $a_k = b_k = 0$, all k). Because of the severe ill-conditioning of the map from the $2\texttt{N}$ moments μ_k to the $2\texttt{N}$ recurrence coefficients $\alpha_k, \beta_k, 0 \leq k \leq \texttt{N}-1$, the desired accuracy of `nofdig` decimal digits can be achieved only if the computation is carried out in a precision considerably higher than `nofdig`. We determine this required precision iteratively by starting with a precision of `dig0=nofdig` decimal digits and increasing it in steps of $\texttt{dd} = 10$ digits until the desired accuracy of `nofdig` digits is achieved. (The choice $\texttt{dd} = 10$ was arrived at by experimentation as being reasonably efficient, but can easily be changed if deemed necessary.)

The procedure requires two variable-precision algorithms, the first one for computing the $2\texttt{N}$ moments μ_k in `dig`-digit arithmetic, and the second one implementing the Chebyshev algorithm in the same precision of `dig` decimal digits. If the given weight function w depends on parameters, then so does the first routine,

$$\texttt{mom} = \texttt{mom}\mathit{name}(\texttt{dig}, \texttt{N}, \ldots), \tag{3}$$

where *name* is the name for the orthogonal polynomials in question, and the three dots indicate the list of parameters. The second routine is the symbolic Chebyshev algorithm (cf. Section 1),

$$\texttt{ab} = \texttt{schebyshev}(\texttt{dig}, \texttt{N}, \texttt{mom}). \tag{4}$$

The details of the iterative procedure can be gathered from the following Matlab function.

```
% SR_name  This computes the first N recurrence
% coefficients αk, βk, k=0,1,...,N-1, to an
% accuracy of nofdig digits for the system of
% orthogonal polynomials named name. The output
% variable ab is the N×2 array of the nofdig-digit
% recurrence coefficients, and dig is the number
% of digits required to achieve the target
% precision of nofdig decimal places.
%
function [ab,dig]=sr_name(N,...,nofdig)
syms mom ab ab0 ab1
dd=10; dig0=nofdig;
i=dig0-dd;
maxerr=1;
while maxerr>.5*10^(-nofdig)
   i=i+dd; dig=i;
   mom=momname(dig,N,...);
   if i==dig0
      ab0=schebyshev(dig,N,mom);
```

```
            else
               ab1=schebyshev(dig,N,mom);
               serr=vpa(abs(ab1-ab0),dig);
               err=subs(serr);
               maxerr=max(max(err));
               ab0=ab1;
            end
         end
         ab=vpa(ab1,nofdig);
```

The procedure is essentially the same for any particular system of orthogonal polynomials and requires only the specification of the routine `mom`*name*. Still, there are instances where instead of an absolute error criterion, a relative one may be preferable, either for the α-coefficients, or the β-coefficients, or both. If, for example, we want to control the absolute error in the α-coefficients and the relative error in the β-coefficients, we let the statement defining `serr` be followed by

$$\texttt{serr}(:,2) = \texttt{vpa}(\texttt{abs}((\texttt{ab1}(:,2) - \texttt{ab0}(:,2))./\texttt{ab1}(:,2)), \texttt{dig}). \tag{5}$$

Similarly for other combinations of absolute and relative error. In the special case of symmetric weight functions, where all $\alpha_k = 0$, to control the relative error of the β-coefficients, it suffices to define `serr` by the right-hand side of (5).

3 Freud and half-range Hermite polynomials

Freud orthogonal polynomials are associated with the weight function

$$w(x) = |x|^\alpha \exp(-|x|^\beta), \quad x \in \mathbb{R},\ \alpha > -1,\ \beta > 0. \tag{6}$$

Its moments are easily found to be

$$\mu_k = \begin{cases} 0 & \text{if } k \text{ is odd,} \\ \dfrac{2}{\beta}\,\Gamma\left(\dfrac{k+\alpha+1}{\beta}\right) & \text{if } k \text{ is even.} \end{cases} \tag{7}$$

The `dig`-digit routine `mom`*freud*`.m`, therefore, looks as follows.

```
% MOMFREUD
%
function mom=momfreud(dig,N,alpha,beta)
digits(dig);
for k=1:2*N
   if rem(k,2)==0
      mom(k)=0;
```

```
    else
      mom(k)=vpa(2*gamma(vpa((k+alpha)/beta))/beta);
    end
  end
```

Since w is symmetric, all $\alpha_k = 0$. The special case $\beta = 2$, giving rise to generalized Hermite polynomials, may serve as a test example, since the recurrence coefficients are known to be $\beta_0 = \Gamma((\alpha+1)/2)$, $\beta_k = (k + \varepsilon_k \alpha)/2$, where $\varepsilon_k = 0$ if k is even, and $\varepsilon_k = 1$ if k is odd. Running `sr_`*freud*`(N,alpha,beta,nofdig)` with `N` $= 100$, `alpha` $= \pm 1/2$, `beta` $=$ 2, and `nofdig` $= 40$, indeed passed the test.

The routine was then applied to generate the first `N` $= 100$ recurrence coefficients β_k for the Freud weight (6) with $\alpha = 0$, $\beta = 4{:}2{:}10$, calling for `nofdig`=32 digit accuracy. The results can be found in the files `coefffreud`4–10 on the web site http://www.cs.purdue.edu/archives/2001/wxg/tables. In the case $\beta = 4$, they are in complete agreement with results obtained with *Mathematica* by A. Cvetković and G. V. Milovanović, using a different method (the nonlinear recurrence relation in [7]). Each case took about 30 min. to run on a Sun Ultra 5 workstation and required as much as 112-digit calculations.

As a matter of curiosity, when $\beta = 6$, we observed that $\beta_1 = \beta_2$. Generally, however, the β_k slowly increase monotonically (except for the first few). In fact, for $\beta \geq 2$, the following asymptotic result holds (cf. [6, eq (1.10)], adapted to our notations, which differ from those in [6]),

$$\beta_k = \frac{1}{4}\,(\gamma k)^{2/\beta} + O(k^{2/\beta - 2}), \quad k \to \infty, \tag{8}$$

where

$$\gamma = \frac{\Gamma(\beta/2)\Gamma(1/2)}{\Gamma((\beta+1)/2)}\,. \tag{9}$$

Our computations, moreover, suggest that

$$\frac{4\beta_k}{(\gamma k)^{2/\beta}} \downarrow 1 \quad \text{for } k \geq k_0(\beta), \tag{10}$$

where $k_0(\beta)$ is relatively small (equal to $5, 6, 4, 4$ for respectively $\beta =$ $4, 6, 8, 10$).

A weight function somewhat related to Freud's is the half-range Hermite weight function

$$w(x) = \exp(-x^2), \quad x \in \mathbb{R}_+, \tag{11}$$

whose moments are given by

$$\mu_k = \frac{1}{2}\,\Gamma\left(\frac{k+1}{2}\right), \quad k = 0, 1, 2, \ldots, 2\,\mathrm{N} - 1, \tag{12}$$

and evaluated to `dig` decimal places by the routine `mom`*halfrangehermite*`.m`:

```
% MOMHALFRANGEHERMITE
%
function mom=momhalfrangehermite(dig,N)
digits(dig);
for k=1:2*N
   mom(k)=vpa(gamma(vpa(k/2))/2);
end
```

The first 100 recurrence coefficients are generated by the routine `sr_`*halfrangehermite*`(N,nofdig)` to `nofdig` = 32 decimal places and stored in the file `coeffhalfrangehermite` of the web site indicated above. They agree (except for occasional last-digit discrepancies of one unit) with all 25-digit values produced by other methods and stored in the `OPQ` file `abhrhermite` (cf. [2, Example 2.31]).

4 Bose–Einstein polynomials

Polynomials orthogonal with respect to the weight function

$$\left(\frac{x}{e^{\omega x}-1}\right)^r \quad \text{on } \mathbb{R}_+,\ \omega>0,\ r\in\mathbb{N}_+$$

we call Bose–Einstein polynomials since for $r=1$ the weight function in statistical mechanics defines a Bose–Einstein distribution. For the purpose of computing their recurrence coefficients, it suffices to consider the special case $\omega=1$, since the α-coefficients in this special case, if divided by ω, and the β-coefficients divided by ω^2, yield the recurrence coefficients in the general case $\omega>0$. So let the weight function be

$$w(x)=\left(\frac{x}{e^{x}-1}\right)^r \quad \text{on } \mathbb{R}_+,\ r\in\mathbb{N}_+. \tag{13}$$

The moments of w,

$$\mu_k^{(r)}=\int_0^\infty \frac{x^{k+r}}{(e^x-1)^r}\,\mathrm{d}x, \qquad k=0,1,2,\ldots,2\mathrm{N}-1, \tag{14}$$

are known explicitly when $r=1$,

$$\mu_k^{(1)}=\Gamma(k+2)\zeta(k+2) \tag{15}$$

(cf. [5, eq 3.411.1]). To obtain the moments for any fixed $r>1$, we observe that for $\rho>1$,

$$\begin{aligned}\mu_{k+1}^{(\rho-1)}&=\int_0^\infty \frac{x^{k+\rho}}{(e^x-1)^{\rho-1}}\,\mathrm{d}x=\int_0^\infty \frac{x^{k+\rho}}{(e^x-1)^{\rho}}\,[e^x-1]\mathrm{d}x\\&=\int_0^\infty \frac{x^{k+\rho}}{(e^x-1)^{\rho}}\,e^x\mathrm{d}x-\mu_k^{(\rho)},\end{aligned}$$

that is,

$$\mu_k^{(\rho)} = \int_0^\infty \frac{x^{k+\rho}}{(e^x-1)^\rho}\, e^x \mathrm{d}x - \mu_{k+1}^{(\rho-1)}.$$

The integral on the right can be evaluated using integration by parts,

$$\begin{aligned}\int_0^\infty \frac{x^{k+\rho}}{(e^x-1)^\rho}\, e^x \mathrm{d}x &= -\frac{1}{\rho-1}\int_0^\infty x^{k+\rho}\frac{\mathrm{d}}{\mathrm{d}x}\left\{\frac{1}{(e^x-1)^{\rho-1}}\right\}\mathrm{d}x \\ &= -\frac{1}{\rho-1}\left\{\frac{x^{k+\rho}}{(e^x-1)^{\rho-1}}\Big|_0^\infty - (k+\rho)\int_0^\infty \frac{x^{k+\rho-1}}{(e^x-1)^{\rho-1}}\,\mathrm{d}x\right\} \\ &= -\frac{1}{\rho-1}\left\{-(k+\rho)\mu_k^{(\rho-1)}\right\}.\end{aligned}$$

Therefore,

$$\mu_k^{(\rho)} = \frac{k+\rho}{\rho-1}\,\mu_k^{(\rho-1)} - \mu_{k+1}^{(\rho-1)}. \tag{16}$$

Since the moments $\mu_k^{(1)}$ are known by (15), the relation (16) allows us to compute from the first $2\mathrm{N}+r$ of them recursively (in ρ) all the desired moments in (14). For example, with $\rho=2$, we find $\mu_k^{(2)} = (k+2)\mu_k^{(1)} - \mu_{k+1}^{(1)}$, hence, by (15), $\mu_k^{(2)} = \Gamma(k+3)[\zeta(k+2)-\zeta(k+3)]$, which is a known result (cf. [5, eq 3.423.1]).

The following routine `mom`*boseeinstein*`.m` implements the above procedure in `dig`-digit arithmetic.

```
% MOMBOSEEINSTEIN
%
function mom=momboseeinstein(dig,N,r)
digits(dig);
for k=1:2*N+r
   m(k,1)=gamma(vpa(k+1))*zeta(vpa(k+1));
   if r==1 & k<=2*N
      mom(k)=m(k,1);
   end
end
if r>1
   for rho=2:r
      for k=1:2*N+r-rho
         m(k,rho)=vpa(((k+rho-1)/(rho-1))*m(k,rho-1)...
            -m(k+1,rho-1));
```

```
                if rho==r
                   mom(k)=m(k,rho);
                end
             end
          end
       end
```

In the corresponding procedure `sr_`*boseeinstein*`(N,r,nofdig)`, it is advisable to use the relative error criterion, both for the α- and β-coefficients (cf. the final paragraph in Section 2). When run with `N` $= 100$, `r` $= 1$ and 2, `nofdig` $= 32$, it reproduced the first 40 recurrence coefficients in Table 1 and Table 2 of [4, Appendix 1], with almost perfect agreement in all 25 decimal digits given there, the exceptions being occasional discrepancies of one unit in the last decimal place. The running times on a Sun Ultra 5 workstation, for $r = 1:4$, were respectively about 52, 72, 86, and 102 min., and the required precisions 142, 182, 212, and 242 digits. The results, along with those for `r` $= 3$ and 4, are posted in the files `coeffboseeinstein1`–4 on the web site given in Section 3.

5 Fermi–Dirac polynomials

We call Fermi–Dirac polynomials those that are orthogonal with respect to the weight function

$$\left(\frac{1}{e^{\omega x}+1}\right)^r \quad \text{on } \mathbb{R}_+,\ \omega > 0,\ r \in \mathbb{N}_+$$

since, for $r = 1$, it defines in statistical mechanics a Fermi–Dirac distribution. As explained in Section 4, it suffices to deal with the case $\omega = 1$,

$$w(x) = \left(\frac{1}{e^{x}+1}\right)^r \quad \text{on } R_+,\ r \in \mathbb{N}_+. \tag{17}$$

To compute the moments

$$\mu_k^{(r)} = \int_0^\infty \frac{x^k}{(e^x+1)^r}\,\mathrm{d}x, \qquad k = 0, 1, \ldots, 2\mathrm{N}-1, \tag{18}$$

of w, for fixed $r \geq 1$, we first observe that

$$\begin{aligned}\mu_0^{(1)} &= \int_0^\infty \frac{\mathrm{d}x}{e^x+1} = \int_1^\infty \frac{\mathrm{d}t}{t(t+1)} = \lim_{u\to\infty}\int_1^u \left(\frac{1}{t} - \frac{1}{t+1}\right)\mathrm{d}t \\ &= \lim_{u\to\infty}\left(\ln\frac{u}{u+1} + \ln 2\right) = \ln 2,\end{aligned} \tag{19}$$

and, for $k > 0$,

$$\mu_k^{(1)} = (1-2^{-k})\Gamma(k+1)\zeta(k+1), \qquad k = 1, 2, \ldots, 2\mathrm{N}-1 \tag{20}$$

(cf. [5, eq 3.411.3]). Furthermore,

$$\mu_k^{(\rho+1)} - \mu_k^{(\rho)} = \int_0^\infty \frac{x^k}{(e^x+1)^{\rho+1}}[1-(e^x+1)]\mathrm{d}x$$
$$= -\int_0^\infty x^k \frac{e^x}{(e^x+1)^{\rho+1}}\mathrm{d}x = \frac{1}{\rho}\int_0^\infty x^k \frac{\mathrm{d}}{\mathrm{d}x}\left\{\frac{1}{(e^x+1)^\rho}\right\}\mathrm{d}x, \tag{21}$$

and using integration by parts,

$$\mu_k^{(\rho+1)} = \mu_k^{(\rho)} - \begin{cases} \dfrac{1}{\rho\cdot 2^\rho} & \text{if } k=0, \\ \dfrac{k}{\rho}\,\mu_{k-1}^{(\rho)} & \text{if } k>0. \end{cases} \tag{22}$$

The Eq. 22 with $k=0$, in combination with (19), allows us to compute

$$\mu_0^{(\rho)} \quad \text{for} \quad \rho = 1, 2, \ldots, r, \tag{23}$$

which, in particular, gives us the zero-order moment $\mu_0^{(r)}$. Next, (20) can be used to compute

$$\mu_k^{(1)} \quad \text{for} \quad k = 1, 2, \ldots, 2\mathrm{N}-1, \tag{24}$$

which, if $r=1$, gives us the remaining higher-order moments. If $r>1$, we use (23) and (24) as initial values to compute from (22), successively for $k=1,2,\ldots,2\mathrm{N}-1$, the quantities $\mu_k^{(\rho)}$ for $\rho=2,3,\ldots,r$, which yields the higher-order moments $\mu_k^{(r)}$, $k=1,2,\ldots,2\mathrm{N}-1$.

The procedure outlined above is implemented in the following `dig`-digit routine `mom`*fermidirac*`.m`.

```
% MOMFERMIDIRAC
%
function mom=momfermidirac(dig,N,r)
digits(dig);
m(1,1)=vpa('log(2)');
if r==1
   mom(1)=m(1,1);
else
   for rho=1:r-1
      m(1,rho+1)=vpa((m(1,rho)-1/(rho*(2^rho)));
   end
   mom(1)=m(1,r);
end
```

```
for k=2:2*N
   m(k,1)=vpa((1-2^(1-k))*gamma(vpa(k))*zeta(vpa(k)));
   if r==1
      mom(k)=m(k,1);
   end
end
if r>1
   for k=2:2*N
      for rho=1:r-1
         m(k,rho+1)=vpa(m(k,rho)-(k-1)*m(k-1,rho)/rho);
      end
      mom(k)=m(k,r);
   end
end
```

The procedure `sr`_*fermidirac*`(N,r,nofdig)` (with relative error control in both the α- and β-coefficients) was run with $\mathtt{N} = 100$, $\mathtt{r} = 1:4$ and $\mathtt{nofdig} = 32$. The results, obtained with an effort comparable to the one in the case of Bose–Einstein coefficients, are stored in the files `coefffermidirac`1–4 on the web site mentioned in Section 3. In the process, Tables 3–4 in [4] of the first 40 recurrence coefficients for $r = 1$ and 2 were checked and found to be correct in all 25 decimal digits given there.

Acknowledgement The author thanks Doron Lubinsky for Reference [6].

References

1. Cvetković, A., Milovanović, G.V.: The mathematica package "`OrthogonalPolynomials`". Facta Univ. (Niš), Math. Inform. **19**, 17–36 (2004)
2. Gautschi, W.: Orthogonal polynomials: computation and approximation. Numerical Mathematics and Scientific Computation. Oxford University Press, Oxford (2004)
3. Gautschi, W.: Orthogonal polynomials, quadrature, and approximation: computational methods and software (in Matlab). In: Marcellán, F., Van Assche, W. (eds.) Orthogonal Polynomials and Special Functions: Computation and Applications. Lecture Notes in Mathematics, vol. 1883. Springer, Berlin (2006)
4. Gautschi, W., Milovanović, G.V.: Gaussian quadrature involving Einstein and Fermi functions with an application to summation of series. Math. Comput. **44**(169), 177–190 (1985)
5. Gradshteyn, I.S., Ryzhik, I.M.: Tables of Integrals, Series, and Products, 7th edn. Elsevier/Academic, Amsterdam (2007)
6. Kriecherbauer, T., McLaughlin, K.T.-R.: Strong asymptotics of polynomials orthogonal with respect to Freud weights. Int. Math. Res. Not. **6**, 299–333 (1999)
7. Noschese, S., Pasquini, L.: On the nonnegative solution of a Freud three-term recurrence. J. Approx. Theory **99**(1), 54–67 (1999)