Computing polynomials orthogonal with respect to densely oscillating and exponentially decaying weight functions and related integrals

Walter Gautschi

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907-2066
U.S.A.

Abstract

Software (in Matlab) is developed for computing variable-precision recurrence coefficients for orthogonal polynomials with respect to the weight functions $1+\sin(1/t)$, $1+\cos(1/t)$, $e^{-1/t}$ on $[0,1]$, as well as $e^{-1/t-t}$ on $[0,\infty]$ and e^{-1/t^2-t^2} on $[-\infty,\infty]$. Numerical examples are given involving Gauss quadrature relative to these weight functions.

Key words:
Orthogonal polynomials, densely oscillating weight functions, exponentially decaying weight functions, Gaussian quadrature

1 Introduction

The availability of constructive methods and related software for orthogonal polynomials makes it possible to numerically generate such polynomials relative to weight functions of ever increasing complexity. This is illustrated here in the case of weight functions on $[0,1]$, such as $1+\sin(1/t)$ or $1+\cos(1/t)$, that are densely oscillating near the origin, and weight functions decaying exponentially fast near the origin, such as $e^{-1/t}$ on $[0,1]$, none of which is in

Email address: wxg@cs.purdue.edu (Walter Gautschi).
URL: http://www.cs.purdue.edu/people/wxg (Walter Gautschi).

the Szegö class. We also consider the weight function $e^{-1/t-t}$ on $[0,\infty]$ and e^{-1/t^2-t^2} on $[-\infty,\infty]$. In all these cases, the moments of the weight function are expressible in terms of special functions (sine, cosine, and exponential integrals and modified Bessel functions), which can all be evaluated to arbitrary precision. With the moments at hand, an algorithm due to Chebyshev can be applied to compute the recurrence coefficients for the orthogonal polynomials from the given moments. Although there is a great deal of ill-conditioning involved in this approach, it can be surmounted by symbolic computation and variable-precision arithmetic as supplied, e.g., by the current release of Matlab.

A number of Matlab routines are developed for computing the recurrence coefficients of the desired orthogonal polynomials to arbitrary precision, and files are produced containing the first 40 of the coefficients to 34 decimal digits. All Matlab routines and files referenced in this paper are downloadable individually from the web site

`http://www.cs.purdue.edu/archives/2002/wxg/codes/`

containing the Matlab package `OPQ` along with additional routines. Numerical examples involving integration of weighted integrals by Gaussian quadrature are provided illustrating the power and effectiveness of the software.

2 Densely oscillating trigonometric weight functions

2.1 The weight function $w(t) = 1 + \sin(1/t)$ on $[0,1]$

Constructing orthogonal polynomials relative to this weight function is a challenging task, given the densely oscillating behavior of w near the origin. The only approach that seems feasible is one based on the moments of w; indeed, the Chebyshev algorithm (cf. [2, §2.1.7]) generates the three-term recurrence relation for the (monic) orthogonal polynomials $\pi_k(\,\cdot\,) = \pi_k(\,\cdot\,;w)$ from the moments

$$\mu_k = \int_0^1 t^k[1+\sin(1/t)]\mathrm{d}t, \quad k = 0,1,2,\ldots\,. \tag{1}$$

In view of the well-known ill-conditioning of this approach ([2, §2.1.4]), it can only succeed if high-precision arithmetic is used. We propose to invoke the symbolic/variable-precision capabilities of Matlab 6, Release 12, for this purpose.

To compute the moments (1), we first consider the "core moments" $\mu_k^0 = \int_0^1 t^k \sin(1/t)\mathrm{d}t$. By the change of variable $t \mapsto 1/t$, one has

$$\mu_k^0 = \int_1^\infty t^{-(k+2)} \sin t \, \mathrm{d}t,$$

and two integrations by part will show that μ_k^0 satisfies the recurrence relation

$$\mu_{k+1}^0 = \frac{1}{k+2}\left[\frac{1}{k+1}(\cos 1 - \mu_{k-1}^0) + \sin 1\right], \quad k = 0, 1, 2, \ldots . \tag{2}$$

Here,

$$\mu_{-1}^0 = \int_1^\infty \frac{\sin t}{t}\,\mathrm{d}t = \frac{\pi}{2} - \mathrm{Si}(1), \quad \mu_0^0 = \int_1^\infty \frac{\sin t}{t^2}\,\mathrm{d}t = \sin 1 - \mathrm{Ci}(1), \tag{3}$$

where Si and Ci are the sine and cosine integrals, respectively (cf. [1, eqns 5.2.1 and 5.2.2]). In terms of the core moments μ_k^0, the actual moments μ_k are simply

$$\mu_k = \frac{1}{k+1} + \mu_k^0, \quad k = 0, 1, 2, \ldots . \tag{4}$$

The Chebyshev algorithm now takes the first $2n$ moments μ_k, $k = 0, 1, \ldots, 2n-1$, and from them produces the first n recurrence coefficients α_k, β_k, $k = 0, 1, \ldots, n-1$, in the three-term recurrence relation

$$\begin{aligned} \pi_{k+1}(t) = (t - \alpha_k)\pi_k(t) - \beta_k \pi_{k-1}(t), \quad k = 0, 1, \ldots, n-1, \\ \pi_{-1}(t) = 0, \quad \pi_0(t) = 1, \end{aligned} \tag{5}$$

for the orthogonal polynomials π_k. This is implemented in the Matlab routine `schebyshev.m`, a symbolic version of the `OPQ` routine `chebyshev.m`. The Matlab script below uses this routine with `d`-decimal-digit arithmetic to generate the first `N` recurrence coefficients in (5).

```
% SR_SIN0 Symbolic/variable-precision recurrence coefficients
%    for the weight function w(x)=1+sin(1/x) on [0,1]
%
syms mom ab
digits(d); dig=d;
mom(1)='sin(1)-Ci(1)';
mom(2)='(Si(1)-pi/2+sin(1)+cos(1))/2';
for k=3:2*N
   mom(k)=(('cos(1)'-mom(k-2))/(k-1)+'sin(1)')/k;
end
for k=1:2*N
   mom(k)=mom(k)+1/k;
end
ab=schebyshev(dig,N,mom);
```

How much precision is needed to obtain 34 good decimal digits for the first $\mathtt{N} = 40$ recurrence coefficients? Using $\mathtt{d} = 70$ in the above routine, one obtains

$$\alpha_{39} = .510028114107\underline{742}\ldots,$$
$$\beta_{39} = .609363829421\underline{165}\ldots(-1).$$

Comparing this with 95-digit computation, one observes agreement to only the first 12 digits; the digits underlined are therefore in error. As k is decreased down from 39, the results for α_k, β_k gradually become more accurate. This behavior is typical for the ill-conditioning of the underlying moment map, which here, in the worst case, causes a loss of $70-12 = 58$ digits. To obtain 34 good digits for all results, one thus expects to need about 92-digit computation. It is found that 95- and 100-digit computation indeed yield results which agree to at least 37 digits. Rounded to 34 digits, the results are stored in the file `absin0`; a few beginning and final entries of the file are displayed below to 16 digits. Note that the αs are hovering around 1/2 and the βs around 1/16, the limit values that would be attained if the weight function were in the Szegö class.

```
ab =
[    .5841029561609566,    1.504067061906928]
[    .4634474607770499, .7094822535096882e-1]
[    .4977629714178322, .7892077774694954e-1]
[    .5356590088623750, .5547885019105795e-1]
[    .4669144430825117, .6259489484316939e-1]
[    .4951204560106238, .7262419731845188e-1]
          ... ...             ... ...
[    .5109646577717308, .5919672749794708e-1]
[    .4919346767523436, .6218164541801303e-1]
[    .4973070307106440, .6494413841533854e-1]
[    .5100281141083978, .6093638294208964e-1]
```

Some of the early coefficients α_k, β_k were checked (successfully) by a quadruple-precision Fortran program.

Example 1.1. Compute the integral $\int_0^1 f(t)\sin(1/t)\mathrm{d}t$, where $f(t) = \tan((\frac{1}{2}\pi - \delta)t)$, $\delta = .1$.

We use

$$\int_0^1 f(t)\sin(1/t)\mathrm{d}t = \int_0^1 f(t)[1+\sin(1/t)]\mathrm{d}t - \int_0^1 f(t)\mathrm{d}t \tag{6}$$

and apply to the first integral on the right Gaussian quadrature relative to the weight function w and to the second Gauss–Legendre quadrature on $[0,1]$. This is carried out in the following script:

```
% INTSIN0 Integration relative to the weight function
%    w(t)=sin(1/t)
%
f1='%5.0f %21.13e %21.13e %21.13e\n';
fprintf('\n')
disp('     n      n-point Gauss')
load -ascii absin0;
ab=absin0; abl=r_jacobi01(40);
delta=.1;
for n=4:4:36
   xwl=gauss(n,abl); xw=gauss(n,ab);
   intl=sum(xwl(:,2).*tan((pi/2-delta).*xwl(:,1)));
   ints=sum(xw(:,2).*tan((pi/2-delta).*xw(:,1)));
   int=ints-intl;
   fprintf(f1,n,int)
end
```

The output of the script is shown in Table 1.1.

```
>> intsin0

    n      n-point Gauss
    4   1.2716655036125e+00
    8   1.2957389942560e+00
   12   1.2961790099686e+00
   16   1.2961860624657e+00
   20   1.2961861691603e+00
   24   1.2961861708344e+00
   28   1.2961861708631e+00
   32   1.2961861708636e+00
   36   1.2961861708636e+00
```

Table 1.1. Results for Example 1.1

There is practically no cancellation occurring when computing the difference of the two integrals on the right of (6).

Convergence is seen to be relatively fast (it is faster for $\delta = .3$ or $\delta = .5$), but is the limit correct? Seeing some other quadrature scheme yielding similar, albeit less accurate, results would give us more confidence in the limit value of Table 1.1. We tried the N-point Gauss–Legendre rule on $[0, 1]$ for large values of N, applied directly to the integral on the left of (6), and found

N	N-point GL
200	1.29623...e+00
400	1.29614...e+00
600	1.29621...e+00
800	1.29617...e+00
1000	1.29619...e+00

The first four digits are the same as in Table 1.1 and are established rather quickly (relatively speaking). Evidently they correspond to the part of the integral away from zero. The difficult behavior of the integrand very close to zero causes the remaining digits to converge very hesitantly. Nevertheless, the corrrectness of the limit in Table 1.1 seems to be beyond doubt.

2.2 The weight function $w(t) = 1 + \cos(1/t)$ on $[0, 1]$

Analogously to §2.1, we define

$$\mu_k = \int_0^1 t^k [1 + \cos(1/t)] \mathrm{d}t, \quad k = 0, 1, 2, \ldots, \tag{7}$$

and $\mu_k^0 = \int_0^1 t^k \cos(1/t) \mathrm{d}t$, and find for μ_k^0 the recurrence relation

$$\mu_{k+1}^0 = \frac{1}{k+2} \left[\cos 1 - \frac{1}{k+1} \left(\sin 1 + \mu_{k-1}^0 \right) \right], \quad k = 0, 1, 2, \ldots, \tag{8}$$

with

$$\mu_{-1}^0 = -\mathrm{Ci}(1), \quad \mu_0^0 = \cos 1 + \mathrm{Si}(1) - \frac{\pi}{2}. \tag{9}$$

This yields μ_k as in (4) and gives rise to a routine `sr_cos0.m` similar to `sr_sin0.m`. The first `N` = 40 recurrence coefficients can again be obtained to 34 decimal digits using 95- resp. 100-digit arithmetic; they are stored in the file `abcos0`. The first six and last four, rounded to 16 digits, are shown below.

```
ab =
[    .5658844678158393,    .9155890494404261]
[    .4366405849780814,    .1027346437337914]
[    .5796274186498604, .5270964252517353e-1]
[    .4239578421345528, .6199408887322155e-1]
[    .5354625727005339, .7671995359431511e-1]
[    .5373660897782133, .4556076502616208e-1]
          ... ...            ... ...
[    .5186905674873959, .6081660814714736e-1]
[    .5003375738978038, .5901663246762277e-1]
[    .4979767366985249, .6422084071680227e-1]
[    .4970744762251129, .6172998651366585e-1]
```

Example 1.2. Compute $\int_0^1 f(t)\sin(1/t+t)\mathrm{d}t$.

We write

$$\begin{aligned}\int_0^1 f(t)\sin(1/t+t)\mathrm{d}t \\ = \int_0^1 f(t)\cos t\,[1+\sin(1/t)]\mathrm{d}t + \int_0^1 f(t)\sin t\,[1+\cos(1/t)]\mathrm{d}t \\ - \int_0^1 f(t)(\cos t+\sin t)\mathrm{d}t\end{aligned} \tag{10}$$

and use Gaussian quadrature relative to the weight functions $1+\sin(1/t)$ and $1+\cos(1/t)$ for the first two integrals, and Gauss–Legendre quadrature on $[0,1]$ for the last. The Matlab script below implements this for $f(t)=e^{-t}$.

```
% INTSIN01 Integration relative to the weight function
%    w(t)=sin(1/t+t)
%
f1='%5.0f %21.13e\n';
fprintf('\n')
disp('     n     n-point Gauss')
load -ascii absin0; abs=absin0;
load -ascii abcos0; abc=abcos0;
abl=r_jacobi01(40);
for n=1:7
   xwl=gauss(n,abl); xws=gauss(n,abs); xwc=gauss(n,abc);
   intl=sum(xwl(:,2).*exp(-xwl(:,1)).*(cos(xwl(:,1))...
     +sin(xwl(:,1))));
   ints=sum(xws(:,2).*exp(-xws(:,1)).*cos(xws(:,1)));
   intc=sum(xwc(:,2).*exp(-xwc(:,1)).*sin(xwc(:,1)));
   int=ints+intc-intl;
   fprintf(f1,n,int)
end
```

The results are shown in Table 1.2.

```
>> intsin01

    n      n-point Gauss
    1   1.5532688394788e-01
    2   1.5896667464309e-01
    3   1.5875741460598e-01
    4   1.5875671404065e-01
    5   1.5875671541036e-01
    6   1.5875671541391e-01
    7   1.5875671541391e-01
```

Table 1.2. Results for Example 1.2 when $f(t) = e^{-t}$

Gauss–Legendre quadrature applied directly to the integral on the left of (10), with N=200:200:1000 points, manages to confirm only the first two digits, but enough to have confidence in the rapidly converging sequence of approximations displayed in Table 1.2.

3 Rapidly decaying exponential weight functions

Replacing the trigonometric functions in §2.1 and 2.2 by the exponential function yields rapidly decaying exponential weight functions.

3.1 The weight function $w(t) = \exp(-1/t)$ on $[0, 1]$

Proceeding as in §2, we start from the moments

$$\mu_k = \int_0^1 t^k e^{-1/t} \mathrm{d}t, \quad k = 0, 1, 2, \ldots, \tag{11}$$

and use the Chebyshev algorithm in high precision to generate the recurrence coefficients of the desired orthogonal polynomials $\pi_k(\,\cdot\,; w)$. By a change of variable, we have

$$\mu_k = \int_1^\infty t^{-(k+2)} e^{-t} \mathrm{d}t = E_{k+2}(1), \tag{12}$$

where E_n is the exponential integral (cf. [1, eqn 5.1.4]), which can be computed recursively ([1, eqn 5.1.14]), giving

$$\mu_{k+1} = \frac{1}{k+2}\,(e^{-1} - \mu_k), \quad k = 0, 1, 2, \ldots, \qquad \mu_0 = E_2(1). \tag{13}$$

This is incorporated in the routine `sr_exp0.m`, as shown below.

```
% SR_EXP0 Symbolic/variable-precision recurrence coefficients
%    for the weight function w(x)=exp(-1/x) on [0,1]
%
syms mom ab
digits(d); dig=d;
mom(1)=vpa('Ei(2,1)',d);
for k=2:2*N
   mom(k)=('exp(-1)'-mom(k-1))/k;
end
ab=schebyshev(dig,N,mom);
```

For `N` = 40, the choice `d` = 70 yields results correct to 9 or more decimal digits, corresponding to a loss of as much as 61 digits. With `d` = 95 and `d` = 100, therefore, we can again secure 34 correct decimals. The respective results are stored in the file `abexp0`.

Example 3.1. Compute $\int_0^1 \ln(1+t)e^{-1/t}\mathrm{d}t$.

The routine `sr_exp0.m` is used, in conjunction with the routine `gauss.m`, to generate Gaussian quadrature rules for the weight function $\exp(-1/t)$, which, applied to the integral of Example 3.1, produce results as shown in Table 3.1.

```
>> intexp0

 n      n-point Gauss
 2   8.1262554100479e-02
 4   8.1255735149253e-02
 6   8.1255733983155e-02
 8   8.1255733982820e-02
10   8.1255733982819e-02
12   8.1255733982819e-02
```

Table 3.1. Results for Example 3.1

The same limit value is obtained by 102-point Gauss–Laguerre quadrature of $e^{-1}(1+t)^{-2}\log(1+(1+t)^{-1})$; see the routine `intexp0.m`.

3.2 The weight function $w(t) = \exp(-1/t - t)$ on $[0, \infty]$

Here, the moments of w are expressible in terms of the modified Bessel functions according to ([3, eqn 3.471.9])

$$\mu_k = \int_0^\infty t^k e^{-(1/t+t)} \mathrm{d}t = 2K_{k+1}(2), \quad k = 0, 1, 2, \ldots . \tag{14}$$

This can be computed recursively as follows (cf. [1, eqn 9.6.26]):

$$\begin{aligned} \mu_{k+1} &= (k+1)\mu_k + \mu_{k-1}, \quad k = 0, 1, 2, \ldots, \\ \mu_{-1} &= 2K_0(2), \quad \mu_0 = 2K_1(2), \end{aligned} \tag{15}$$

and gives rise to the routine `sr_exp0inf.m`:

```
% SR_EXP0INF Symbolic/variable-precision recurrence
%    coefficients for the weight function
%    w(x)=exp(-1/x-x) on [0,inf]
%
syms mom ab
digits(d); dig=d;
mom(1)=vpa('2*BesselK(1,2)',dig);
mom(2)=mom(1)+vpa('2*BesselK(0,2)',dig);
for k=3:2*N
   mom(k)=(k-1)*mom(k-1)+mom(k-2);
end
ab=schebyshev(dig,N,mom);
```

In the case `N`$=40$, `d`$=70$, the loss of accuracy is at most 36 digits, leaving us with about 34 correct digits. This is confirmed by running `sr_exp0inf` with `d`$=75$ and `d`$=80$, the results of which agree to at least 38 digits. The file `abexp0inf` is provided with 34-digit values of the desired recurrence coefficients. The beginning and end of the file, rounded to 16 decimals, are shown below.

```
ab =
[ 1.814307758763789, .2797317636330449]
[ 3.647885050815283, 1.336902874017094]
[ 5.563608408242503, 4.576187502809998]
[ 7.510248881089434, 9.776110045536486]
[ 9.472385776425876, 16.95364518291704]
[ 11.44360258233455, 26.11622048172850]
       ... ...             ... ...
[ 73.23900952424970, 1300.294223771922]
[ 75.23687505008481, 1373.374066736622]
[ 77.23481475899122, 1448.453190623454]
[ 79.23282425676095, 1525.531620678047]
```

Example 3.2. Compute $\int_0^\infty J_0(t)e^{-1/t-t}\mathrm{d}t$.

Gauss quadrature relative to the weight function $w(t) = \exp(-1/t - t)$ yields results as shown in Table 3.2.

```
>> intexp0inf

 n      n-point Gauss
 4   1.1162402700893e-01
 8   1.1153340191221e-01
12   1.1153288987809e-01
16   1.1153289176609e-01
20   1.1153289176207e-01
24   1.1153289176207e-01
```

Table 3.2. Results for Example 3.2

In contrast, Gauss–Laguerre quadrature of $J_0(t)e^{-1/t}$ requires $\mathtt{N} = 1000$ points only to get 11-digit accuracy. Such is the debilitating effect of the strong decay of $e^{-1/t}$ as $t \downarrow 0$. On the other hand, writing

$$\int_0^\infty J_0(t)e^{-1/t-t}\mathrm{d}t = \int_0^1 [J_0(t)e^{-t}]e^{-1/t}\mathrm{d}t + e^{-1}\int_0^\infty [J_0(1+t)e^{-1/(1+t)}]e^{-t}\mathrm{d}t$$

and evaluating the first integral on the right by n-point Gauss quadrature with respect to the weight function $e^{-1/t}$ (cf. §3.1) and the second by n-point Gauss–Laguerre quadrature yields also 11-digit accuracy, but already with $\mathtt{N} = 40$. Nevertheless, this requires two different, more slowly convergent, quadrature routines, compared to just one in the solution given in Example 3.2. The slowdown is actually caused by the Gauss-Laguerre quadrature of the second integral. See the routine `intexp0inf.m`.

3.3 The weight function $\exp(-1/t^2 - t^2)$ *on* $[-\infty, \infty]$

Similarly as in §3.2, one can express the moments in terms of modified Bessel functions: all moments of odd order are zero, while those of even order are

$$\mu_{2k} = 2K_{k+1/2}(2), \quad k = 0, 1, 2, \ldots, \tag{16}$$

and can be computed recursively by

$$\begin{aligned} \mu_{2k+2} &= (k+\tfrac{1}{2})\mu_{2k} + \mu_{2k-2}, \quad k = 0, 1, 2, \ldots, \\ \mu_{-2} &= \mu_0 = 2K_{1/2}(2). \end{aligned} \tag{17}$$

This is done in the routine `sr_expminfpinf.m`:

```
% SR_EXPMINFPINF Symbolic/variable-precision recurrence
%    coefficients for the weight function
%    w(x)=exp(-1/x^2-x^2) on [-inf,inf]
%
syms mom ab
digits(d); dig=d;
mom(2:2:2*N)=vpa(0,dig);
mom(1)=vpa('2*BesselK(1/2,2)',dig);
mom(3)=3*mom(1)/2;
for k=3:2*N
   mom(2*k-1)=(k-3/2)*mom(2*k-3)+mom(2*k-5);
end
ab=schebyshev(dig,N,mom);
```

Ill-conditioning of the moment map here is considerably less severe than in the case of the weight function $\exp(-1/t - t)$, a phenomenon similar to one observed for Laguerre vs Hermite weight functions (see [2, Tables 2.2 and 2.3]). Comparing the results of `sr_expminfpinf.m` for `d` = 50 with those for `d` = 55 reveals a loss of at most 17 digits as compared to 36 digits in the case of `sr_exp0inf.m` (cf. §3.2). The choices `d` = 55 and `d` = 60 produce results that agree to at least 38 digits. They are stored in the file `abexpminfpinf` to 34 digits and, rounded to 16 digits, are partially displayed below.

```
ab =
[0, .2398755439361229]
[0, 1.500000000000000]
[0, .6666666666666667]
[0, 2.583333333333333]
[0, 1.475806451612903]
[0, 3.659439450026441]
 ...       ...
[0, 16.49963635631090]
[0, 20.31604933808658]
[0, 17.46711169129393]
[0, 21.34281282504185]
```

Example 3.3. Compute $\int_{-\infty}^{\infty} e^{-t} \cos t \, e^{-1/t^2 - t^2} \mathrm{d}t$.

Table 3.3 illustrates the use of Gauss quadrature relative to the weight function $w(t) = \exp(-1/t^2 - t^2)\mathrm{d}t$.

```
>> intexpminfpinf

    n      n-point Gauss
    2   1.5040374279876e-01
    4   1.1308193957943e-01
    6   1.1342796227803e-01
    8   1.1342695840745e-01
   10   1.1342695981592e-01
   12   1.1342695981475e-01
   14   1.1342695981475e-01
```

Table 3.3. Results for Example 3.3

In stark contrast, 1000-point Gauss–Hermite quadrature of $e^{-t-1/t^2}\cos t$ yields only 8 correct digits; see the routine `intexpminfpinf.m`.

References

[1] Abramowitz, Milton and Stegun, Irene (eds), *Handbook of mathematical functions with formulas, graphs, and mathematical tables.* National Bureau of Standards, Appl. Math. Ser. 55, U.S. Government Printing Office, Washington, D.C., 1964.

[2] Gautschi, Walter, *Orthogonal polynomials: computation and approximation.* Numerical Mathematics and Scientific Computation, Oxford University Press, Oxford, 2004.

[3] Gradshteyn, I.S. and Ryzhik, I.M., *Table of integrals, series, and products*, 6th ed., Academic Press, San Diego, CA, 2000.