Orthogonal Polynomials, Quadrature, and Approximation: Computational Methods and Software (in Matlab)

WALTER GAUTSCHI

Abstract

Orthogonal polynomials, unless they are classical, require special techniques for their computation. One of the central problems is to generate the coefficients in the basic three-term recurrence relation they are known to satisfy. There are two general approaches for doing this: methods based on moment information, and discretization methods. In the former, one develops algorithms that take as input given moments, or modified moments, of the underlying measure and produce as output the desired recurrence coefficients. In theory, these algorithms yield exact answers. In practice, owing to rounding errors, the results are potentially inaccurate depending on the numerical condition of the mapping from the given moments (or modified moments) to the recurrence coefficients. A study of related condition numbers is therefore of practical interest. In contrast to moment-based algorithms, discretization methods are basically approximate methods: one approximates the underlying inner product by a discrete inner product and takes the recurrence coefficients of the corresponding discrete orthogonal polynomials to approximate those of the desired orthogonal polynomials. Finding discretizations that yield satisfactory rates of convergence requires a certain amount of skill and creativity on the part of the user, although general-purpose discretizations are available if all else fails.

Other interesting problems have as objective the computation of new orthogonal polynomials out of old ones. If the measure of the new

orthogonal polynomials is the measure of the old ones multiplied by a rational function, one talks about modification of orthogonal polynomials and modification algorithms that carry out the transition from the old to the new orthogonal polynomials. This enters into a circle of ideas already investigated by Christoffel in the 1850s, but effective algorithms have been obtained only very recently. They require the computation of Cauchy integrals of orthogonal polynomials — another interesting computational problem.

In the 1960s, a new type of orthogonal polynomials emerged — the so-called Sobolev orthogonal polynomials — which are based on inner products involving derivatives. Although they present their own computational challenges, moment-based algorithms and discretization methods are still two of the main stocks of the trade. The computation of zeros of Sobolev orthogonal polynomials is of particular interest in practice.

An important application of orthogonal polynomials is to quadrature, specifically quadrature rules of the highest algebraic degree of exactness. Foremost among them is the Gaussian quadrature rule and its close relatives, the Gauss–Radau and Gauss–Lobatto rules. More recent extensions are due to Kronrod, who inserts $n+1$ new nodes into a given n-point Gauss formula, again optimally with respect to degree of exactness, and to Turán, who allows derivative terms to appear in the quadrature sum. When integrating functions having poles outside the interval of integration, quadrature rules of polynomial/rational degree of exactness are of interest. Poles inside the interval of integration give rise to Cauchy principal value integrals, which pose computational problems of their own. Interpreting Gaussian quadrature sums in terms of matrices allows interesting applications to the computation of matrix functionals.

In the realm of approximation, orthogonal polynomials, especially discrete ones, find use in curve fitting, e.g. in the least squares approximation of discrete data. This indeed is the problem in which orthogonal polynomials (in substance if not in name) first appeared in the 1850s in work of Chebyshev. Sobolev orthogonal polynomials also had their origin in least squares approximation, when one tries to fit simultaneously functions together with some of their derivatives. Physically motivated are approximations by spline functions that preserve as many moments as possible. Interestingly, these also are related to orthogonal polynomials via Gauss and generalized Gauss-type quadrature formulae. Slowly convergent series whose sum can be expressed

as a definite integral naturally invite the application of Gauss-type quadratures to speed up their convergence. An example are series whose general term is expressible in terms of the Laplace transform or its derivative of a known function. Such series occur prominently in plate contact problems.

Table of Contents

Part III. Approximation

18. Polynomial least squares approximation
    - classical
    - constrained
    - in Sobolev spaces
19. Moment-preserving spline approximation
    - on the positive real line
    - on a compact interval
20. Slowly convergent series
    - generated by a Laplace transform or derivative thereof
    - occurring in plate contact problems

Exercises to Part III

References

- W. Gautschi, Orthogonal polynomials: computation and approximation. Numerical Mathematics and Scientific Computation. Oxford University Press, Oxford, 2004. (This item will be referenced by Ga04.)
- A suite of Matlab routines, called `OPQ`, to be found at the web site `http://www.cs.purdue.edu/archives/2002/wxg/codes` Each routine can be downloaded individually.
- G. Szegö, Orthogonal polynomials (4th edn), AMS Colloq. Publ. 23, Amer. Math. Soc., Providence, RI, 1975. (This item will be referenced by Sz75.)
- M. Abramowitz and I.A. Stegun (eds), Handbook of mathematical functions, Dover Publ., New York, 1992. (This item will be referenced by AS92.)
- I.S. Gradshteyn and I.M. Ryzhik, Tables of integrals, series, and products (6th edn), Academic Press, San Diego, CA, 2000. (This item will be referenced by GR00.)

PART Ia ORTHOGONAL POLYNOMIALS

1 Recurrence coefficients

1.1 Background and Notations

Orthogonality is defined with respect to an inner product, which in turn involves a measure of integration, $\mathrm{d}\lambda$. An *absolutely continuous* measure has the form

$$\mathrm{d}\lambda(t) = w(t)\mathrm{d}t \quad \text{on } [a,b], \quad -\infty \le a < b \le \infty,$$

where w is referred to as a *weight function*. Usually, w is positive on (a,b), in which case $\mathrm{d}\lambda$ is said to be a *positive measure* and $[a,b]$ is called the *support* of $\mathrm{d}\lambda$. A *discrete measure* has the form

$$\mathrm{d}\lambda_N(t) = \sum_{k=1}^{N} w_k \delta(t - x_k)\mathrm{d}t, \quad x_1 < x_2 < \cdots < x_N,$$

where δ is the Dirac delta function, and usually $w_k > 0$. The support of $\mathrm{d}\lambda_N$ consists of its N *support points* $x_1, x_2, \ldots, x_N$. For absolutely continuous measures, we make the standing assumption that all *moments*

$$\mu_r = \int_{\mathbb{R}} t^r \mathrm{d}\lambda(t), \quad r = 0, 1, 2, \ldots,$$

exist and are finite. The *inner product* of two polynomials p and q relative to the measure $\mathrm{d}\lambda$ is then well defined by

$$(p, q)_{\mathrm{d}\lambda} = \int_{\mathbb{R}} p(t)q(t)\mathrm{d}\lambda(t),$$

and the *norm* of a polynomial p by

$$\|p\|_{\mathrm{d}\lambda} = \sqrt{(p,p)_{\mathrm{d}\lambda}}.$$

Orthogonal polynomials relative to the (positive) measure $\mathrm{d}\lambda$ are defined by

$$\pi_k(\,\cdot\,) = \pi_k(\,\cdot\,; \mathrm{d}\lambda) \text{ a polynomial of exact degree } k, \ k = 0, 1, 2, \ldots,$$

$$(\pi_k, \pi_\ell)_{\mathrm{d}\lambda} \begin{cases} = 0, & k \neq \ell, \\ > 0, & k = \ell. \end{cases}$$

They are uniquely defined up to the leading coefficient, if $\mathrm{d}\lambda$ is absolutely continuous, and are called *monic* if the leading coefficient is equal to 1. For a discrete measure $\mathrm{d}\lambda_N$, there are exactly N orthogonal polynomials $\pi_0, \pi_1, \ldots, \pi_{N-1}$. *Orthonormal polynomials* are defined and denoted by

$$\tilde{\pi}_k(\,\cdot\,; \mathrm{d}\lambda) = \frac{\pi_k(\,\cdot\,; \mathrm{d}\lambda)}{\|\pi_k\|_{\mathrm{d}\lambda}}, \quad k = 0, 1, 2, \ldots .$$

They satisfy

$$(\tilde{\pi}_k, \tilde{\pi}_\ell)_{\mathrm{d}\lambda} = \delta_{k,\ell} = \begin{cases} 0, & k \neq \ell, \\ 1, & k = \ell. \end{cases}$$

Examples of measures resp. weight functions are shown in Tables 1 and 2. The former displays the most important "classical" weight functions, the latter the best-known discrete measures.

Table 1: "Classical" weight functions $\mathrm{d}\lambda(t) = w(t)\mathrm{d}t$

name	$w(t)$	support	comment
Jacobi	$(1-t)^\alpha(1+t)^\beta$	$[-1, 1]$	$\alpha > -1$, $\beta > -1$
Laguerre	$t^\alpha \mathrm{e}^{-t}$	$[0, \infty]$	$\alpha > -1$
Hermite	$\lvert t\rvert^{2\alpha}\mathrm{e}^{-t^2}$	$[-\infty, \infty]$	$\alpha > -\frac{1}{2}$
Meixner-Pollaczek	$\frac{1}{2\pi}\,\mathrm{e}^{(2\phi-\pi)t}\lvert\Gamma(\lambda+it)\rvert^2$	$[-\infty, \infty]$	$\lambda > 0$, $0 < \phi < \pi$

1.2 Three-term recurrence relation

For any n ($< N-1$ if $\mathrm{d}\lambda = \mathrm{d}\lambda_N$), the first $n+1$ monic orthogonal polynomials satisfy a three-term recurrence relation

$$\begin{gathered} \pi_{k+1}(t) = (t - \alpha_k)\pi_k(t) - \beta_k \pi_{k-1}(t), \quad k = 0, 1, \ldots, n-1, \\ \pi_{-1}(t) = 0, \quad \pi_0(t) = 1, \end{gathered} \tag{1.1}$$

Table 2: "Classical" discrete measures $\mathrm{d}\lambda(t) = \sum_{k=0}^{M} w_k \delta(t-k)\mathrm{d}t$

name	M	w_k	comment
discrete Chebyshev	$N-1$	1	
Krawtchouk	N	$\binom{N}{k} p^k (1-p)^{N-k}$	$0 < p < 1$
Charlier	∞	$\mathrm{e}^{-a} a^k / k!$	a¿0
Meixner	∞	$\frac{c^k}{\Gamma(\beta)} \frac{\Gamma(k+\beta)}{k!}$	$0 < c < 1, \beta > 0$
Hahn	N	$\binom{\alpha+k}{k}\binom{\beta+N-k}{N-k}$	$\alpha > -1, \beta > -1$

where the *recurrence coefficients* $\alpha_k = \alpha_k(\mathrm{d}\lambda)$, $\beta_k = \beta_k(\mathrm{d}\lambda)$ are real and positive, respectively. The coefficient β_0 in (1.1) multiplies $\pi_{-1} = 0$, and hence can be arbitrary. For later use, it is convenient to define

$$\beta_0 = \beta_0(\mathrm{d}\lambda) = \int_{\mathbb{R}} \mathrm{d}\lambda(t). \tag{1.2}$$

The proof of (1.1) is rather simple if one expands $\pi_{k+1}(t) - t\pi_k(t) \in \mathbb{P}_k$ in orthogonal polynomials $\pi_0, \pi_1, \ldots, \pi_k$ and observes orthogonality and the obvious, but important, property $(tp, q)_{\mathrm{d}\lambda} = (p, tq)_{\mathrm{d}\lambda}$ of the inner product. As a by-product of the proof, one finds the formulae of Darboux,

$$\begin{aligned} \alpha_k(\mathrm{d}\lambda) &= \frac{(t\pi_k, \pi_k)_{\mathrm{d}\lambda}}{(\pi_k, \pi_k)_{\mathrm{d}\lambda}}, \quad k = 0, 1, 2, \ldots, \\ \beta_k(\mathrm{d}\lambda) &= \frac{(\pi_k, \pi_k)_{\mathrm{d}\lambda}}{(\pi_{k-1}, \pi_{k-1})_{\mathrm{d}\lambda}}, \quad k = 1, 2, \ldots . \end{aligned} \tag{1.3}$$

The second yields

$$\|\pi_k\|^2_{\mathrm{d}\lambda} = \beta_0 \beta_1 \cdots \beta_k. \tag{1.4}$$

Placing the coefficients α_k on the diagonal, and $\sqrt{\beta_k}$ on the two side diagonals

of a matrix produces what is called the *Jacobi matrix* of the measure $\mathrm{d}\lambda$,

$$\boldsymbol{J}(\mathrm{d}\lambda) = \begin{bmatrix} \alpha_0 & \sqrt{\beta_1} & & & \mathbf{0} \\ \sqrt{\beta_1} & \alpha_1 & \sqrt{\beta_2} & & \\ & \sqrt{\beta_2} & \alpha_2 & \ddots & \\ & & \ddots & \ddots & \\ \mathbf{0} & & & & \end{bmatrix}. \tag{1.5}$$

It is a real, symmetric, tridiagonal matrix of infinite order, in general. Its principal minor matrix of order n will be denoted by

$$\boldsymbol{J}_n(\mathrm{d}\lambda) = \boldsymbol{J}(\mathrm{d}\lambda)_{[1:n,1:n]}. \tag{1.6}$$

Noting that the three-term recurrence relation for the ortho*normal* polynomials is

$$\begin{gathered} \sqrt{\beta_{k+1}}\tilde{\pi}_{k+1}(t) = (t-\alpha_k)\tilde{\pi}_k(t) - \sqrt{\beta_k}\tilde{\pi}_{k-1}(t), \quad k = 0, 1, 2, \ldots, \\ \tilde{\pi}_{-1}(t) = 0, \quad \tilde{\pi}_0(t) = 1/\sqrt{\beta_0}, \end{gathered} \tag{1.7}$$

or, in matrix form, with $\tilde{\boldsymbol{\pi}}(t) = [\tilde{\pi}_0(t), \tilde{\pi}_1(t), \ldots, \tilde{\pi}_{n-1}(t)]^{\mathrm{T}}$,

$$t\tilde{\boldsymbol{\pi}}(t) = \boldsymbol{J}_n(\mathrm{d}\lambda)\tilde{\boldsymbol{\pi}}(t) + \sqrt{\beta_n}\tilde{\pi}_n(t)\boldsymbol{e}_n, \tag{1.8}$$

one sees that *the zeros* τ_ν *of* $\tilde{\pi}_n(\,\cdot\,;\mathrm{d}\lambda)$ *are precisely the eigenvalues of* $\boldsymbol{J}_n(\mathrm{d}\lambda)$*, and* $\tilde{\boldsymbol{\pi}}(\tau_\nu)$ *corresponding eigenvectors*. This is only one of many reasons why knowledge of the Jacobi matrix, i.e. of the recurrence coefficients, is of great practical interest. For classical measures as the ones in Tables 1 and 2, all recurrence coefficients are explicitly known (cf. Ga04, Tables 1.1 and 1.2). In most other cases, they must be computed numerically.

In the `OPQ` package, routines generating recurrence coefficients have the syntax `ab=r_`*name*`(N)`, where *name* identifies the name of the orthogonal polynomial and `N` is an input parameter specifying the number of α_k and of β_k desired. There may be additional input parameters. The αs and βs are stored in the `N`$\times 2$ array `ab`:

α_0	β_0
α_1	β_1
$\vdots$	$\vdots$
α_{N-1}	β_{N-1}

$N \in \mathbb{N}$.

For example, `ab=r_jacobi(N,a,b)` generates the first `N` recurrence coefficients of the Jacobi polynomial with parameters α=`a`, β=`b`.

Demo#1 The first ten recurrence coefficients for the Jacobi polynomials with parameters $\alpha = -\frac{1}{2}$, $\beta = \frac{3}{2}$.

The Matlab command, followed by the output, is shown in the box below.

```
>> ab=r_jacobi(10,-.5,1.5)
ab =
      6.666666666666666e-01  4.712388980384690e+00
      1.333333333333333e-01  1.388888888888889e-01
      5.714285714285714e-02  2.100000000000000e-01
      3.174603174603174e-02  2.295918367346939e-01
      2.020202020202020e-02  2.376543209876543e-01
      1.398601398601399e-02  2.417355371900826e-01
      1.025641025641026e-02  2.440828402366864e-01
      7.843137254901961e-03  2.455555555555556e-01
      6.191950464396285e-03  2.465397923875433e-01
      5.012531328320802e-03  2.472299168975069e-01
```

2 Modified Chebyshev algorithm

The first $2n$ moments $\mu_0, \mu_1, \ldots, \mu_{2n-1}$ of a measure $\mathrm{d}\lambda$ uniquely determine the first n recurrence coefficients $\alpha_k(\mathrm{d}\lambda)$ and $\beta_k(\mathrm{d}\lambda)$, $k = 0, 1, \ldots, n-1$. However, the corresponding moment map $\mathbb{R}^{2n} \mapsto \mathbb{R}^{2n} : [\mu_k]_{k=0}^{2n-1} \mapsto [\alpha_k, \beta_k]_{k=0}^{n-1}$ is severely ill-conditioned when n is large. Therefore, other moment maps must be sought that are better conditioned. One that has been studied extensively in the literature is based on *modified moments*

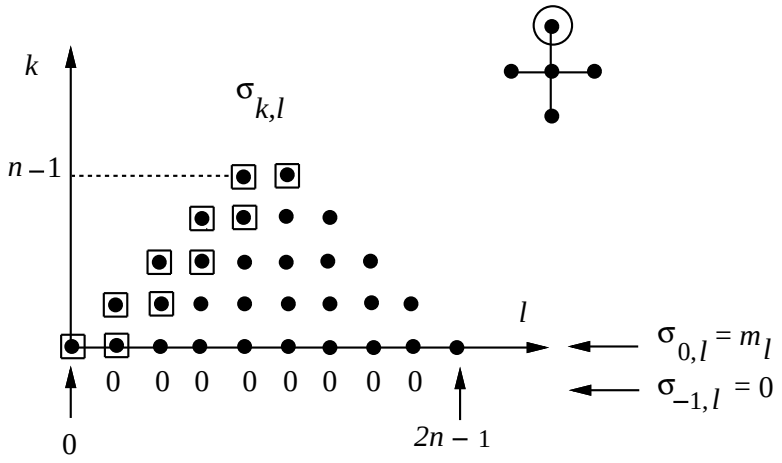
$$m_k = \int_{\mathbb{R}} p_k(t)\mathrm{d}\lambda(t), \quad k = 0, 1, 2, \ldots, \tag{2.1}$$

where $\{p_k\}$, $p_k \in \mathbb{P}_k$, is a given system of polynomials chosen to be close in some sense to the desired polynomials $\{\pi_k\}$. We assume that p_k, like π_k, satisfies a three-term recurrence relation

$$\begin{aligned} p_{k+1}(t) &= (t - a_k)p_k(t) - b_k\pi_{k-1}(t), \quad k = 0, 1, 2, \ldots, \\ p_{-1}(t) &= 0, \quad p_0(t) = 1, \end{aligned} \tag{2.2}$$

but with coefficients $a_k \in \mathbb{R}$, $b_k \geq 0$, that are known. The case $a_k = b_k = 0$ yields powers $p_k(t) = t^k$, hence ordinary moments μ_k, which however, as already mentioned, is not recommended.

Figure 1: Modified Chebyshev algorithm, schematically



The modified moment map

$$\mathbb{R}^{2n} \mapsto \mathbb{R}^{2n} : \quad [m_k]_{k=0}^{2n-1} \mapsto [\alpha_k, \beta_k]_{k=0}^{n-1} \tag{2.3}$$

and related maps have been well studied from the point of view of conditioning (cf. Ga04, §2.1.5 and 2.1.6). The maps are often remarkably well-conditioned, especially for measures supported on a finite interval, but can still be ill-conditioned otherwise.

An algorithm that implements the map (2.3) is the *modified Chebyshev algorithm* (cf. Ga04, §2.1.7), which improves on Chebyshev's original algorithm based on ordinary moments. To describe it, we need the *mixed moments*

$$\sigma_{k\ell} = \int_{\mathbb{R}} \pi_k(t; \mathrm{d}\lambda) p_\ell(t) \mathrm{d}\lambda(t), \quad k, \ell \geq -1, \tag{2.4}$$

which by orthogonality are clearly zero if $\ell < k$.

Algorithm 1 Modified Chebyshev algorithm

initialization:

$$\begin{aligned} &\alpha_0 = a_0 + m_1/m_0, \quad \beta_0 = m_0, \\ &\sigma_{-1,\ell} = 0, \quad \ell = 1, 2, \ldots, 2n-2, \\ &\sigma_{0,\ell} = m_\ell, \quad \ell = 0, 1, \ldots, 2n-1 \end{aligned}$$

continuation (if $n > 1$): for $k = 1, 2, \ldots, n-1$ do

$$\sigma_{k\ell} = \sigma_{k-1,\ell+1} - (\alpha_{k-1} - a_\ell)\sigma_{k-1,\ell} - \beta_{k-1}\sigma_{k-2,\ell}$$

$$+b_\ell \sigma_{k-1,\ell-1}, \quad \ell = k, k+1, \ldots, 2n-k-1,$$

$$\alpha_k = a_k + \frac{\sigma_{k,k+1}}{\sigma_{kk}} - \frac{\sigma_{k-1,k}}{\sigma_{k-1,\ell-1}}, \quad \beta_k = \frac{\sigma_{kk}}{\sigma_{k-1,k-1}}.$$

If $a_k = b_k = 0$, Algorithm 1 reduces to Chebyshev's original algorithm.

Figure 1 depicts the trapezoidal array of the mixed moments and the computing stencil showing that the circled entry is computed in terms of the four entries below. The entries in boxes are those used to compute the αs and βs.

The `OPQ` Matlab command that implements the modified Chebyshev algorithm has the form `ab=chebyshev(N,mom,abm)`, where `mom` is the 1×2`N` array of the modified moments, and `abm` the $(2$`N`$-1)\times 2$ array of the recurrence coefficients a_k, b_k from (2.2) needed in Algorithm 1:

m_0	m_1	m_2	$\cdots$	m_{2N-1}

`mom`

a_0	b_0
a_1	b_1
$\vdots$	$\vdots$
a_{2N-2}	b_{2N-2}

`abm`

If the input parameter `abm` is omitted, the routine assumes $a_k = b_k = 0$ and implements Chebyshev's original algorithm.

Demo#2 "Elliptic" orthogonal polynomials

These are orthogonal relative to the measure

$$\mathrm{d}\lambda(t) = [(1-\omega^2 t^2)(1-t^2)]^{-1/2}\mathrm{d}t \ \text{ on } [-1,1], \quad 0 \le \omega < 1.$$

To apply the modified Chebyshev algorithm, it seems natural to employ Chebyshev moments (i.e. p_k = the monic Chebyshev polynomial of degree k)

$$m_0 = \int_{-1}^{1} \mathrm{d}\lambda(t), \quad m_k = \frac{1}{2^{k-1}} \int_{-1}^{1} T_k(t)\mathrm{d}\lambda(t), \quad k \ge 1.$$

Their computation is not entirely trivial (cf. Ga04, Example 2.29), but a stable algorithm is available as `OPQ` routine `mm_ell.m`, which for given `N` generates the first 2`N` modified moments of $\mathrm{d}\lambda$ with ω^2 being input via the parameter `om2`. The complete Matlab routine is as follows:

```
function ab=r_elliptic(N,om2)
   abm=r_jacobi(2*N-1,-1/2);
   mom=mm_ell(N,om2);
   ab=chebyshev(N,mom,abm)
```

For `om2`=.999 and `N`=40, results produced by the routine are partially shown in the box below.

```
ab =
      0  9.682265121100620e+00
      0  7.937821421385184e-01
      0  1.198676724605757e-01
      0  2.270401183698990e-01
      0  2.410608787266061e-01
      0  2.454285325203698e-01
      0  2.473016530297635e-01
      0  2.482587060199245e-01
      :  :
      0  2.499915376529289e-01
      0  2.499924312667191e-01
      0  2.499932210069769e-01
```

Clearly, $\beta_k \to \frac{1}{4}$ as $k \to \infty$, which is consistent with the fact that $\mathrm{d}\lambda$ belongs to the Szegö class (cf. Ga04, p. 12). Convergence, in fact, is monotone for $k \geq 2$.

3 Discrete Stieltjes and Lanczos algorithm

Computing the recurrence coefficients of a discrete measure is a prerequisite for discretization methods to be discussed in the next section. Given the measure

$$\mathrm{d}\lambda_N(t) = \sum_{k=1}^{N} w_k \delta(t - x_k)\mathrm{d}t, \tag{3.1}$$

the problem is to compute $\alpha_{\nu,N} = \alpha_\nu(\mathrm{d}\lambda_N)$, $\beta_{\nu,N} = \beta_\nu(\mathrm{d}\lambda_N)$ for all $\nu \leq n-1$, $n \leq N$, which will provide access to the discrete orthogonal polynomials of

degrees up to n, or else, to determine the Jacobi matrix $\boldsymbol{J}_N(\mathrm{d}\lambda_N)$, which will provide access to all discrete orthogonal polynomials. There are two methods in use, a discrete Stieltjes procedure and the Lanczos algorithm.

3.1 Discrete Stieltjes procedure

Since the inner product for the measure (3.1) is a finite sum,

$$(p,q)_{\mathrm{d}\lambda_N} = \sum_{k=1}^{N} w_k p(x_k) q(x_k), \tag{3.2}$$

Darboux's formulae (1.3) seem to offer attractive means of computing the desired recurrence coefficients, since all inner products appearing in these formulae are finite sums. The only problem is that we do not yet know the orthogonal polynomials $\pi_k = \pi_{k,N}$ involved. For this, however, we can make use of an idea already expressed by Stieltjes in 1884: combine Darboux's formulae with the basic three-term recurrence relation. Indeed, when $k = 0$ we know that $\pi_{0,N} = 1$, so that Darboux's formula for $\alpha_0(\mathrm{d}\lambda_N)$ can be applied, and $\beta_0(\mathrm{d}\lambda_N)$ is simply the sum of the weights w_k. Now that we know $\alpha_0(\mathrm{d}\lambda_N)$, we can apply the recurrence relation (1.1) for $k = 0$ to compute $\pi_{1,N}(t)$ for $t = x_k$, $k = 1, 2, \ldots, N$. We then have all the information at hand to reapply Darboux's formulae for $\alpha_{1,N}$ and $\beta_{1,N}$, which in turn allows us to compute $\pi_{2,N}(t)$ for all $t = x_k$ from (1.1). In this manner we proceed until all $\alpha_{\nu,N}$, $\beta_{\nu,N}$, $\nu \leq n-1$, are determined. If $n = N$, this will yield the Jacobi matrix $\boldsymbol{J}_N(\mathrm{d}\lambda_N)$.

The procedure is quite effective, at least when $n \ll N$. As n approaches N, instabilities may develop, particularly if the support points x_k of $\mathrm{d}\lambda_N$ are equally, or nearly equally, spaced.

The `OPQ` routine implementing Stieltjes's procedure is called by `ab=stieltjes(n,xw)`, where `n` $\leq N$, and `xw` is an $N \times 2$ array containing the support points and weights of the inner product,

x_1	w_1
x_2	w_2
$\vdots$	$\vdots$
x_N	w_N

`xw`

As usual, the recurrence coefficients $\alpha_{\nu,N}$, $\beta_{\nu,N}$, $0 \le \nu \le n-1$, are stored in the $\mathtt{n}\times 2$ array `ab`.

3.2 Lanczos's algorithm

Lanczos's algorithm is a general procedure to orthogonally tridiagonalize a given symmetric matrix $\boldsymbol{A}$. Thus, it finds an orthogonal matrix $\boldsymbol{Q}$ and a symmetric tridiagonal matrix $\boldsymbol{T}$ such that $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{Q} = \boldsymbol{T}$. Both $\boldsymbol{Q}$ and $\boldsymbol{T}$ are uniquely determined by the first column of $\boldsymbol{Q}$.

Given the measure (3.1), it is known that an orthogonal matrix $\boldsymbol{Q} \in \mathbb{R}^{(N+1)\times(N+1)}$ exists, with the first column being $\boldsymbol{e}_1 = [1, 0, \ldots, 0]^{\mathrm{T}} \in \mathbb{R}^{N+1}$, such that (see Ga04, Corollary to Theorem 3.1)

(3.3)
$$\boldsymbol{Q}^{\mathrm{T}} \begin{bmatrix} 1 & \sqrt{w_1} & \sqrt{w_2} & \cdots & \sqrt{w_N} \\ \sqrt{w_1} & x_1 & 0 & \cdots & 0 \\ \sqrt{w_2} & 0 & x_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sqrt{w_N} & 0 & 0 & \cdots & x_N \end{bmatrix} \boldsymbol{Q} = \begin{bmatrix} 1 & \sqrt{\beta_0} & 0 & \cdots & 0 \\ \sqrt{\beta_0} & \alpha_0 & \sqrt{\beta_1} & \cdots & 0 \\ 0 & \sqrt{\beta_1} & \alpha_1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \alpha_{N-1} \end{bmatrix},$$

where $\alpha_k = \alpha_{k,N}$, $\beta_k = \beta_{k,N}$. We are thus in the situation described above, where $\boldsymbol{A}$ is the matrix displayed on the left and $\boldsymbol{T}$ the matrix on the right, the desired Jacobi matrix $\boldsymbol{J}_N(\mathrm{d}\lambda_N)$ bordered by a first column and a first row containing β_0. The computation can be arranged so that only the leading principal minor matrix of order $n+1$ is obtained.

Lanczos's algorithm in its original form (published in 1950) is numerically unstable, but can be stabilized using ideas of Rutishauser (1963). An algorithm and pseudocode, using a sequence of Givens rotations to construct the matrix $\boldsymbol{Q}$ in (3.3), forms the basis for the `OPQ` Matlab code `ab=lanczos(n,xw)`, where the input and output parameters have the same meaning as in the routine `stieltjes.m`.

This routine enjoys good stability properties but may be considerably slower than Stieltjes's procedure.

4 Discretization methods

The basic idea is to discretize the given measure $\mathrm{d}\lambda$, i.e. approximate it by a discrete measure

$$\mathrm{d}\lambda(t) \approx \mathrm{d}\lambda_N(t), \tag{4.1}$$

and then use the recurrence coefficients $\alpha_k(\mathrm{d}\lambda_N)$, $\beta_k(\mathrm{d}\lambda_N)$ of the discrete measure to approximate $\alpha_k(\mathrm{d}\lambda)$, $\beta_k(\mathrm{d}\lambda)$. The former are computed by either Stieltjes's procedure or the Lanczos algorithm. The effectiveness of the method is crucially tied to the quality of the discretization. We illustrate this by a simple, yet interesting, example.

Example 1. Chebyshev weight function plus a constant,

$$w(t) = (1-t^2)^{-1/2} + c \ \text{ on } [-1,1], \quad c > 0.$$

It suffices to approximate the inner product for the weight function w. This can always be done by using appropriate quadrature formulae. In the case at hand, it is natural to treat the two parts of the weight function separately, indeed to use Gauss–Chebyshev quadrature for the first part and Gauss–Legendre quadrature for the second,

$$\begin{aligned}(p,q)_w &= \int_{-1}^{1} p(t)q(t)(1-t^2)^{-1/2}\mathrm{d}t + c\int_{-1}^{1} p(t)q(t)\mathrm{d}t \\ &\approx \sum_{k=1}^{M} w_k^{\mathrm{Ch}} p(x_k^{\mathrm{Ch}})q(x_k^{\mathrm{Ch}}) + c\sum_{k=1}^{M} w_k^{\mathrm{L}} p(x_k^{\mathrm{L}})q(x_k^{\mathrm{L}}).\end{aligned} \tag{4.2}$$

Here, x_k^{Ch}, w_k^{Ch} are the nodes and weights of the M-point Gauss–Chebyshev quadrature rule, and x_k^{L}, w_k^{L} those of the Gauss–Legendre quadrature rule. The discrete measure implied by (4.2) is $\mathrm{d}\lambda_N$ with $N = 2M$ and

$$\mathrm{d}\lambda_N(t) = \sum_{k=1}^{M} w_k^{\mathrm{Ch}}\delta(t - x_k^{\mathrm{Ch}}) + c\sum_{k=1}^{M} w_k^{\mathrm{L}}\delta(t - x_k^{\mathrm{L}}). \tag{4.3}$$

What is attractive about this choice is the fact that the approximation in (4.2) is actually an equality whenever the product $p \cdot q$ is a polynomial of degree $\leq 2M-1$. Now if we are interested in computing $\alpha_k(w)$, $\beta_k(w)$ for $k \leq n-1$, then the products $p \cdot q$ that occur in Darboux's formulae are all of

degree $\leq 2n-1$. Therefore, we have equality in (4.2) if $n \leq M$. It therefore suffices to take $M = n$ in (4.2) to obtain the first n recurrence coefficients exactly.

In general, the quadrature rules will not produce exact results, and N will have to be increased through a sequence of integers until convergence occurs.

Example 1 illustrates the case of a 2-component discretization. In a general *multiple-component discretization*, the support $[a, b]$ of $\mathrm{d}\lambda$ is decomposed into s intervals,

$$[a, b] = \bigcup_{j=1}^{s} [a_j, b_j], \tag{4.4}$$

where the intervals $[a_j, b_j]$ may or may not be disjoint. The measure $\mathrm{d}\lambda$ is then discretized on each interval $[a_j, b_j]$ using either a tailor-made quadrature (as in Example 1), or a general-purpose quadrature. For the latter, a Fejér quadrature rule on $[-1, 1]$, suitably transformed to $[a_j, b_j]$, has been found useful. (The Fejér rule is the interpolatory quadrature formula based on Chebyshev points.) If the original measure $\mathrm{d}\lambda$ has also a discrete component, this component is simply added on. Rather than go into details (which are discussed in Ga04, §2.2.4), we present the Matlab implementation, another illustrative example, and a demo.

The `OPQ` routine for the multiple-component discretization is `ab=mcdis (n,eps0,quad,Nmax)`, where in addition to the variables `ab` and `n`, which have the usual meaning, there are three other parameters, `eps0`: a prescribed accuracy tolerance, `quad`: the name of a quadrature routine carrying out the discretization on each subinterval if tailor-made (otherwise, `quadgp.m`, a general-purpose quadrature routine can be used), `Nmax`: a maximal allowable value for the discretization parameter N. The decomposition (4.4) is input via the `mc`$\times 2$ array

$$\mathtt{AB}= \left\| \begin{array}{c|c} a_1 & b_1 \\ a_2 & b_2 \\ \vdots & \vdots \\ a_{\mathrm{mc}} & b_{\mathrm{mc}} \end{array} \right\|,$$

where `mc` is the number of components (the s in (4.4)). A discrete component which may possibly be present in $\mathrm{d}\lambda$ is input via the array

$$\texttt{DM}=\left\|\begin{array}{c|c} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_{\rm mp} & y_{\rm mp} \end{array}\right\|,$$

with the first column containing the support points, and the second column the associated weights. The number of support points is `mp`. Both `mc` and `mp`, as well as `AB` and `DM`, are global variables. Another global variable is `iq`, which has to be set equal to 1 if the user provides his or her own quadrature routine, and equal to 0 otherwise.

Example 2. The normalized Jacobi weight function plus a discrete measure.

This is the measure

$$\mathrm{d}\lambda(t) = (\beta_0^{\rm J})^{-1}(1-t)^\alpha(1+t)^\beta \mathrm{d}t + \sum_{j=1}^{p} w_j \delta(t-x_j)\mathrm{d}t \text{ on } [-1,1],$$

where

$$\beta_0^{\rm J} = \int_{-1}^{1}(1-t)^\alpha(1+t)^\beta \mathrm{d}t, \quad \alpha > -1,\ \beta > -1.$$

Here, one single component suffices to do the discretization, and the obvious choice of quadrature rule is the Gauss–Jacobi N-point quadrature formula to which the discrete component is added on. Similarly as in Example 1, taking $N = n$ yields the first n recurrence coefficients $\alpha_k(\mathrm{d}\lambda)$, $\beta_k(\mathrm{d}\lambda)$, $k \le n-1$, exactly. The global parameters in Matlab are here `mc`=1, `mp`=p, `iq`=1, and

$$\texttt{AB}=\left\|\begin{array}{c|c} -1 & 1 \end{array}\right\| \qquad \texttt{DM}=\left\|\begin{array}{c|c} x_1 & w_1 \\ x_2 & w_2 \\ \vdots & \vdots \\ x_p & w_p \end{array}\right\|$$

Demo#3 Logistic density function,

$$\mathrm{d}\lambda(t) = \frac{\mathrm{e}^{-t}}{(1+\mathrm{e}^{-t})^2}\,\mathrm{d}t, \quad t \in \mathbb{R}.$$

The discretization is conveniently effected by the quadrature rule

$$\begin{aligned}\int_{\mathbb{R}} p(t)\mathrm{d}\lambda(t) &= \int_0^\infty \frac{p(-t)}{(1+\mathrm{e}^{-t})^2}\,\mathrm{e}^{-t}\mathrm{d}t + \int_0^\infty \frac{p(t)}{(1+\mathrm{e}^{-t})^2}\,\mathrm{e}^{-t}\mathrm{d}t \\ &\approx \sum_{\nu=1}^{N} \lambda_\nu^{\rm L}\,\frac{p(-\tau_\nu^{\rm L})+p(\tau_\nu^{\rm L})}{(1+\mathrm{e}^{-\tau_\nu^{\rm L}})^2},\end{aligned}$$

where τ_k^{L}, λ_k^{L} are the nodes and weights of the N-point Gauss–Laguerre quadrature formula. This no longer produces exact results for $N = n$, but converges rapidly as $N \to \infty$. The exact answers happen to be known,

$$\begin{aligned} &\alpha_k(\mathrm{d}\lambda) = 0 \quad \text{by symmetry,} \\ &\beta_0(\mathrm{d}\lambda) = 1, \quad \beta_k(\mathrm{d}\lambda) = \frac{k^4\pi^2}{4k^2-1}, \quad k \ge 1. \end{aligned}$$

Numerical results produced by `mcdis.m` with `N`=40, `eps0`=$10^3\times$`eps`, along with errors (absolute errors for α_k, relative errors for β_k) are shown in the box below. The two entries in the last row are the maximum errors taken over $0 \le n \le 39$.

n	β_n	err α	err β
0	1.0000000000(0)	7.18(–17)	3.33(–16)
1	3.2898681337(0)	1.29(–16)	2.70(–16)
6	8.9447603523(1)	4.52(–16)	1.43(–15)
15	5.5578278399(2)	2.14(–14)	0.00(+00)
39	3.7535340252(3)	6.24(–14)	4.48(–15)
		6.24(–14)	8.75(–15)

5 Cauchy integrals of orthogonal polynomials

5.1 The Jacobi continued fraction

The *Jacobi continued fraction* associated with the measure $\mathrm{d}\lambda$ is

$$\mathcal{J} = \mathcal{J}(t;\mathrm{d}\lambda) = \frac{\beta_0}{t-\alpha_0-}\,\frac{\beta_1}{t-\alpha_1-}\,\frac{\beta_2}{t-\alpha_2-}\,\cdots\,, \tag{5.1}$$

where $\alpha_k = \alpha_k(\mathrm{d}\lambda)$, $\beta_k = \beta_k(\mathrm{d}\lambda)$. From the theory of continued fractions it is readily seen that the nth convergent of $\mathcal{J}$ is

$$\frac{\beta_0}{z-\alpha_0-}\,\frac{\beta_1}{z-\alpha_1-}\cdots\frac{\beta_{n-1}}{z-\alpha_{n-1}} = \frac{\sigma_n(z;\mathrm{d}\lambda)}{\pi_n(z;\mathrm{d}\lambda)}, \quad n = 1,2,3,\ldots, \tag{5.2}$$

where π_n is the monic orthogonal polynomial of degree n, and σ_n a polynomial of degree $n-1$ satisfying the same basic three-term recurrence relation as π_n, but with different starting values,

$$\begin{aligned} &\sigma_{k+1}(z) = (z-\alpha_k)\sigma_k(z) - \beta_k\sigma_{k-1}(z), \quad k = 1,2,3,\ldots, \\ &\sigma_0(z) = 0, \quad \sigma_1(z) = \beta_0. \end{aligned} \tag{5.3}$$

Recall that $\beta_0 = \int_{\mathbb{R}} \mathrm{d}\lambda(t)$. If we define $\sigma_{-1} = -1$, then (5.3) holds also for $k = 0$. We have, moreover,

$$\sigma_n(z) = \int_{\mathbb{R}} \frac{\pi_n(z) - \pi_n(t)}{z - t} \, \mathrm{d}\lambda(t), \quad n = 0, 1, 2, \dots \,, \tag{5.4}$$

as can be seen by showing that the integral on the right also satisfies (5.3). If we define

$$F(z) = F(z; \mathrm{d}\lambda) = \int_{\mathbb{R}} \frac{\mathrm{d}\lambda(t)}{z - t} \tag{5.5}$$

to be the *Cauchy transform* of the measure $\mathrm{d}\lambda$, and more generally consider

$$\rho_n(z) = \rho_n(z; \mathrm{d}\lambda) = \int_{\mathbb{R}} \frac{\pi_n(t)}{z - t} \, \mathrm{d}\lambda(t), \tag{5.6}$$

the *Cauchy integral* of the orthogonal polynomial π_n, we can give (5.4) the form

$$\sigma_n(z) = \pi_n(z) F(z) - \rho_n(z), \tag{5.7}$$

and hence

$$\frac{\sigma_n(z)}{\pi_n(z)} = F(z) - \frac{\rho_n(z)}{\pi_n(z)} \,. \tag{5.8}$$

An important result from the theory of the moment problem tells us that, whenever the moment problem for $\mathrm{d}\lambda$ is determined, then

$$\lim_{n \to \infty} \frac{\sigma_n(z)}{\pi_n(z)} = F(z) \quad \text{for } z \in \mathbb{C} \backslash [a, b], \tag{5.9}$$

where $[a, b]$ is the support of the measure $\mathrm{d}\lambda$. If $[a, b]$ is a finite interval, then the moment problem is always determined, and (5.9) is known as *Markov's theorem*.

Note from (5.7) that, since $\sigma_{-1} = -1$, we have

$$\rho_{-1}(z) = 1, \tag{5.10}$$

and the sequence $\{\rho_n\}_{n=-1}^{\infty}$ satisfies the same three-term recurrence relation as $\{\pi_n\}_{n=-1}^{\infty}$. As a consequence of (5.8) and (5.9), however, it behaves quite differently at infinity,

$$\lim_{n \to \infty} \frac{\rho_n(z)}{\pi_n(z)} = 0, \tag{5.11}$$

which implies that $\{\rho_n(z)\}$ is the *minimal solution* of the three-term recurrence relation having the initial value (5.10). It is well known that a minimal solution of a three-term recurrence relation is uniquely determined by its starting value, and, moreover, that

$$\frac{\rho_n(z)}{\rho_{n-1}(z)} = \frac{\beta_n}{z-\alpha_n-}\,\frac{\beta_{n+1}}{z-\alpha_{n+1}-}\,\frac{\beta_{n+2}}{z-\alpha_{n+2}-}\,\cdots\,, \tag{5.12}$$

i.e. the successive ratios of the minimal solution are the successive *tails* of the Jacobi continued fraction (Pincherele's theorem). In particular, by (5.12) for $n = 0$, and (5.2) and (5.9),

$$\rho_0(z) = F(z), \tag{5.13}$$

i.e. ρ_0 is the Cauchy transform of the measure.

We remark that (5.6) is meaningful also for real $z = x$ in (a, b), if the integral is interpreted as a *Cauchy principal value integral* (cf. (15.1))

$$\rho_n(x) = \fint_{\mathbb{R}} \frac{\pi_n(t;\mathrm{d}\lambda)}{x-t}\,\mathrm{d}\lambda(t), \quad x \in (a,b), \tag{5.14}$$

and the sequence $\{\rho_n(x)\}$ satisfies the basic three-term recurrence relation with initial values

$$\rho_{-1}(x) = 1, \quad \rho_0(x) = \fint_{\mathbb{R}} \frac{\mathrm{d}\lambda(t)}{x-t}\,, \tag{5.15}$$

but is no longer minimal.

5.2 Continued fraction algorithm

This is an algorithm for computing the minimal solution $\rho_n(z)$, $z \in \mathbb{C}\backslash[a,b]$, of the basic three-term recurrence relation. Denote the ratio in (5.12) by

$$r_{n-1} = \frac{\rho_n(z)}{\rho_{n-1}(z)}\,. \tag{5.16}$$

Then, clearly,

$$r_{n-1} = \frac{\beta_n}{z-\alpha_n-r_n}\,. \tag{5.17}$$

If, for some $\nu \geq N$, we knew r_ν, we could apply (5.17) for $r = \nu, \nu-1, \ldots, 0$, and then obtain

$$\rho_n(z) = r_{n-1}\rho_{n-1}(z), \quad n = 0, 1, \ldots, N. \tag{5.18}$$

The *continued fraction algorithm* is precisely this algorithm, except that r_ν is replaced by 0. All quantities generated then depend on ν, which is indicated by a superscript.

Algorithm 2 Continued fraction algorithm

backward phase; $\nu \geq N$:

$$r_\nu^{[\nu]} = 0, \quad r_{n-1}^{[\nu]} = \frac{\beta_n}{z - \alpha_n - r_n^{[\nu]}}, \quad n = \nu, \nu-1, \ldots, 0$$

forward phase:

$$\rho_{-1}^{[\nu]}(z) = 1, \quad \rho_n^{[\nu]}(z) = r_{n-1}^{[\nu]}\rho_{n-1}^{[\nu]}(z), \quad n = 0, 1, \ldots, N$$

It can be shown that, as a consequence of the minimality of $\{\rho_n(z)\}$ (cf. Ga04, pp. 114–115),

$$\lim_{\nu\to\infty} \rho_n^{[\nu]}(z) = \rho_n(z), \quad n = 0, 1, \ldots, N, \quad \text{if } z \in \mathbb{C}\backslash[a, b]. \tag{5.19}$$

Convergence is faster the larger $\mathrm{dist}(z, [a, b])$. To compute $\rho_n(z)$, it suffices to apply Algorithm 2 for a sequence of increasing values of ν until convergence is achieved to within the desired accuracy.

The `OPQ` command implementing this algorithm is

```
[rho,r,nu]=cauchy(N,ab,z,eps0,nu0,numax)
```

where the meanings of the output variables `rho`, `r` and input variable `ab` are as shown below.

`rho`	`r`	`ab`
$\rho_0(z)$	$r_0(z)$	α_0 β_0
$\rho_1(z)$	$r_1(z)$	α_1 β_1
$\vdots$	$\vdots$	$\vdots$ $\vdots$
$\rho_N(z)$	$r_N(z)$	α_{numax} β_{numax}

The input variable `eps0` is an error tolerance, the variable `nu0` a suitable starting value of ν in Algorithm 2, which is incremented in steps of, say 5, until the algorithm converges to the accuracy `eps0`. If convergence does not occur within $\nu \leq$`numax`, an error message is issued, otherwise the value of ν yielding convergence is output as `nu`.

6 Modification algorithms

By "modification" of a measure $\mathrm{d}\lambda$, we mean here multiplication of $\mathrm{d}\lambda$ by a rational function r which is positive on the support $[a, b]$ of $\mathrm{d}\lambda$. The *modified measure* thus is

$$\mathrm{d}\hat{\lambda}(t) = r(t)\mathrm{d}\lambda(t), \quad r \text{ rational and } r > 0 \text{ on } [a, b]. \tag{6.1}$$

We are interested in determining the recurrence coefficients $\hat{\alpha}_k$, $\hat{\beta}_k$ for $\mathrm{d}\hat{\lambda}$ in terms of the recurrence coefficients α_k, β_k of $\mathrm{d}\lambda$. An algorithm that carries out the transition from α_k, β_k to $\hat{\alpha}_k$, $\hat{\beta}_k$ is called a *modification algorithm*. While the passage from the orthogonal polynomials relative to $\mathrm{d}\lambda$ to those relative to $\mathrm{d}\hat{\lambda}$ is classical (at least in the case when r is a polynomial), the transition in terms of recurrence coefficients is more recent. It was first treated for linear factors in 1971 by Galant.

Example 3. Linear factor $r(t) = s(t - c)$, $c \in \mathbb{R}\backslash[a, b]$, $s = \pm 1$.

Here, s is a sign factor to make $r(t) > 0$ on (a, b). Galant's approach is to determine the Jacobi matrix of $\mathrm{d}\hat{\lambda}$ from the Jacobi matrix of $\mathrm{d}\lambda$ by means of one step of the symmetric, shifted LR algorithm: by the choice of s, the matrix $s[\boldsymbol{J}_{n+1}(\mathrm{d}\lambda) - c\boldsymbol{I}]$ is symmetric positive definite, hence admits a Choleski decomposition

$$s[\boldsymbol{J}_{n+1}(\mathrm{d}\lambda) - c\boldsymbol{I}] = \boldsymbol{L}\boldsymbol{L}^{\mathrm{T}},$$

where $\boldsymbol{L}$ is lower triangular. The Jacobi matrix $\boldsymbol{J}_n(\mathrm{d}\hat{\lambda})$ is now obtained by reversing the order of the product on the right, adding back the shift c, and then discarding the last row and column,[1]

$$\boldsymbol{J}_n(\mathrm{d}\hat{\lambda}) = \left(\boldsymbol{L}^{\mathrm{T}}\boldsymbol{L} + c\boldsymbol{I}\right)_{[1:n,1:n]}.$$

Since the matrices involved are tridiagonal, the procedure can be implemented by simple nonlinear recurrence relations. These can also be obtained more systematically via Christoffel's theorem and its generalizations.

[1] See, e.g. W. Gautschi, "The interplay between classical analysis and (numerical) linear algebra—a tribute to Gene H. Golub", Electron. Trans. Numer. Anal. 13 (2002), 119–147, where it is also shown how a quadratic factor $(t - c_1)(t - c_2)$ can be dealt with by one step of the QR algorithm; see in particular §3.2 and 3.3

6.1 Generalized Christoffel's theorem

We write

$$\mathrm{d}\hat{\lambda}(t) = \frac{u(t)}{v(t)}\,\mathrm{d}\lambda(t), \quad u(t) = \pm\prod_{\lambda=1}^{\ell}(t-u_\lambda),\ v(t) = \prod_{\mu=1}^{m}(t-v_\mu), \tag{6.2}$$

where u_λ and v_μ are real numbers outside the support of $\mathrm{d}\lambda$. The sign of $u(t)$ is chosen so that $\mathrm{d}\hat{\lambda}$ is a positive measure. Christoffel's original theorem (1858) relates to the case $v(t) = 1$, i.e. $m = 0$. The generalization to arbitrary v is due to Uvarov (1969). It has a different form depending on whether $m \le n$ or $m > n$. In the first case, it states that

$$u(t)\pi_n(t;\mathrm{d}\hat{\lambda}) = \text{const} \times \begin{vmatrix} \pi_{n-m}(t) & \cdots & \pi_{n-1}(t) & \pi_n(t) & \cdots & \pi_{n+\ell}(t) \\ \pi_{n-m}(u_1) & \cdots & \pi_{n-1}(u_1) & \pi_n(u_1) & \cdots & \pi_{n+\ell}(u_1) \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \pi_{n-m}(u_\ell) & \cdots & \pi_{n-1}(u_\ell) & \pi_n(u_\ell) & \cdots & \pi_{n+\ell}(u_\ell) \\ \rho_{n-m}(v_1) & \cdots & \rho_{n-1}(v_1) & \rho_n(v_1) & \cdots & \rho_{n+\ell}(v_1) \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \rho_{n-m}(v_m) & \cdots & \rho_{n-1}(v_m) & \rho_n(v_m) & \cdots & \rho_{n+\ell}(v_m) \end{vmatrix}, \tag{6.3}$$

where

$$\rho_k(z) = \int_{\mathbb{R}} \frac{\pi_k(t;\mathrm{d}\lambda)}{z-t}\,\mathrm{d}\lambda(t), \quad k = 0, 1, 2, \ldots,$$

are the Cauchy integrals of the orthogonal polynomials π_k. They occur only if $m > 0$. To get monic polynomials, the constant in (6.3) must be taken to be the reciprocal of the (signed) cofactor of the element $\pi_{n+\ell}(t)$.

If $m > n$, the generalized Christoffel theorem has the form

$$u(t)\pi_n(t;\mathrm{d}\hat{\lambda}) = \text{const} \times \begin{vmatrix} 0 & 0 & \cdots & 0 & \pi_0(t) & \cdots & \pi_{n+\ell}(t) \\ 0 & 0 & \cdots & 0 & \pi_0(u_1) & \cdots & \pi_{n+\ell}(u_1) \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 & \pi_0(u_\ell) & \cdots & \pi_{n+\ell}(u_\ell) \\ 1 & v_1 & \cdots & v_1^{m-n-1} & \rho_0(v_1) & \cdots & \rho_{n+\ell}(v_1) \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & v_m & \cdots & v_m^{m-n-1} & \rho_0(v_m) & \cdots & \rho_{n+\ell}(v_m) \end{vmatrix}. \tag{6.4}$$

Both versions of the theorem remain valid for complex u_λ, v_μ if orthogonality is understood in the sense of *formal orthogonality*.

6.2 Linear factors

Generalizing Example 3 to arbitrary complex shifts, we let

$$\mathrm{d}\hat{\lambda}(t) = (t-z)\mathrm{d}\lambda(t), \quad z \in \mathbb{C}\backslash[a,b]. \tag{6.5}$$

Using Christoffel's theorem, letting $\hat{\pi}_n(\,\cdot\,) = \pi_n(\,\cdot\,;\mathrm{d}\hat{\lambda})$, we have

$$(t-z)\hat{\pi}_n(t) = \frac{\begin{vmatrix} \pi_n(t) & \pi_{n+1}(t) \\ \pi_n(z) & \pi_{n+1}(z) \end{vmatrix}}{-\pi_n(z)} = \pi_{n+1}(t) - r_n\pi_n(t), \tag{6.6}$$

where

$$r_n = \frac{\pi_{n+1}(z)}{\pi_n(z)}\,. \tag{6.7}$$

Following Verlinden (1999), we write $(t-z)t\hat{\pi}_n(t)$ in two different ways: in the first, we use the three-term recurrence relation for π_k to obtain

$$\begin{aligned}(t-z)t\hat{\pi}_k(t) &= t\pi_{k+1}(t) - r_k\cdot t\pi_k(t)\\ &= \pi_{k+2}(t) + (\alpha_{k+1}-r_k)\pi_{k+1}(t) + (\beta_{k+1}-r_k\alpha_k)\pi_k(t) - r_k\beta_k\pi_{k-1}(t);\end{aligned}$$

in the second, we use the three-term recurrence relation directly on $\hat{\pi}_k$, and then apply (6.6), to write

$$\begin{aligned}(t-z)t\hat{\pi}_k(t) &= (t-z)[\hat{\pi}_{k+1} + \hat{\alpha}_k\hat{\pi}_k(t) + \hat{\beta}_k\hat{\pi}_{k-1}(t)]\\ &= \pi_{k+2}(t) + (\hat{\alpha}_k - r_{k+1})\pi_{k+1}(t) + (\hat{\beta}_k - r_k\hat{\alpha}_k)\pi_k(t) - r_{k-1}\hat{\beta}_k\pi_{k-1}(t).\end{aligned}$$

Since orthogonal polynomials are linearly independent, the coefficients in the two expressions obtained must be the same. This yields

$$\hat{\alpha}_k - r_{k+1} = \alpha_{k+1} - r_k, \quad r_{k-1}\hat{\beta}_k = r_k\beta_k,$$

hence the following algorithm.

Algorithm 3 Modification by a linear factor $t-z$
initialization:

$$\begin{aligned}r_0 &= z-\alpha_0, \quad r_1 = z-\alpha_1-\beta_1/r_0,\\ \hat{\alpha}_0 &= \alpha_1 + r_1 - r_0, \quad \hat{\beta}_0 = -r_0\,\beta_0.\end{aligned}$$

continuation (if $n > 1$): for $k = 1, 2, \ldots, n-1$ do

$$r_{k+1} = z - \alpha_{k+1} - \beta_{k+1}/r_k,$$
$$\hat{\alpha}_k = \alpha_{k+1} + r_{k+1} - r_k,$$
$$\hat{\beta}_k = \beta_k r_k / r_{k-1}.$$

Note that this requires α_n, β_n in addition to the usual n recurrence coefficients α_k, β_k for $k \le n-1$. Algorithm 3 has been found to be numerically stable.

The `OPQ` Matlab command implementing Algorithm 3 is

```
ab=chri1(N,ab0,z)
```

where `ab0` is an `(N+1)`×2 array containing the recurrence coefficients α_k, β_k, $k = 0, 1, \ldots,$ `N`.

6.3 Quadratic factor

We consider (real) quadratic factors $(t-x)^2 + y^2 = (t-z)(t-\overline{z})$, $z = x + \mathrm{i}y$, $y > 0$. Christoffel's theorem is now applied with $u_1 = z$, $u_2 = \overline{z}$ to express $(t-z)(t-\overline{z})\hat{\pi}_n(t)$ as a linar combination of π_n, π_{n+1}, and π_{n+2},

$$(t-z)(t-\overline{z})\hat{\pi}_n(t) = \pi_{n+2}(t) + s_n \pi_{n+1}(t) + t_n \pi_n(t), \tag{6.8}$$

where

$$s_n = -\left(r'_{n+1} + \frac{r''_{n+1}}{r''_n}\, r'_n \right), \quad t_n = \frac{r''_{n+1}}{r''_n}\, |r_n|^2. \tag{6.9}$$

Here we use the notation

$$r'_n = \operatorname{Re} r_n(z),\ r''_n = \operatorname{Im} r_n(z),\ |r_n|^2 = |r_n(z)|^2, \quad n = 0, 1, 2, \ldots, \tag{6.10}$$

where $r_n(z)$ continues to be the quantity defined in (6.7). The same technique used in §6.2 can be applied to (6.8): express $(t-z)(t-\overline{z})t\hat{\pi}_k(t)$ in two different ways as a linear combination of $\pi_{k+3}, \pi_{k+2}, \ldots, \pi_{k-1}$ and compare the respective coefficients. The result gives rise to

Algorithm 4 Modification by a quadratic factor $(t-z)(t-\overline{z})$, $z = x + \mathrm{i}y$

initialization:

$$r_0 = z - \alpha_0,\ r_1 = z - \alpha_1 - \beta_1/r_0,\ r_2 = z - \alpha_2 - \beta_2/r_1,$$
$$\hat{\alpha}_0 = \alpha_2 + r_2' + \frac{r_2''}{r_1''}\, r_1' - \left(r_1' + \frac{r_1''}{r_0''}\, r_0' \right),$$
$$\hat{\beta}_0 = \beta_0(\beta_1 + |r_0|^2).$$

continuation (if $n > 1$): for $k = 1, 2, \ldots, n-1$ do

$$r_{k+2} = z - \alpha_{k+2} - \beta_{k+2}/r_{k+1},$$
$$\hat{\alpha}_k = \alpha_{k+2} + r_{k+2}' + \frac{r_{k+2}''}{r_{k+1}''}\, r_{k+1}' - \left(r_{k+1}' + \frac{r_{k+1}''}{r_k''}\, r_k' \right),$$
$$\hat{\beta}_k = \beta_k \frac{r_{k+1}'' r_{k-1}''}{[r_k'']^2} \left| \frac{r_k}{r_{k-1}} \right|^2.$$

Note that this requires α_k, β_k for k up to $n+1$. Algorithm 4 is also quite stable, numerically.

The `OPQ` routine for Algorithm 4 is

```
ab=chri2(N,ab0,x,y)
```

with obvious meanings of the variables involved.

Since any real polynomial can be factored into a product of real linear and quadratic factors of the type considered, Algorithms 3 and 4 can be applied repeatedly to deal with modification by an arbitrary polynomial which is positive on the support $[a, b]$.

6.4 Linear divisor

In analogy to (6.5), we consider

$$\mathrm{d}\hat{\lambda}(t) = \frac{\mathrm{d}\lambda(t)}{t - z}, \quad z \in \mathbb{C}\backslash[a, b]. \tag{6.11}$$

Now the *generalized* Christoffel theorem (with $\ell = 0$, $m = 1$) comes into play, giving

$$\hat{\pi}_n(t) = \frac{\begin{vmatrix} \pi_{n-1}(t) & \pi_n(t) \\ \rho_{n-1}(z) & \rho_n(z) \end{vmatrix}}{-\rho_{n-1}(z)} = \pi_n(t) - r_{n-1}\pi_{n-1}(t), \tag{6.12}$$

where now

$$r_n = \frac{\rho_{n+1}(z)}{\rho_n(z)}. \tag{6.13}$$

Similarly as in §6.2 and 6.3, we express $t\hat{\pi}_k(t)$ in two different ways as a linear combination of $\pi_{k+1}, \pi_k, \ldots, \pi_{k-2}$ and compare coefficients. By convention,

$$\hat{\beta}_0 = \int_{\mathbb{R}} \mathrm{d}\hat{\lambda}(t) = \int_{\mathbb{R}} \frac{\mathrm{d}\lambda(t)}{t-z} = -\rho_0(z).$$

The result is:

Algorithm 5 Modification by a linear divisor

initialization:

$$\hat{\alpha}_0 = \alpha_0 + r_0, \quad \hat{\beta}_0 = -\rho_0(z).$$

continuation (if $n > 1$): for $k = 1, 2, \ldots, n-1$ do

$$\hat{\alpha}_k = \alpha_k + r_k - r_{k-1},$$
$$\hat{\beta}_k = \beta_{k-1} r_{k-1} / r_{k-2}.$$

Note that here no coefficient α_k, β_k beyond $k \leq n-1$ is needed, not even β_{n-1}.

The ratios r_k of Cauchy integrals that appear in Algorithm 5 can be precomputed by Algorithm 2, where only the backward phase is relevant, convergence being tested on the $r_k^{[\nu]}$. Once converged, the algorithm also provides $\rho_0(z) = r_{-1}^{[\infty]}$.

As z approaches the support interval $[a, b]$, the strength of minimality of the Cauchy integrals $\{\rho_k(z)\}$ weakens and ceases altogether when $z = x \in [a, b]$. For z very close to $[a, b]$, Algorithm 2 therefore converges very slowly. On the other hand, since minimality is very weak, one can generate ρ_k with impunity, if n is not too large, by forward application of the basic three-term recurrence relation, using the initial values $\rho_{-1}(z) = 1$ and $\rho_0(z)$.

All of this is implemented in the `OPQ` routine

```
[ab,nu]=chri4(N,ab0,z,eps0,nu0,numax,rho0,iopt)
```

where all variables except `rho` and `iopt` have the same meaning as before. The parameter `rho` is $\rho_0(z)$, whereas `iopt` controls the method of computation for r_k: Algorithm 2 if `iopt`=1, and forward recursion otherwise.

6.5 Quadratic divisor

We now consider

$$\text{(6.14)}\quad \mathrm{d}\hat{\lambda}(t) = \frac{\mathrm{d}\lambda(t)}{(t-z)(t-\overline{z})} = \frac{\mathrm{d}\lambda(t)}{(t-x)^2+y^2}\,, \quad z = x+\mathrm{i}y,\ x\in\mathbb{R},\ y>0.$$

Here we have

$$\text{(6.15)}\qquad \hat{\alpha}_0 = \frac{\displaystyle\int_{\mathbb{R}} t\mathrm{d}\lambda(t)/|t-z|^2}{\displaystyle\int_{\mathbb{R}} \mathrm{d}\lambda(t)/|t-z|^2} = x + y\,\frac{\operatorname{Re}\rho_0(z)}{\operatorname{Im}\rho_0(z)}\,, \quad \hat{\beta}_0 = -\frac{1}{y}\operatorname{Im}\rho_0(z).$$

We are in the case $\ell = 0$, $m = 2$ of the generalized Christoffel theorems (6.3) and (6.4), which give respectively

(6.16)

$$\hat{\pi}_n(t) = \frac{\begin{vmatrix} \pi_{n-2}(t) & \pi_{n-1}(t) & \pi_n(t) \\ \rho_{n-2}(z) & \rho_{n-1}(z) & \rho_n(z) \\ \rho_{n-2}(\overline{z}) & \rho_{n-1}(\overline{z}) & \rho_n(\overline{z}) \end{vmatrix}}{\begin{vmatrix} \rho_{n-2}(z) & \rho_{n-1}(z) \\ \rho_{n-2}(\overline{z}) & \rho_{n-1}(\overline{z}) \end{vmatrix}},\ n\geq 2; \quad \hat{\pi}_1(t) = \frac{\begin{vmatrix} 0 & \pi_0(t) & \pi_1(t) \\ 1 & \rho_0(z) & \rho_1(z) \\ 1 & \rho_0(\overline{z}) & \rho_1(\overline{z}) \end{vmatrix}}{\begin{vmatrix} 1 & \rho_0(z) \\ 1 & \rho_0(\overline{z}) \end{vmatrix}}.$$

This becomes

$$\text{(6.17)}\qquad \hat{\pi}_n(t) = \pi_n(t) + s_n\pi_{n-1}(t) + t_n\pi_{n-2}(t), \quad n\geq 1,$$

where

$$\text{(6.18)}\qquad s_n = -\left(r'_{n-1} + \frac{r''_{n-1}}{r''_{n-2}}\,r'_{n-2}\right),\ n\geq 1; \quad t_n = \frac{r''_{n-1}}{r''_{n-2}}\,|r_{n-2}|^2,\ n\geq 2,$$

with r_n as defined in (6.13) and notations as in (6.10). Exactly the same procedure used to obtain Algorithm 5 yields

Algorithm 6 Modification by a quadratic divisor

initialization:

$$\begin{aligned} &\hat{\alpha}_0 = x + \rho'_0 y/\rho''_0, \quad \hat{\beta}_0 = -\rho''_0/y, \\ &\hat{\alpha}_1 = \alpha_1 - s_2 + s_1, \quad \hat{\beta}_1 = \beta_1 + s_1(\alpha_0 - \hat{\alpha}_1) - t_2, \\ &\hat{\alpha}_2 = \alpha_2 - s_3 + s_2, \quad \hat{\beta}_2 = \beta_2 + s_2(\alpha_1 - \hat{\alpha}_2) - t_3 + t_2. \end{aligned}$$

continuation (if $n > 3$): for $k = 3, 4, \ldots, n-1$ do

$$\hat{\alpha}_k = \alpha_k - s_{k+1} + s_k, \quad \hat{\beta}_k = \beta_{k-2}\, t_k / t_{k-1}.$$

The `OPQ` routine for Algorithm 6 is

```
[ab,nu]=chri5(N,ab0,z,eps0,nu0,numax,rho0,iopt)
```

where the input and output variables have the same meaning as in the routine `chri4.m`.

Just like Algorithms 3 and 4, also Algorithms 5 and 6 can be applied repeatedly to deal with more general polynomial divisors.

PART Ib SOBOLEV ORTHOGONAL POLYNOMIALS

7 Sobolev inner product and recurrence relation

In contrast to the orthogonal polynomials considered so far, the inner product here involves not only function values, but also successive derivative values, all being endowed with their own measures. Thus,

$$(p,q)_S = \int_{\mathbb{R}} p(t)q(t)\mathrm{d}\lambda_0(t) + \int_{\mathbb{R}} p'(t)q'(t)\mathrm{d}\lambda_1(t) + \cdots + \int_{\mathbb{R}} p^{(s)}(t)q^{(s)}(t)\mathrm{d}\lambda_s(t), \quad s \geq 1. \tag{7.1}$$

If all the measures $\mathrm{d}\lambda_\sigma$ are positive, as we assume, the inner product (7.1) has associated with it a sequence of (monic) polynomials $\pi_k(\,\cdot\,;S)$, $k = 0, 1, 2, \ldots$, orthogonal in the sense

$$(\pi_k, \pi_\ell)_S \begin{cases} = 0, & k \neq \ell, \\ > 0, & k = \ell. \end{cases} \tag{7.2}$$

These are called *Sobolev orthogonal polynomials*. We cannot expect them to satisfy a three-term recurrence relation, since the inner product no longer has the shift property $(tp, q) = (p, tq)$. However, like any sequence of monic polynomials of degrees $0, 1, 2, \ldots$, orthogonal or not, they must satisfy an extended recurrence relation of the type

$$\pi_{k+1}(t) = t\pi_k(t) - \sum_{j=0}^{k} \beta_j^k \pi_{k-j}(t), \quad k = 0, 1, 2, \ldots . \tag{7.3}$$

Associated with it is the upper Hessenberg *matrix of recurrence coefficients*

$$\boldsymbol{H}_n = \begin{bmatrix} \beta_0^0 & \beta_1^1 & \beta_2^2 & \cdots & \beta_{n-2}^{n-2} & \beta_{n-1}^{n-1} \\ 1 & \beta_0^1 & \beta_1^2 & \cdots & \beta_{n-3}^{n-2} & \beta_{n-2}^{n-1} \\ 0 & 1 & \beta_0^2 & \cdots & \beta_{n-4}^{n-2} & \beta_{n-3}^{n-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & \beta_0^{n-2} & \beta_1^{n-1} \\ 0 & 0 & 0 & \cdots & 1 & \beta_0^{n-1} \end{bmatrix}. \tag{7.4}$$

In the case $s = 0$ (of ordinary orthogonal polynomials) there holds $\beta_j^k = 0$ for $j > 1$, and the matrix $\boldsymbol{H}_n$ is tridiagonal. If symmetrized by a (real) diagonal similarity transformation, it becomes the Jacobi matrix $\boldsymbol{J}_n(\mathrm{d}\lambda_0)$. When $s > 0$, however, symmetrization of $\boldsymbol{H}_n$ is no longer possible, since $\boldsymbol{H}_n$ may well have complex eigenvalues (see Example 5).

8 Moment-based algorithm

There are now $s+1$ sets of modified moments, one set for each measure $\mathrm{d}\lambda_\sigma$,

$$m_k^{(\sigma)} = \int_{\mathbb{R}} p_k(t)\mathrm{d}\lambda_\sigma, \quad k = 0, 1, 2, \ldots; \quad \sigma = 0, 1, \ldots, s. \tag{8.1}$$

The first $2n$ modified moments of all the sets will uniquely determine the matrix $\boldsymbol{H}_n$ in (7.4), i.e. there is a well-determined map

$$[m_k^{(\sigma)}]_{k=0}^{2n-1},\ \sigma = 0, 1, \ldots, s \ \mapsto \boldsymbol{H}_n, \tag{8.2}$$

called *modified moment map* for Sobolev orthogonal polynomials. In the case where the polynomials p_k in (8.1) satisfy a three-term recurrence relation with known coefficients, and for $s = 1$, an algorithm has been developed that implements the map (8.2). It very much resembles the modified Chebyshev algorithm for ordinary orthogonal polynomials, but is technically much more elaborate.[2] The algorithm, however, is implemented in the `OPQ` routine

```
B=chebyshev_sob(N,mom,abm)
```

which produces the `N`×`N` upper triangular matrix `B` of recurrence coefficients, with β_j^k, $0 \le j \le k$, $0 \le k \le$`N`–1, occupying the position $(j+1, k+1)$ in the matrix. The input parameter `mom` is the $2 \times$ (2`N`) array of modified moments $m_k^{(\sigma)}$, $k = 0, 1, \ldots,$ 2`N`–1; $\sigma = 0, 1$, of the two measures $\mathrm{d}\lambda_0$ and $\mathrm{d}\lambda_1$, and `abm` the (2`N`−1) × 2 array of coefficients a_k, b_k, $k = 0, 1, \ldots,$ 2`N`–2, defining the polynomials p_k.

Example 3. Althammer's polynomials (1962)
These are the Sobolev polynomials relative to the measures $\mathrm{d}\lambda_0(t) = \mathrm{d}t$, $\mathrm{d}\lambda_1(t) = \gamma \mathrm{d}t$ on $[-1, 1]$, $\gamma > 0$.

[2] See W. Gautschi and M. Zhang,"Computing orthogonal polynomials in Sobolev spaces", Numer. Math. 71 (1995), 159–183.

A natural choice of modified moments are the Legendre moments, i.e. $p_k(t)$ is the monic Legendre polynomial of degree k. By orthogonality of the Legendre polynomials, all modified moments $m_k^{(0)}$ and $m_k^{(1)}$ are zero for $k > 0$, while $m_0^{(0)} = 2$ and $m_0^{(1)} = 2\gamma$. The following Matlab routine, therefore, can be used to generate the Althammer polynomials.

```
mom=zeros(2,2*N);
mom(1,1)=2; mom(2,1)=2*g;
abm=r_jacobi(2*N-1);
B=chebyshev_sob(N,mom,abm);
```

9 Discretization algorithm

Taking the inner product of both sides of (7.3) with π_{k-j} gives

$$0 = (\pi_{k+1}, \pi_{k-j})_S = (t\pi_k, \pi_{k-j})_S - \beta_j^k (\pi_{k-j}, \pi_{k-j})_S, \quad j = 0, 1, \ldots, k,$$

hence

$$\beta_j^k = \frac{(t\pi_k, \pi_{k-j})_S}{(\pi_{k-j}, \pi_{k-j})_S}, \quad j = 0, 1, \ldots, k;\ k = 0, 1, \ldots, n-1. \tag{9.1}$$

These are the analogues of Darboux's formulae for ordinary orthogonal polynomials, and like these, can be combined with the recurrence relation (7.3) to successively build up the recurrence coefficients β_j^k in the manner of Stieltjes's procedure. The technical details, of course, are more involved, since we must generate not only the polynomials π_k, but also their derivatives, in order to be able to compute the Sobolev inner products in (9.1). This all is implemented, for arbitrary $s \geq 1$, in the Matlab routine `stieltjes_sob.m`. The basic assumption in the design of this routine is the availability, for each measure $\mathrm{d}\lambda_\sigma$, of an n_σ-point quadrature rule

$$\int_{\mathbb{R}} p(t)\,\mathrm{d}\lambda_\sigma(t) = \sum_{k=1}^{n_\sigma} w_k^{(\sigma)} p(x_k^{(\sigma)}), \quad p \in \mathbb{P}_{2(n-\sigma)-1}, \quad \sigma = 0, 1, \ldots, s, \tag{9.2}$$

that is exact for polynomials p of degree $\leq 2(n-\sigma)-1$. These are typically Gaussian quadrature rules, possibly with discrete components (present in $\mathrm{d}\lambda_\sigma$) added on. The information is supplied to the routine via the $1 \times (s+1)$ array

$$\texttt{nd}= [n_0, n_1, \ldots, n_s]$$

and the $\texttt{md}\times(2s+2)$ array

$$\texttt{xw}= \left[\begin{array}{ccc|ccc} x_1^{(0)} & \cdots & x_1^{(s)} & w_1^{(0)} & \cdots & w_1^{(s)} \\ x_2^{(0)} & \cdots & x_2^{(s)} & w_2^{(0)} & \cdots & w_2^{(s)} \\ \vdots & & \vdots & \vdots & & \vdots \\ x_{\rm md}^{(0)} & \cdots & x_{\rm md}^{(s)} & w_{\rm md}^{(0)} & \cdots & w_{\rm md}^{(s)} \end{array}\right]$$

where `md=max(nd)`. In each column of `xw` the entries after $x_{n_\sigma}^{(\sigma)}$ resp. $w_{n_\sigma}^{(\sigma)}$ (if any) are not used by the routine. Two more input parameters are needed; the first is `a0`, the coefficient $\alpha_0(\mathrm{d}\lambda_0)$, which allows us to initialize the matrix of recurrence coefficients,

$$\beta_0^0 = \frac{(t,1)_S}{(1,1)_S} = \frac{(t,1)_{\mathrm{d}\lambda_0}}{(1,1)_{\mathrm{d}\lambda_0}} = \alpha_0(\mathrm{d}\lambda_0).$$

The other, `same`, is a logical variable set equal to 1 if all quadrature rules have the same set of nodes, and equal to 0 otherwise. The role of this parameter is to switch to a simplified, and thus faster, procedure if `same`=1. A call to the routine, therefore, has the form

```
B=stieltjes_sob(N,s,nd,xw,a0,same)
```

Example 4. Althammer's polynomials, revisited.

Here, the obvious choice of the quadrature rule for $\mathrm{d}\lambda_0$ and $\mathrm{d}\lambda_1$ is the n-point Gauss–Legendre rule. This gives rise to the following routine:

```
s=1; nd=[N N];
a0=0; same=1;
ab=r_jacobi(N);
zw=gauss(N,ab);
xw=[zw(:,1) zw(:,1) ...
     zw(:,2) g*zw(:,2)];
B=stieltjes_sob(N,s,nd,xw,a0,same);
```

The results are identical with those obtained in Example 3.

10 Zeros

If we let $\boldsymbol{\pi}^{\mathrm{T}}(t) = [\pi_0(t), \pi_1(t), \ldots, \pi_{n-1}(t)]$, where π_k are the Sobolev orthogonal polynomials, then the recurrence relation (7.3) can be written in matrix form as

$$t\boldsymbol{\pi}^{\mathrm{T}}(t) = \boldsymbol{\pi}^{\mathrm{T}}(t)\boldsymbol{H}_n + \pi_n(t)\boldsymbol{e}_n^{\mathrm{T}} \tag{10.1}$$

in terms of the matrix $\boldsymbol{H}_n$ in (7.4). This immediately shows that the zeros τ_ν of π_n are the eigenvalues of $\boldsymbol{H}_n$ and $\boldsymbol{\pi}^{\mathrm{T}}(\tau_\nu)$ corresponding left eigenvectors. Naturally, there is no guarantee that the eigenvalues are real; some may well be complex. Also, if n is large, there is a good chance that some of the eigenvalues are ill-conditioned.

The `OPQ` routine for the zeros of π_n is

```
z=sobzeros(n,N,B)
```

where `B` is the `N`×`N` matrix returned by `chebyshev_sob.m` or `stieltjes_sob.m`, and `z` the n-vector of the zeros of π_n, $1 \le n \le N$.

Example 5. Sobolev orthogonal polynomials with only a few real zeros (Meijer, 1994).

The Sobolev inner product in question is

$$(u, v)_S = \int_{-1}^{3} u(t)v(t)\,\mathrm{d}t + \gamma \int_{-1}^{1} u'(t)v'(t)\,\mathrm{d}t + \int_{1}^{3} u'(t)v'(t)\,\mathrm{d}t, \quad \gamma > 0. \tag{10.2}$$

Meijer proved that for $n(\text{even}) \ge 2$ and γ sufficiently large, the polynomial $\pi_n(\,\cdot\,; S)$ has exactly two real zeros, one in $[-3, -1]$ and the other in $[1, 3]$. If $n(\text{odd}) \ge 3$, there is exactly one real zero, located in $[1, 3]$, if γ is sufficiently large. We use the routine `stieltjes_sob.m` and `sobzeros.m` to illustrate this for $n = 6$ and $\gamma = 44,000$. (The critical value of γ above which Meijer's theorem takes hold is about $\gamma = 43,646.2$; see Ga04, Table 2.30.)

The inner product corresponds to the case $s = 1$ and

$$\mathrm{d}\lambda_0(t) = \mathrm{d}t \text{ on } [-1, 3], \quad \mathrm{d}\lambda_1(t) = \begin{cases} \gamma \mathrm{d}t & \text{if } t \in [-1, 1], \\ \mathrm{d}t & \text{if } t \in (1, 3]. \end{cases}$$

Thus, we can write, with suitable transformations of variables,

$$\int_{-1}^{3} p(t)\,\mathrm{d}\lambda_0(t) = 2\int_{-1}^{1} p(2x{+}1)\,\mathrm{d}x, \quad \int_{-1}^{3} p(t)\,\mathrm{d}\lambda_1(t) = \int_{-1}^{1} [\gamma p(x){+}p(x{+}2)]\,\mathrm{d}x$$

and apply n-point Gauss–Legendre quadrature to the integrals on the right. This will produce the matrix $\boldsymbol{H}_n$ exactly. The parameters in the routine `stieltjes_sob.m` have to be chosen as follows:

$$\texttt{nd} = [n, 2n], \quad \texttt{xw} = \begin{bmatrix} 2\tau_1^G + 1 & \tau_1^G & 2\lambda_1^G & \gamma\lambda_1^G \\ \vdots & \vdots & \vdots & \vdots \\ 2\tau_n^G + 1 & \tau_n^G & 2\lambda_n^G & \gamma\lambda_n^G \\ & \tau_1^G + 2 & & \lambda_1^G \\ & \vdots & & \vdots \\ & \tau_n^G + 2 & & \lambda_n^G \end{bmatrix} \in \mathbb{R}^{2n\times 4},$$

where τ_ν^G, λ_ν^G are the nodes and weight of the n-point Gauss–Legendre quadrature rule. Furthermore, `a0`=1 and `same`=1. The complete program, therefore, is as follows:

```
N=6; s=1; a0=1; same=0; g=44000; nd=[N 2*N];
ab=r_jacobi(N); zw=gauss(N,ab);
xw=zeros(2*N,2*(s+1));
xw(1:N,1)=2*zw(:,1)+1; xw(1:N,2)=zw(:,1);
xw(1:N,3)=2*zw(:,2); xw(1:N,4)=g*zw(:,2);
xw(N+1:2*N,2)=zw(:,1)+2; xw(N+1:2*N,4)=zw(:,2);
B=stieltjes_sob(N,s,nd,xw,a0,same);
z=sobzeros(N,N,B)
```

It produces the output

```
   z =

-4.176763898909848e-01 - 1.703657992747233e-01i
-4.176763898909848e-01 + 1.703657992747233e-01i
 8.453761089539369e-01 - 1.538233952529940e-01i
 8.453761089539369e-01 + 1.538233952529940e-01i
-1.070135059563751e+00
 2.598402134930250e+00
```

confirming Meijer's theorem for $n = 6$. A more detailed numerical study, also in the case of odd values of n, has been made in Ga04, Table 2.30.

Exercises to Part I (Stars indicate more advanced exercises.)

1. Explain why, under the assumptions made about the measure $\mathrm{d}\lambda$, the inner product $(p, q)_{\mathrm{d}\lambda}$ of two polynomials p, q is well defined.

2. Show that monic orthogonal polynomials relative to an absolutely continuous measure are uniquely defined. {*Hint*: Use Gram-Schmidt orthogonalization.} Discuss the uniqueness in the case of discrete measures.

3. Supply the details of the proof of (1.1). In particular, derive (1.3) and (1.4).

4. Derive the three-term recurrence relation (1.7) for the orthonormal polynomials.

5. (a) With $\tilde{\pi}_k$ denoting the orthonormal polynomials relative to a measure $\mathrm{d}\lambda$, show that

$$\int_{\mathbb{R}} t\tilde{\pi}_k(t)\tilde{\pi}_\ell(t)\mathrm{d}\lambda(t) = \begin{cases} 0 & \text{if } |k-\ell| > 1, \\ \sqrt{\beta_{k+1}} & \text{if } |k-\ell| = 1, \\ \alpha_k & \text{if } k = \ell, \end{cases}$$

   where $\alpha_k = \alpha_k(\mathrm{d}\lambda)$, $\beta_k = \beta_k(\mathrm{d}\lambda)$.

   (b) Use (a) to prove

$$\boldsymbol{J} = \boldsymbol{J}_n(\mathrm{d}\lambda) = \int_{\mathbb{R}} t\boldsymbol{p}(t)\boldsymbol{p}^{\mathrm{T}}(t)\mathrm{d}\lambda(t),$$

   where $\boldsymbol{p}^{\mathrm{T}}(t) = [\tilde{\pi}_0(t), \tilde{\pi}_1(t), \ldots, \tilde{\pi}_{n-1}(t)]$.

   (c) With notations as in (b), prove

$$t\boldsymbol{p}(t) = \boldsymbol{J}\boldsymbol{p}(t) + \sqrt{\beta_n}\tilde{\pi}_n(t)\boldsymbol{e}_n,$$

   where $\boldsymbol{e}_n = [0, 0, \ldots, 1]^{\mathrm{T}} \in \mathbb{R}^n$.

6. Let $\mathrm{d}\lambda(t) = w(t)\mathrm{d}t$ be symmetric on $[-a, a]$, $a > 0$, that is, $w(-t) = w(t)$ on $[-a, a]$. Show that $\alpha_k(\mathrm{d}\lambda) = 0$, all $k \geq 0$.

7*. Symmetry of orthogonal polynomials.

   Let $\mathrm{d}\lambda(t) = w(t)\mathrm{d}t$ be symmetric in the sense of Exercise 6.

(a) Show that

$$\pi_{2k}(t;\mathrm{d}\lambda) = \pi_k^+(t^2), \quad \pi_{2k+1}(t;\mathrm{d}\lambda) = t\pi_k^-(t^2),$$

where $\pi_k^\pm$ are the monic polynomials orthogonal on $[0, a^2]$ with respect to $\mathrm{d}\lambda^\pm(t) = t^{\mp 1/2} w(t^{1/2})\mathrm{d}t$.

(b) Let (cf. Exercise 6)

$$\pi_{k+1}(t) = t\pi_k(t) - \beta_k \pi_{k-1}(t), \quad k = 0, 1, 2, \ldots,$$
$$\pi_{-1}(t) = 0, \quad \pi_0(t) = 1$$

be the recurrence relation for $\{\pi_k(\,\cdot\,;\mathrm{d}\lambda)\}$, and let $\alpha_k^\pm$, $\beta_k^\pm$ be the recurrence coefficients for $\{\pi_k^\pm\}$. Show that

$$\left.\begin{aligned} \beta_1 &= \alpha_0^+ \\ \beta_{2k} &= \beta_k^+/\beta_{2k-1} \\ \beta_{2k+1} &= \alpha_k^+ - \beta_{2k} \end{aligned}\right\} \; k = 1, 2, 3, \ldots .$$

(c) Derive relations similar to those in (b) which involve α_0^+ and α_k^-, β_k^-.

(d) Write a Matlab program that checks the numerical stability of the nonlinear recursions in (b) and (c) when $\{\pi_k\}$ are the monic Legendre polynomials.

8. The recurrence relation, in Matlab, of the Chebyshev polynomials of the second kind.

(a) Using Matlab, compute $U_k(x)$ for $1 \le k \le N$ either by means of the three-term recurrence relation $U_{n+1}(x) = 2xU_n(x) - U_{n-1}(x)$ for $n = 0, 1, \ldots, N-1$ (where $U_{-1}(x) = 0$, $U_0(x) = 1$), or else by putting $n = 1 : N$ in the explicit formula $U_n(\cos\theta) = \sin(n+1)\theta/\sin\theta$, where $x = \cos\theta$. For selected values of x and N, determine which of the two methods, by timing each, is more efficient.

(b) Using Matlab, compute the single value $U_N(x)$ either by use of the three-term recurrence relation, or by direct computation based on

the trigonometric formula for $U_N(\cos\theta)$. For selected values of x and N, determine which of the two methods, by timing each, is more efficient.

9*. Orthogonality on two separate (symmetric) intervals.

Let $0 < \xi < 1$ and consider orthogonal polynomials π_k relative to the weight function

$$w(t) = \begin{cases} |t|^\gamma (1-t^2)^\alpha (t^2-\xi^2)^\beta, & t \in [-1,\xi] \cup [\xi,1], \\ 0, & \text{otherwise.} \end{cases}$$

Here, $\gamma \in \mathbb{R}$ and $\alpha > -1$, $\beta > -1$. Evidently, w is a symmetric weight function (in the sense of Exercise 6). Define $\pi_k^\pm$ as in Exercise 7(a).

(a) Transform the polynomials $\pi_k^\pm$ orthogonal on $[\xi^2, 1]$ to orthogonal polynomials $\mathring{\pi}_k^\pm$ on the interval $[-1,1]$ and obtain the respective weight function $\mathring{w}^\pm$.

(b) Express β_{2k} and β_{2k+1} in terms of $\mathring{\gamma}_r^\pm$, the leading coefficient of the orthonormal polynomial of degree r relative to the weight function $\mathring{w}^\pm$ on $[-1,1]$. {*Hint*: Use $\beta_r = \|\pi_r\|^2/\|\pi_{r-1}\|^2$ (cf. eqn (1.4)) and relate this to the leading coefficients γ_k, $\gamma_k^\pm$, and $\mathring{\gamma}_k^\pm$, with obvious notations.}

(c) Prove that

$$\lim_{k\to\infty} \beta_{2k} = \frac{1}{4}(1-\xi)^2, \quad \lim_{k\to\infty} \beta_{2k+1} = \frac{1}{4}(1+\xi)^2.$$

{*Hint*: Use the result of (b) in combination with the asymptotic equivalence

$$\mathring{\gamma}_k^\pm \sim 2^k \mathring{\gamma}^\pm, \quad \mathring{\gamma}^\pm = \pi^{-1/2} \exp\left\{ -\frac{1}{2\pi} \int_{-1}^{1} \ln \mathring{w}^\pm(x) (1-x^2)^{-1/2} \mathrm{d}x \right\}, \quad \text{as } k \to \infty$$

(cf. Sz75, eqn (12.7.2)). You may also want to use

$$\int_0^1 \ln(1-a^2x^2)(1-x^2)^{-1/2}\mathrm{d}x = \pi \ln \frac{1+(1-a^2)^{1/2}}{2}, \quad a^2 < 1$$

(see GR00, eqn 4.295.29).}

(d) Prove that

$$\lim_{k\to\infty} \alpha_k^{\pm} = \frac{1+\xi^2}{2}\,, \quad \lim_{k\to\infty} \beta_k^{\pm} = \left(\frac{1-\xi^2}{4}\right)^2 .$$

{*Hint*: Express $\alpha_k^{\pm}$, $\beta_k^{\pm}$ in terms of $\mathring{\alpha}_k^{\pm}$, $\mathring{\beta}_k^{\pm}$, and use the fact that the weight function $\mathring{w}^{\pm}$ is in the Szegö class.}

(e) The recurrence coefficients $\{\beta_k\}$ must satisfy the two nonlinear recursions of Exercise 7(b),(c). Each of them can be interpreted as a pair of fixed-point iterations for the even-indexed and for the odd-indexed subsequence, the fixed points being respectively the limits in (c). Show that, asymptotically, both fixed points are "attractive" for the recursion in 7(b), and "repelling" for the one in 7(c). Also show that in the latter, the fixed points become attractive if they are switched. What are the numerical implications of all this?

(f) Consider the special case $\gamma = \pm 1$ and $\alpha = \beta = -\frac{1}{2}$. In the case $\gamma = 1$, use Matlab to run the nonlinear recursion of Exercise 7(b) and compare the results with the known answers

$$\beta_{2k} = \frac{1}{4}\,(1-\xi)^2\,\frac{1+\eta^{2k-2}}{1+\eta^{2k}}\,, \quad k = 1, 2, 3, \ldots, \quad 0 \le t \le 1$$

and

$$\beta_1 = \frac{1}{2}(1+\xi^2), \quad \beta_{2k+1} = \frac{1}{4}\,(1+\xi)^2\,\frac{1+\eta^{2k+2}}{1+\eta^{2k}}\,, \quad k = 1, 2, 3, \ldots,$$

where $\eta = (1-\xi)/(1+\xi)$ (see Ga04, Example 2.30). Likewise, in the case $\gamma = -1$, run the nonlinear recursion of Exercise 7(c) and compare the results with the exact answers

$$\beta_2 = \tfrac{1}{2}(1-\xi)^2, \quad \beta_{2k} = \tfrac{1}{4}(1-\xi)^2, \quad k = 2, 3, \ldots,$$

and

$$\beta_1 = \xi, \quad \beta_{2k+1} = \tfrac{1}{4}(1+\xi)^2, \quad k = 1, 2, 3, \ldots .$$

Comment on what you observe.

10. Prove the validity of Algorithm 1.

(a) Verify the initialization part.

(b) Combine $\sigma_{k+1,k-1} = 0$ with the three-term recurrence relation for π_k to prove the formula for β_k in the continuation part.

(c) Combine $\sigma_{k+1,k} = 0$ with the three-term recurrence relation for both, π_k and p_k, and use the result of (b), to prove the formula for α_k in the continuation part.

11*. Orthogonal polynomials $\{\pi_k(\,\cdot\,; w)\}$ relative to the weight function ("hat function")

$$w(t) = \begin{cases} 1+t & \text{if } -1 \le t \le 0, \\ 1-t & \text{if } 0 \le t \le 1, \\ 0 & \text{otherwise.} \end{cases}$$

(a) Develop a modified Chebyshev algorithm for generating the first n recurrence coefficients $\beta_k(w)$, $k = 0, 1, \ldots, n-1$ (all $\alpha_k(w) = 0$; why?). Define modified moments with respect to a suitable system of (monic) orthogonal polynomials.

(b) What changes in the routine are required if one wants $\{\pi_k(\,\cdot\,; 1 - w)\}$, or $\{\pi_k(\,\cdot\,; w(1-w))\}$, or $\{\pi_k(\,\cdot\,; w^p)\}$ where $p > -1$?

(c) Download the routine `chebyshev.m`, write a routine `mom.m` for the modified moments to be used in conjunction with `chebyshev.m` to implement (a), and write a Matlab driver to produce results for selected values of n.

(d) Devise a 2-component discretization scheme for computing the first n recurrence coefficients $\beta_k(w)$, $k = 0, 1, 2, \ldots, n-1$, which uses an n-point discretization of the inner product on each component interval and is to yield exact answers (in the absence of rounding errors).

(e) Same as (b).

(f) Download the routine `mcdis`, write a quadrature routine `qhatf` necessary to implement (d), and append a script to the driver of (c) that produces results of the discretization procedure for selected values of n. Download whatever additional routines you need. Run the procedure with `irout` $= 1$ and `irout` $\neq 1$ and observe the respective timings and the maximum discrepancy between the two sets of answers. Verify that the routine "converges" after one

iteration if `idelta` is properly set. Compare the results with those of (a).

(g) Use the routines `acondG.m` and `rcondG.m` to print the absolute and relative condition numbers of the relevant map G_n. Do any of these correlate well with the numerical results obtained in (c)? If not, why not?

12*. Orthogonal polynomials $\{\pi_k(\,\cdot\,; w)\}$ relative to the weight function ("exponential integral")

$$w(t) = E_1(t), \quad E_1(t) = \int_1^\infty \frac{e^{-ts}}{s}\,\mathrm{d}s \ \text{ on } [0,\infty].$$

These are of interest in the theory of radiative transfer; see S. Chandrasekhar, Radiative transfer, Oxford Univ. Press, Oxford, 1950, Chapter II, §23.

(a) Develop and run a multiple-component discretization routine for generating the first n recurrence coefficients $\alpha_k(w)$, $\beta_k(w)$, $k = 0, 1, \ldots, n-1$. Check your results for $n = 20$ against B. Danloy, Math. Comp. 27 (1973), 861–869, Table 3. {*Hint*: Decompose the interval $[0,\infty]$ into two subintervals $[0,2]$ and $[2,\infty]$ (additional subdivisions may be necessary to implement the developments that follow) and incorporate the behavior of $E_1(t)$ near $t = 0$ and $t = \infty$ to come up with appropriate discretizations. For $0 \le t \le 2$, use the power series

$$E_1(t) - \ln(1/t) = -\gamma - \sum_{k=1}^{\infty} \frac{(-1)^k t^k}{k\,k!}\,,$$

where $\gamma = .57721566490153286\ldots$ is Euler's constant, and for $t > 2$ the continued fraction (cf. AS92, eqn 5.1.22)

$$te^t E_1(t) = \frac{1}{1+}\,\frac{a_1}{1+}\,\frac{a_2}{1+}\,\frac{a_3}{1+}\,\frac{a_4}{1+}\cdots, \quad a_k = \lceil k/2 \rceil / t.$$

Evaluate the continued fraction recursively by (cf. W. Gautschi, Math. Comp. 31 (1977), 994–999, §2)

$$\frac{1}{1+}\,\frac{a_1}{1+}\,\frac{a_2}{1+}\cdots = \sum_{k=0}^{\infty} t_k,$$

where

$$t_0 = 1, \quad t_k = \rho_1 \rho_2 \cdots \rho_k, \quad k = 1, 2, 3, \ldots,$$

$$\rho_0 = 0, \quad \rho_k = \frac{-a_k(1 + \rho_{k-1})}{1 + a_k(1 + \rho_{k-1})}, \quad k = 1, 2, 3, \ldots .$$

Download the array `abjaclog(101:200,:)` to obtain the recurrence coefficients `ab` for the logarithmic weight function $\ln(1/t)$.}

(b) Do the same for

$$w(t) = E_2(t), \quad E_2(t) = \int_1^\infty \frac{e^{-ts}}{s^2}\,\mathrm{d}s \ \text{ on } [0, \infty].$$

Check your results against the respective two- and three-point Gauss quadrature formulae in Chandrasekhar, *ibid.*, Table VI.

(c) Do the same for

$$w(t) = E_m(t) \ \text{ on } [0, c], \quad 0 < c < \infty, \ m = 1, 2.$$

Check your results against the respective two-point Gauss quadrature formulae in Chandrasekhar, *ibid.*, Table VII.

13. Let $C = b_0 + \frac{a_1}{b_1+} \frac{a_2}{b_2+} \frac{a_3}{b_3+} \cdots$ be an infinite continued fraction, and $C_n = b_0 + \frac{a_1}{b_1+} \cdots \frac{a_n}{b_n} = \frac{A_n}{B_n}$ its nth convergent. From the theory of continued fractions, it is known that

$$\left.\begin{aligned} A_n &= b_n A_{n-1} + a_n A_{n-2} \\ B_n &= b_n B_{n-1} + a_n B_{n_2} \end{aligned}\right\} \quad n = 1, 2, 3, \ldots,$$

where

$$A_{-1} = 1, \ A_0 = b_0; \quad B_{-1} = 0, \ B_0 = 1.$$

Use this to prove (5.2) and (5.3).

14. Prove (5.4).

15. Show that (5.11) implies $\lim_{n\to\infty} \frac{\rho_n}{y_n} = 0$, where y_n is *any* solution of the three-term recurrence relation (satisfied by ρ_n and π_n) which is linearly independent of ρ_n. Thus, $\{\rho_n\}$ is indeed a minimal solution.

16. Show that the minimal solutions of a three-term recurrence relation form a one-dimensional manifold.

17. (a) Derive (6.9).

    (b) Supply the details for deriving Algorithm 4.

18. Supply the details for deriving Algorithm 5.

19. (a) Prove (6.15).

    (b) Prove (6.17), (6.18).

    (c) Supply the details for deriving Algorithm 6.

20. Show that a Sobolev inner product does not satisfy the shift property $(tp, q) = (p, tq)$.

21. Prove (7.3).

22. The Sobolev inner product (7.1) is called *symmetric* if each measure $\mathrm{d}\lambda_\sigma$ is symmetric in the sense of Problem 6. For symmetric Sobolev inner products,

    (a) show that $\pi_k(-t; S) = (-1)^k \pi_k(t; S)$;

    (b) show that $\beta_{2r}^k = 0$ for $r = 0, 1, \ldots, \lfloor k/2 \rfloor$.

PART II QUADRATURE

11 Gauss-type quadrature formulae

11.1 Gauss formula

Given a positive measure $\mathrm{d}\lambda$, the n-point *Gaussian quadrature formula* associated with the measure $\mathrm{d}\lambda$ is

$$\int_{\mathbb{R}} f(t)\mathrm{d}\lambda(t) = \sum_{\nu=1}^{n} \lambda_\nu^G f(\tau_\nu^G) + R_n^G(f), \tag{11.1}$$

which has maximum algebraic degree of exactness $2n-1$,

$$R_n^G(f) = 0 \quad \text{if } f \in \mathbb{P}_{2n-1}. \tag{11.2}$$

It is well known that the nodes τ_ν^G are the zeros of $\pi_n(\,\cdot\,;\mathrm{d}\lambda)$, and hence the eigenvalues of the Jacobi matrix $\boldsymbol{J}_n(\mathrm{d}\lambda)$; cf. §1.2. Interestingly, the weights λ_ν^G, too, can be expressed in terms of spectral data of $\boldsymbol{J}_n(\mathrm{d}\lambda)$; indeed, they are (Golub and Welsch, 1969)

$$\lambda_\nu^G = \beta_0 \boldsymbol{v}_{\nu,1}^2, \tag{11.3}$$

where $\boldsymbol{v}_{\nu,1}$ is the first component of the normalized eigenvector $\boldsymbol{v}_\nu$ corresponding to the eigenvalue τ_ν^G,

$$\boldsymbol{J}_n(\mathrm{d}\lambda)\boldsymbol{v}_\nu = \tau_\nu^G \boldsymbol{v}_\nu, \quad \boldsymbol{v}_\nu^{\mathrm{T}}\boldsymbol{v}_\nu = 1, \tag{11.4}$$

and, as usual, $\beta_0 = \int_{\mathbb{R}} \mathrm{d}\lambda(t)$. This is implemented in the `OPQ` Matlab routine

```
xw=gauss(N,ab)
```

where `ab`, as in all previous routines, is the `N`×2 array of recurrence coefficients for $\mathrm{d}\lambda$, and `xw` the `N`×2 array containing the nodes τ_ν^G in the first column, and the weights λ_ν^G in the second.

We remark, for later purposes, that the Gauss quadrature sum, for f sufficiently regular, can be expressed in matrix form as

$$\sum_{\nu=1}^{n} \lambda_\nu^G f(\tau_\nu^G) = \beta_0 \boldsymbol{e}_1^{\mathrm{T}} f(\boldsymbol{J}_n(\mathrm{d}\lambda))\boldsymbol{e}_1, \quad \boldsymbol{e}_1 = [1,0,\ldots,0]^{\mathrm{T}}. \tag{11.5}$$

This is an easy consequence of (11.3) and the spectral decomposition of $\boldsymbol{J}_n$,

$$\boldsymbol{J}_n(\mathrm{d}\lambda)\boldsymbol{V} = \boldsymbol{V}\boldsymbol{D}_\tau, \quad \boldsymbol{D}_\tau = \mathrm{diag}(\tau_1^G, \tau_2^G, \ldots, \tau_n^G),$$

where $\boldsymbol{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n]$.

Example 6. Zeros of Sobolev orthogonal polynomials of Gegenbauer type (Groenevelt, 2002).

The polynomials in question are those orthogonal with respect to the Sobolev inner product

$$(u, v)_S = \int_{-1}^{1} u(t)v(t)(1-t^2)^{\alpha-1}\mathrm{d}t + \gamma \int_{-1}^{1} u'(t)v'(t)\frac{(1-t^2)^\alpha}{t^2+y^2}\,\mathrm{d}t.$$

Groenevelt proved that in the case $\gamma \to \infty$ the Sobolev orthogonal polynomials of even degrees $n \geq 4$ have complex zeros if y is sufficiently small. By symmetry, they must in fact be purely imaginary, and by the reality of the Sobolev polynomials, must occur in conjugate complex pairs. As we illustrate this theorem, we have an opportunity to apply not only the routine `gauss.m`, but also a number of other routines, specifically the modification algorithm embodied in the routine `chri6.m`, dealing with the special quadratic divisor $t^2 + y^2$ in the second integral, and the routine `stieltjes_sob.m` generating the recurrence matrix of the Sobelev orthogonal polynomials:

```
s=1; same=0; eps0=1e-14; numax=250; nd=[N N];
ab0=r_jacobi(numax,alpha);
z=complex(0,y);
nu0=nu0jac(N,z,eps0); rho0=0; iopt=1;
ab1=chri6(N,ab0,y,eps0,nu0,numax,rho0,iopt);
zw1=gauss(N,ab1);
ab=r_jacobi(N,alpha-1); zw=gauss(N,ab);
xw=[zw(:,1) zw1(:,1) zw(:,2) gamma*zw1(:,2)];
a0=ab(1,1); B=stieltjes_sob(N,s,nd,xw,a0,same);
z=sobzeros(N,N,B)
```

Demo#4 The case `N`=12, $\alpha = \frac{1}{2}$, and $\gamma = 1$ of Example 6.

Applying the above routine for $y = .1$ and $y = .09$ yields the following zeros (with positive imaginary parts; the other six zeros are the same with opposite signs):

y	zeros	y	zeros
.1	.027543282225	.09	.011086169153 i
	.284410786673		.281480077515
	.541878443180		.540697645595
	.756375307278		.755863108617
	.909868274113		.909697039063
	.989848649239		.989830182743

The numerical results (and additional tests) suggest that Groenevelt's theorem also holds for finite, not necessarily large, values of γ, and, when $\gamma = 1$, that the critical value of y below which there are complex zeros must be betweeen .09 and .1.

11.2 Gauss–Radau formula

If there is an interval $[a, \infty]$, $-\infty < a$, containing the support of $\mathrm{d}\lambda$, it may be desirable to have an $(n+1)$-point quadrature rule of maximum degree of exactness that has a as a prescribed node,

$$\int_{\mathbb{R}} f(t)\mathrm{d}\lambda(t) = \lambda_0^a f(a) + \sum_{\nu=1}^{n} \lambda_\nu^a f(\tau_\nu^a) + R_n^a(f). \tag{11.6}$$

Here, $R_n^a(f) = 0$ for all $f \in \mathbb{P}_{2n}$, and τ_ν^a are the zeros of $\pi_n(\,\cdot\,;\mathrm{d}\lambda_a)$, $\mathrm{d}\lambda_a(t) = (t-a)\mathrm{d}\lambda(t)$. This is called the *Gauss–Radau formula*. There is again a symmetric, tridiagonal matrix, the *Jacobi–Radau matrix*

$$\boldsymbol{J}_{n+1}^{R,a}(\mathrm{d}\lambda) = \begin{bmatrix} \boldsymbol{J}_n(\mathrm{d}\lambda) & \sqrt{\beta_n}\boldsymbol{e}_n \\ \sqrt{\beta_n}\boldsymbol{e}_n^T & \alpha_n^R \end{bmatrix}, \ \alpha_n^R = a - \beta_n \frac{\pi_{n-1}(a)}{\pi_n(a)}, \tag{11.7}$$

where $\boldsymbol{e}_n = [0, 0, \ldots, 1]^{\mathrm{T}} \in \mathbb{R}^n$, $\beta_n = \beta_n(\mathrm{d}\lambda)$, and $\pi_k(\,\cdot\,) = \pi_k(\,\cdot\,;\mathrm{d}\lambda)$, which allows the Gauss–Radau formula to be characterized in terms of eigenvalues and eigenvectors: all nodes of (11.6), including the node a, are the eigenvalues of (11.7), and the weights λ_ν^a expressible as in (11.3) in terms of the corresponding normalized eigenvectors $\boldsymbol{v}_\nu$ of (11.7),

$$\lambda_\nu^a = \beta_0 \boldsymbol{v}_{\nu,1}^2, \quad \nu = 0, 1, 2, \ldots, n. \tag{11.8}$$

As in (11.5), this implies that the Gauss–Radau quadrature sum, for smooth f, can be expressed as $\beta_0 \boldsymbol{e}_1^{\mathrm{T}} f(\boldsymbol{J}_n^{R,a})\boldsymbol{e}_1$.

Naturally, if the support of $\mathrm{d}\lambda$ is contained in an interval $[-\infty, b]$, $b < \infty$, there is a companion formula to (11.6) which has the prescribed node b,

$$\int_{\mathbb{R}} f(t)\,\mathrm{d}\lambda(t) = \sum_{\nu=1}^{n} \lambda_\nu^b f(\tau_\nu^b) + \lambda_{n+1}^b f(\tau_{n+1}^b) + R_n^b(f). \tag{11.9}$$

The eigenvalue/vector characterization also holds for (11.9) if in the formula for α_n^R in (11.7), the variable a, at every occurrence, is replaced by b.

The remainder terms of (11.6) and (11.9), if $f \in C^{2n+1}[a, b]$, have the useful property

$$R_n^a(f) > 0, \quad R_n^b(f) < 0 \quad \text{if } \operatorname{sgn} f^{(2n+1)} = 1 \text{ on } [a, b], \tag{11.10}$$

with the inequalities reversed if $\operatorname{sgn} f^{(2n+1)} = -1$.

For Jacobi resp. generalized Laguerre measures with parameters α, β resp. α, the quantity α_n^R is explicitly known (cf. Ga04, Examples 3.4 and 3.5). For example, if $a = -1$ (in the case of Jacobi measures),

$$\alpha_n^R = -1 + \frac{2n(n+\alpha)}{(2n+\alpha+\beta)(2n+\alpha+\beta+1)}, \quad \alpha_n^R = n, \tag{11.11}$$

whereas for $a = 1$, the sign of α_n^R must be changed and α and β interchanged.

The respective `OPQ` Matlab routines are

```
xw=radau(N,ab,end0)
xw=radau_jacobi(N,iopt,a,b)
xw=radau_laguerre(N,a)
```

In the first, `ab` is the (`N`+1)×2 array of recurrence coefficients for $\mathrm{d}\lambda$, and `end0` either a (for (11.6)) or b (for (11.9)). The last two routines make use of the explicit formulae for α_n^R in the case of Jacobi resp. Laguerre measures, the parameters being α=`a`, β=`b`. The parameter `iopt` chooses between the two Gauss–Radau formulae: the left-handed, if `iopt`=1, the right-handed otherwise.

11.3 Gauss–Lobatto formula

If the support of $\mathrm{d}\lambda$ is contained in the finite interval $[a, b]$, we may wish to prescribe two nodes, the points a and b. Maximizing the degree of exactness

subject to these constraints yields the *Gauss–Lobatto formula*

$$\int_a^b f(t)\mathrm{d}\lambda(t) = \lambda_0^L f(a) + \sum_{\nu=1}^{n} \lambda_\nu^L f(\tau_\nu^L) + \lambda_{n+1}^L f(b) + R_n^{a,b}(f), \tag{11.12}$$

which we write as an $(n+2)$-point formula; we have $R_n^{a,b}(f) = 0$ for $f \in \mathbb{P}_{2n+1}$. The internal nodes τ_ν^L are the zeros of $\pi_n(\,\cdot\,;\mathrm{d}\lambda_{a,b})$, $\mathrm{d}\lambda_{a,b}(t) = (t-a)(b-t)\mathrm{d}\lambda(t)$. All nodes and weights can be expressed in terms of eigenvalues and eigenvectors exactly as in the two preceding subsections, except that the matrix involved is the *Jacobi–Lobatto matrix*

$$\boldsymbol{J}_{n+2}^L(\mathrm{d}\lambda) = \begin{bmatrix} \boldsymbol{J}_{n+1}(\mathrm{d}\lambda) & \sqrt{\beta_{n+1}^L}\boldsymbol{e}_{n+1} \\ \sqrt{\beta_{n+1}^L}\boldsymbol{e}_{n+1}^{\mathrm{T}} & \alpha_{n+1}^L \end{bmatrix}, \tag{11.13}$$

where α_{n+1}^L and β_{n+1}^L are the solution of the 2×2 system of linear equations

$$\begin{bmatrix} \pi_{n+1}(a) & \pi_n(a) \\ \pi_{n+1}(b) & \pi_n(b) \end{bmatrix} \begin{bmatrix} \alpha_{n+1}^L \\ \beta_{n+1}^L \end{bmatrix} = \begin{bmatrix} a\pi_{n+1}(a) \\ b\pi_{n+1}(b) \end{bmatrix}. \tag{11.14}$$

For smooth f, the quadrature sum is expressible as $\beta_0 \boldsymbol{e}_1^{\mathrm{T}} f(\boldsymbol{J}_{n+2}^L)\boldsymbol{e}_1$. For $f \in C^{2n+2}[a,b]$ with constant sign on $[a,b]$, the remainder $R_n^{a,b}(f)$ satisfies

$$R_n^{a,b}(f) < 0 \quad \text{if } \operatorname{sgn} f^{(2n+2)} = 1 \text{ on } [a,b], \tag{11.15}$$

with the inequality reversed if $\operatorname{sgn} f^{(2n+2)} = -1$.

The parameters α_{n+1}^L, β_{n+1}^L for Jacobi measures on $[-1,1]$ with parameters α, β and $a = -b = -1$ are explicitly known (cf. Ga04, Example 3.8),

$$\begin{aligned} \alpha_{n+1}^L &= \frac{\alpha-\beta}{2n+\alpha+\beta+2}, \\ \beta_{n+1}^L &= 4\frac{(n+\alpha+1)(n+\beta+1)(n+\alpha+\beta+1)}{(2n+\alpha+\beta+1)(2n+\alpha+\beta+2)}. \end{aligned} \tag{11.16}$$

The `OPQ` Matlab routines are

```
xw=lobatto(N,ab,endl,endr)
xw=lobatto_jacobi(N,a,b)
```

with the meaning of `ab`, `a`, `b` the same as in the Gauss–Radau routines, and `endl`=a, `endr`=b.

We remark that both Gauss–Radau and Gauss–Lobatto formulae can be generalized to include boundary points of multiplicity $r > 1$. The internal (simple) nodes and weights are still related to orthogonal polynomials, but the boundary weights require new techniques for their computation; see Exercises 8–9.

12 Gauss–Kronrod quadrature

In an attempt to estimate the error of the n-point Gauss quadrature rule, Kronrod in 1965 had the idea of inserting $n+1$ additional nodes and choosing them, along with all $2n+1$ weights, in such a way as to achieve maximum degree of exactness. The resulting quadrature rule can be expected to yield much higher accuracy than the Gauss formula, so that the difference of the two provides an estimate of the error in the Gauss formula. The extended formula thus can be written in the form

$$\int_{\mathbb{R}} f(t)\mathrm{d}\lambda(t) = \sum_{\nu=1}^{n} \lambda_\nu^K f(\tau_\nu^G) + \sum_{\mu=1}^{n+1} \lambda_\mu^{*K} f(\tau_\mu^K) + R_n^{GK}(f), \tag{12.1}$$

and having $3n+2$ free parameters λ_ν^K, λ_μ^{*K}, τ_μ^K at disposal, one ought to be able to achieve degree of exactness $3n+1$,

$$R_n^{GK}(f) = 0 \quad \text{for } f \in \mathbb{P}_{3n+1}. \tag{12.2}$$

A quadrature formula (12.1) that satisfies (12.2) is called a *Gauss–Kronrod formula*. The nodes τ_μ^K, called *Kronrod nodes*, are the zeros of the polynomial π_{n+1}^K of degree $n+1$ which is orthogonal to all polynomials of lower degree in the sense

$$\int_{\mathbb{R}} \pi_{n+1}^K(t) p(t) \pi_n(t; \mathrm{d}\lambda)\, \mathrm{d}\lambda(t) = 0 \quad \text{for all } p \in \mathbb{P}_n. \tag{12.3}$$

Note that the measure of orthogonality here is $\pi_n(t;\mathrm{d}\lambda)\mathrm{d}\lambda(t)$ and thus oscillates on the support of $\mathrm{d}\lambda$. Stieltjes (1894) was the first to consider polynomials π_{n+1}^K of this kind (for $\mathrm{d}\lambda(t) = \mathrm{d}t$); a polynomial π_{n+1}^K satisfying (12.3) is therefore called a *Stieltjes polynomial*. Stieltjes conjectured (in the case $\mathrm{d}\lambda(t) = \mathrm{d}t$) that all zeros of π_{n+1}^K are real and interlace with the n

Gauss nodes— a highly desirable configuration! This has been proved only later by Szegö (1935) not only for Legendre measures, but also for a class of Gegenbauer measures. The study of the reality of the zeros for more general measures is an interesting and ongoing activity.

The computation of Gauss–Kronrod formulae is a challenging problem. A solution has been given only recently by Laurie (1997), at least in the case when a Gauss–Kronrod formula exists with real nodes and positive weights. Interestingly, it can be computed again in terms of eigenvalues and eigenvectors of a symmetric tridiagonal matrix, just like the previous Gauss-type formulae. The relevant matrix, however, is the *Jacobi–Kronrod matrix*

$$\boldsymbol{J}^K_{2n+1}(\mathrm{d}\lambda) = \begin{bmatrix} \boldsymbol{J}_n(\mathrm{d}\lambda) & \sqrt{\beta_n}\boldsymbol{e}_n & \boldsymbol{0} \\ \sqrt{\beta_n}\boldsymbol{e}_n^{\mathrm{T}} & \alpha_n & \sqrt{\beta_{n+1}}\boldsymbol{e}_1^{\mathrm{T}} \\ \boldsymbol{0} & \sqrt{\beta_{n+1}}\boldsymbol{e}_1 & \boldsymbol{J}^*_n \end{bmatrix}. \tag{12.4}$$

Here, $\alpha_n = \alpha_n(\mathrm{d}\lambda)$, $\beta_n = \beta_n(\mathrm{d}\lambda)$, etc, and $\boldsymbol{J}^*_n$ (which is partially known) can be computed by *Laurie's algorithm* (cf. Ga04, §3.1.2.2). Should some of the eigenvalues of (12.4) turn out to be complex, this would be an indication that a Gauss–Kronrod formula (with real nodes) does not exist.

There are two routines in `OPQ`,

```
ab=r_kronrod(N,ab0)
xw=kronrod(n,ab)
```

that serve to compute Gauss–Kronrod formulae. The first generates the Jacobi-Kronrod matrix of order 2N+1, the other the nodes and weights of the quadrature formula, stored respectively in the first and second column of the (2N+1)×2 array `xw`. The recurrence coefficients of the given measure $\mathrm{d}\lambda$ are input via the $\lceil 3\mathtt{N}/2+1\rceil\times 2$ array `ab0`.

13 Gauss–Turán quadrature

The idea of allowing derivatives to appear in a Gauss-type quadrature formula is due to Turán (1950). He considered the case where each node has the same multiplicity $r \geq 1$, that is,

$$\int_{\mathbb{R}} f(t)\,\mathrm{d}\lambda(t) = \sum_{\nu=1}^{n} [\lambda_\nu f(\tau_\nu) + \lambda'_\nu f'(\tau_\nu) + \cdots + \lambda_\nu^{(r-1)} f^{(r-1)}(\tau_\nu)] + R_n(f). \tag{13.1}$$

This is clearly related to Hermite interpolation. Indeed, if all nodes were prescribed and distinct, one could use Hermite interpolation to obtain a formula with degree of exactness $rn-1$ (there are rn free parameters). Turán asked, like Gauss before him, whether one can do better by choosing the nodes τ_ν judiciously. The answer is yes; more precisely, we can get degree of exactness $rn-1+k$, $k>0$, if and only if

$$\int_{\mathbb{R}} \omega_n^r(t)p(t)\,\mathrm{d}\lambda(t) = 0 \quad \text{for all } p \in \mathbb{P}_{k-1}, \tag{13.2}$$

where $\omega_n(t) = \prod_{\nu=1}^n (t-\tau_\nu)$ is the *node polynomial* of (13.1). We have here a new type of orthogonality: the rth power of ω_n, not ω_n, must be orthogonal to all polynomials of degree $k-1$. This is called *power orthogonality*. It is easily seen that r must be odd,

$$r = 2s+1, \quad s \geq 0, \tag{13.3}$$

so that (13.1) becomes

$$\int_{\mathbb{R}} f(t)\,\mathrm{d}\lambda(t) = \sum_{\nu=1}^{n}\sum_{\sigma=0}^{2s} \lambda_\nu^{(\sigma)} f^{(\sigma)}(\tau_\nu) + R_{n,s}(f). \tag{13.4}$$

Then in (13.2), necessarily $k \leq n$, and $k=n$ is optimal. The maximum possible degree of exactness, therefore, is $(2s+2)n-1$, and is achieved if

$$\int_{\mathbb{R}} [\omega_n(t)]^{2s+1} p(t)\,\mathrm{d}\lambda(t) = 0 \quad \text{for all } p \in \mathbb{P}_{n-1}. \tag{13.5}$$

The polynomial $\omega_n = \pi_{n,s}$ satisfying (13.5) is called *s-orthogonal*. It exists uniquely and has distinct simple zeros contained in the support interval of $\mathrm{d}\lambda$. The formula (13.4) is the *Gauss-Turán formula* if its node polynomial ω_n satisfies (13.5) and the weights $\lambda_\nu^{(\sigma)}$ are obtained by Hermite interpolation.

The computation of Gauss-Turán formulae is not as simple as in the case of ordinary Gauss-type formulae. The basic idea, however, is to consider the positive measure $\mathrm{d}\lambda_{n,s}(t) = [\pi_{n,s}(t)]^{2s}\mathrm{d}\lambda(t)$ and to note that $\pi_{n,s}$ is the nth-degree polynomial orthogonal relative to $\mathrm{d}\lambda_{n,s}$. The difficulty is that this defines $\pi_{n,s}$ implicitly, since $\pi_{n,s}$ already occurs in the measure $\mathrm{d}\lambda_{n,s}$. Nevertheless, the difficulty can be surmounted, but at the expense of having to solve a system of nonlinear equations; for details, see Ga04, §3.1.3.2. The procedure is embodied in the `OPQ` routine

```
xw=turan(n,s,eps0,ab0,hom)
```

where the nodes are stored in the first column of the n×(2s+2) array xw, and the successive weights in the remaining 2s+1 columns. The input parameter eps0 is an error tolerance used in the iterative solution of the nonliner system of equations, and the measure $\mathrm{d}\lambda$ is specified by the ((s+1)n)×2 input array ab0 of its recurrence coefficients. Finally, hom=1 or hom≠1 depending on whether or not a certain homotopy in the variable s is used to facilitate convergence of Newton's method for solving the system of nonlinear equations.

14 Quadrature formulae based on rational functions

All quadrature formulae considered so far were based on polynomial degree of exactness. This is meaningful if the integrand is indeed polynomial-like. Not infrequently, however, it happens that the integrand has poles outside the interval of integration. In this case, exactness for appropriate rational functions, in addition to polynomials, is more natural. We discuss this for the simplest type of quadrature rule,

$$\int_{\mathbb{R}} g(t)\mathrm{d}\lambda(t) = \sum_{\nu=1}^{n} \lambda_\nu g(\tau_\nu) + R_n(g). \tag{14.1}$$

The problem, more precisely, is to determine λ_ν, τ_ν such that $R_n(g) = 0$ if $g \in \mathbb{S}_{2n}$, where $\mathbb{S}_{2n}$ is a space of dimension $2n$ consisting of rational functions and polynomials,

$$\begin{aligned} &\mathbb{S}_{2n} = \mathbb{Q}_m \oplus \mathbb{P}_{2n-m-1}, \quad 0 \le m \le 2n, \\ &\mathbb{P}_{2n-m-1} = \text{polynomials of degree } \le 2n-m-1, \\ &\mathbb{Q}_m = \text{rational functions with prescribed poles.} \end{aligned} \tag{14.2}$$

Here, m is an integer of our choosing, and

$$\mathbb{Q}_m = \text{span}\left\{ r(t) = \frac{1}{1+\zeta_\mu t}, \quad \mu = 1, 2, \ldots, m \right\}, \tag{14.3}$$

where

$$\zeta_\mu \in \mathbb{C}, \quad \zeta_\mu \neq 0, \quad 1 + \zeta_\mu t \neq 0 \text{ on } \operatorname{supp}(\mathrm{d}\lambda). \tag{14.4}$$

The idea is to select the poles $-1/\zeta_\mu$ of the rational functions in $\mathbb{Q}_m$ to match the pole(s) of g closest to the support interval of $\mathrm{d}\lambda$.

In principle, the solution of the problem is rather simple: put $\omega_m(t) = \prod_{\mu=1}^{m}(1 + \zeta_\mu t)$ and construct, if possible, the n-point (poynomial) Gauss formula

$$\int_{\mathbb{R}} g(t)\frac{\mathrm{d}\lambda(t)}{\omega_m(t)} = \sum_{\nu=1}^{n} \lambda_\nu^G g(\tau_\nu^G), \quad g \in \mathbb{P}_{2n-1}, \tag{14.5}$$

for the modified measure $\mathrm{d}\hat{\lambda}(t) = \mathrm{d}\lambda(t)/\omega_m(t)$. Then

$$\tau_\nu = \tau_\nu^G, \quad \lambda_\nu = \omega_m(\tau_\nu^G)\lambda_\nu^G, \quad \nu = 1, 2, \ldots, n, \tag{14.6}$$

are the desired nodes and weights in (14.1).

We said "if possible", since in general ω_m is complex-valued, and the existence of a Gauss formula for $\mathrm{d}\hat{\lambda}$ is not guaranteed. There is no problem, however, if $\omega_m \geq 0$ on the support of $\mathrm{d}\lambda$. Fortunately, in many instances of practical interest, this is indeed the case.

There are a number of ways the formula (14.5) can be constructed: a discretization method using Gauss quadrature relative to $\mathrm{d}\lambda$ to do the discretization; repeated application of modification algorithms involving linear or quadratic divisors; special techniques to handle "difficult" poles, that is, poles very close to the support interval of $\mathrm{d}\lambda$. Rather than going into details (which can be found in Ga04, §3.1.4), we present an example taken from solid state physics.

Example 7. Generalized Fermi–Dirac integral.

This is the integral

$$F_k(\eta, \theta) = \int_0^\infty \frac{t^k\sqrt{1 + \theta t/2}}{\mathrm{e}^{-\eta+t} + 1}\,\mathrm{d}t,$$

where $\eta \in \mathbb{R}$, $\theta \geq 0$, and k is the Boltzmann constant ($=\frac{1}{2}$, $\frac{3}{2}$, or $\frac{5}{2}$). The ordinary Fermi–Dirac integral corresponds to $\theta = 0$.

The integral is conveniently rewritten as

$$F_k(\eta, \theta) = \int_0^\infty \frac{\sqrt{1 + \theta t/2}}{\mathrm{e}^{-\eta} + e^{-t}}\,\mathrm{d}\lambda^{[k]}(t), \quad \mathrm{d}\lambda^{[k]}(t) = t^k\mathrm{e}^{-t}\mathrm{d}t, \tag{14.7}$$

which is of the form (14.1) with $g(t) = \sqrt{1+\theta t/2}/(\mathrm{e}^{-\eta}+\mathrm{e}^{-t})$ and $\mathrm{d}\lambda = \mathrm{d}\lambda^{[k]}$ a generalized Laguerre measure. The poles of g evidently are $\eta + \mu i\pi$, $\mu = \pm 1, \pm 3, \pm 5, \ldots$, and all are "easy", that is, at a comfortable distance from the interval $[0, \infty]$. It is natural to take m even, and to incorporate the first $m/2$ pairs of conjugate complex poles. An easy computation then yields

$$\omega_m(t) = \prod_{\nu=1}^{m/2} [(1+\xi_\nu t)^2 + \eta_\nu t^2], \quad 2 \le m(\text{even}) \le 2n, \tag{14.8}$$

where

$$\xi_\nu = \frac{-\eta}{\eta^2 + (2\nu-1)^2\pi^2}, \quad \eta_\nu = \frac{(2\nu-1)\pi}{\eta^2 + (2\nu-1)^2\pi^2}. \tag{14.9}$$

Once the nodes and weights τ_ν, λ_ν have been obtained according to (14.6), the rational/polynomial quadrature approximation is given by

$$F_k(\eta, \theta) \approx \sum_{n=1}^{N} \lambda_n \frac{\sqrt{1+\theta\tau_n/2}}{\mathrm{e}^{-\eta} + \mathrm{e}^{-\tau_n}}. \tag{14.10}$$

It is computed in the `OPQ` routine

```
xw=fermi_dirac(N,m,eta,theta,k,eps0,Nmax)
```

where `eps0` is an error tolerance, `Nmax` a limit on the discretization parameter, and the other variables having obvious meanings.

15 Cauchy principal value integrals

When there is a (simple) pole inside the support interval $[a, b]$ of the measure $\mathrm{d}\lambda$, the integral must be taken in the sense of a *Cauchy principal value integral*

$$(\mathcal{C}f)(x; \mathrm{d}\lambda) := \int_a^b \frac{f(t)}{x-t}\,\mathrm{d}\lambda(t) = \lim_{\varepsilon \downarrow 0} \left(\int_a^{x-\varepsilon} + \int_{x+\varepsilon}^b \right) \frac{f(t)}{x-t}\,\mathrm{d}\lambda(t), \quad x \in (a,b). \tag{15.1}$$

There are two types of quadrature rules for Cauchy principal value integrals: one in which x occurs as a node, and one in which it does not. They have essentially different character and will be considered separately.

15.1 Modified quadrature rule

This is a quadrature rule of the form

$$(\mathcal{C}f)(x;\mathrm{d}\lambda) = c_0(x)f(x) + \sum_{\nu=1}^{n} c_\nu(x) f(\tau_\nu) + R_n(f;x). \tag{15.2}$$

It can be made "Gaussian", that is, $R_n(f;x) = 0$ for $f \in \mathbb{P}_{2n}$, by rewriting the integral in (15.1) as

$$(\mathcal{C}f)(x;\mathrm{d}\lambda) = f(x) \int_{\mathbb{R}} \frac{\mathrm{d}\lambda(t)}{x-t} - \int_{\mathbb{R}} \frac{f(x)-f(t)}{x-t}\,\mathrm{d}\lambda(t) \tag{15.3}$$

and applying the n-point Gauss formula for $\mathrm{d}\lambda$ to the second integral. The result is

$$(\mathcal{C}f)(x;\mathrm{d}\lambda) = \frac{\rho_n(x)}{\pi_n(x)}\, f(x) + \sum_{\nu=1}^{n} \lambda_\nu^G \frac{f(\tau_\nu^G)}{x-\tau_\nu^G} + R_n(f;x), \tag{15.4}$$

where $\rho_n(x)$ is the Cauchy principal value integral (5.14) and τ_ν^G, λ_ν^G are the Gauss nodes and weights for $\mathrm{d}\lambda$.

Formula (15.4) is not without numerical difficulties. The major one occurs when x approaches one of the Gauss nodes τ_ν^G, in which case two terms on the right go to infinity, but with opposite signs. In effect, this means that for x near a Gaussian node severe cancellation must occur.

The problem can be avoided by expanding the integral (15.1) in Cauchy integrals $\rho_k(x)$. Let $p_n(f;\,\cdot\,)$ be the polynomial of degree n interpolating f at the n Gauss nodes τ_ν^G and at x. The quadrature sum in (15.4) is then precisely the Cauchy integral of p_n,

$$(\mathcal{C}f)(x;\mathrm{d}\lambda) = \int_a^b \frac{p_n(f;t)}{x-t}\,\mathrm{d}\lambda(t) + R_n(f;x). \tag{15.5}$$

Expanding p_n in the orthogonal polynomials π_k,

$$p_n(f;t) = \sum_{k=0}^{n} a_k \pi_k(t), \quad a_k = \frac{1}{\|\pi_k\|^2} \int_a^b p_n(f;t)\pi_k(t)\mathrm{d}\lambda(t), \tag{15.6}$$

and integrating, one finds

$$(\mathcal{C}f)(x;\mathrm{d}\lambda) = \sum_{k=0}^{n} a_k \rho_k(x) + R_n(f;x), \tag{15.7}$$

where

$$(15.8)\quad a_k = \frac{1}{\|\pi_k\|^2}\sum_{\nu=1}^{n}\lambda_\nu^G f(\tau_\nu^G)\pi_k(\tau_\nu^G),\ k<n;\quad a_n = \sum_{\nu=1}^{n}\frac{f(x)-f(\tau_\nu^G)}{(x-\tau_\nu^G)\pi_n'(\tau_\nu^G)}\,.$$

The Cauchy integrals $\rho_k(x)$ in (15.7) can be computed in a stable manner by forward recursion; cf. the last paragraph of §5.1. This requires $\rho_0(x)$, which is either explicitly known or can be computed by the continued fraction algorithm; cf. §5.2. Some care must be exercised in computing the divided difference of f in the formula for a_n.

The procedure is inplemented in the `OPQ` routine

```
cpvi=cauchyPVI(N,x,f,ddf,iopt,ab,rho0)
```

with `iopt`$\neq$1, which produces the (`N+1`)-term approximation (15.7) where $R_n(f;x)$ is neglected. The input parameter `ddf` is a routine for computing the divided difference of f in a stable manner. It is used only if `iopt`$\neq$1. The meaning of the other parameters is obvious.

15.2 Quadrature rule in the strict sense

This rule, in which the node $t=x$ is absent, is obtained by interpolating f at the n Gauss nodes τ_ν^G by a polynomial $p_{n-1}(f;\,\cdot\,)$ of degree $n-1$,

$$f(t) = p_{n-1}(f;t) + E_{n-1}(f;t),\quad p_{n-1}(f;t) = \sum_{\nu=1}^{n}\frac{\pi_n(t)}{(t-\tau_\nu^G)\pi_n'(\tau_\nu^G)}\,f(\tau_\nu^G),$$

where E_{n-1} is the interpolation error, which vanishes identically if $f\in\mathbb{P}_{n-1}$. The formula to be derived, therefore, will have degree of exactness $n-1$ (which can be shown to be maximum possible). Integrating in the sense of (15.1) yields

$$(15.9)\qquad (\mathcal{C}f)(x;\mathrm{d}\lambda) = \sum_{\nu=1}^{n}\frac{\rho_n(x)-\rho_n(\tau_\nu^G)}{(x-\tau_\nu^G)\pi_n'(\tau_\nu^G)}\,f(\tau_\nu^G) + R_n^*(f;x),$$

where $R_n^*(f;x) = \int_a^b E_{n-1}(f;t)\mathrm{d}\lambda(t)/(x-t)$.

This formula, too, suffers from severe cancellation errors when x is near a Gauss node. The resolution of this problem is similar (in fact, simpler) as

in §15.1: expand $p_{n-1}(f;\,\cdot\,)$ in the orthogonal polynomials π_k to obtain

$$(\mathcal{C}f)(x;\mathrm{d}\lambda) = \sum_{k=0}^{n-1} a'_k \rho_k(x) + R^*_n(f;x), \qquad a'_k = \frac{1}{\|\pi_k\|^2}\int_a^b p_{n-1}(f;t)\pi_k(t)\mathrm{d}\lambda(t). \tag{15.10}$$

It turns out that

$$a'_k = a_k, \quad k = 0, 1, \ldots, n-1, \tag{15.11}$$

where a_k, $k < n$, is given by (15.8). This is implemented in the `OPQ` routine `cauchyPVI.m` with `iopt`=1.

16 Polynomials orthogonal on several intervals

We are given a finite set of intervals $[c_j, d_j]$, which may be disjoint or not, and on each interval a positive measure $\mathrm{d}\lambda_j$. Let $\mathrm{d}\lambda$ be the "composite" measure

$$\mathrm{d}\lambda(t) = \sum_j \chi_{[c_j,d_j]}(t)\mathrm{d}\lambda_j(t), \tag{16.1}$$

where $\chi_{[c_j,d_j]}$ is the characteristic function of the interval $[c_j, d_j]$. Assuming known the Jacobi matrices $\boldsymbol{J}^{(j)} = \boldsymbol{J}_n(\mathrm{d}\lambda_j)$ of the component measures $\mathrm{d}\lambda_j$, we now consider the problem of determining the Jacobi matrix $\boldsymbol{J} = \boldsymbol{J}_n(\mathrm{d}\lambda)$ of the composed measure $\mathrm{d}\lambda$. We provide two solutions, one based on Stieltjes's procedure, and one based on the modified Chebyshev algorithm.

16.1 Solution by Stieltjes's procedure

The main problem in applying Stieltjes's procedure is to compute the inner products $(t\pi_k, \pi_k)_{\mathrm{d}\lambda}$ and $(\pi_k, \pi_k)_{\mathrm{d}\lambda}$ for $k = 0, 1, 2, \ldots, n-1$. This can be done by using n-point Gaussian quadrature on each component interval,

$$\int_{c_j}^{d_j} p(t)\mathrm{d}\lambda_j(t) = \sum_{\nu=1}^{n} \lambda_\nu^{(j)} p(\tau_\nu^{(j)}), \quad p \in \mathbb{P}_{2n-1}. \tag{16.2}$$

Here we use (11.5) to express the quadrature sum in terms of the Jacobi matrix $\boldsymbol{J}^{(j)}$,

$$\int_{c_j}^{d_j} p(t)\mathrm{d}\lambda_j(t) = \beta_0^{(j)} \boldsymbol{e}_1^{\mathrm{T}} p(\boldsymbol{J}^{(j)})\boldsymbol{e}_1, \quad \beta_0^{(j)} = \int_{c_j}^{d_j} \mathrm{d}\lambda_j(t). \tag{16.3}$$

Then, for the inner products $(t\pi_k, \pi_k)_{\mathrm{d}\lambda}$, $k \le n-1$, we get

$$\begin{aligned}(t\pi_k, \pi_k)_{\mathrm{d}\lambda} &= \int_{\mathbb{R}} t\pi_k^2(t)\mathrm{d}\lambda(t) = \sum_j \int_{c_j}^{d_j} t\pi_k^2(t)\mathrm{d}\lambda_j(t) \\ &= \sum_j \beta_0^{(j)} \boldsymbol{e}_1^{\mathrm{T}} \boldsymbol{J}^{(j)} [\pi_k(\boldsymbol{J}^{(j)})]^2 \boldsymbol{e}_1 \\ &= \sum_j \beta_0^{(j)} \boldsymbol{e}_1^{\mathrm{T}} [\pi_k(\boldsymbol{J}^{(j)})]^{\mathrm{T}} \boldsymbol{J}^{(j)} \pi_k(\boldsymbol{J}^{(j)}) \boldsymbol{e}_1\end{aligned}$$

and for $(\pi_k, \pi_k)_{\mathrm{d}\lambda}$ similarly (in fact, simpler)

$$(\pi_k, \pi_k)_{\mathrm{d}\lambda} = \sum_j \beta_0^{(j)} \boldsymbol{e}_1^{\mathrm{T}} [\pi_k(\boldsymbol{J}^{(j)})]^{\mathrm{T}} \pi_k(\boldsymbol{J}^{(j)}) \boldsymbol{e}_1.$$

This can be conveniently expressed in terms of the vectors

$$\boldsymbol{\zeta}_k^{(j)} := \pi_k(\boldsymbol{J}^{(j)})\boldsymbol{e}_1, \quad \boldsymbol{e}_1 = [1, 0, \ldots, 0]^T,$$

which, as required in Stieltjes's procedure, can be updated by means of the basic three-term recurrence relation. This leads to the following algorithm.

Algorithm 7 Stieltjes procedure for polynomials orthogonal on several intervals

initialization:

$$\begin{aligned}&\boldsymbol{\zeta}_0^{(j)} = \boldsymbol{e}_1, \quad \boldsymbol{\zeta}_{-1}^{(j)} = 0 \quad (\text{all } j), \\ &\alpha_0 = \frac{\sum_j \beta_0^{(j)} \boldsymbol{e}_1^{\mathrm{T}} \boldsymbol{J}^{(j)} \boldsymbol{e}_1}{\sum_j \beta_0^{(j)}}, \quad \beta_0 = \sum_j \beta_0^{(j)}.\end{aligned}$$

continuation (if $n > 1$): for $k = 0, 1, \ldots, n-2$ do

$$\begin{aligned}&\boldsymbol{\zeta}_{k+1}^{(j)} = (\boldsymbol{J}^{(j)} - \alpha_k \boldsymbol{I})\boldsymbol{\zeta}_k^{(j)} - \beta_k \boldsymbol{\zeta}_{k-1}^{(j)} \quad (\text{all } j), \\ &\alpha_{k+1} = \frac{\sum_j \beta_0^{(j)} \boldsymbol{\zeta}_{k+1}^{(j)\mathrm{T}} \boldsymbol{J}^{(j)} \boldsymbol{\zeta}_{k+1}^{(j)}}{\sum_j \beta_0^{(j)} \boldsymbol{\zeta}_{k+1}^{(j)\mathrm{T}} \boldsymbol{\zeta}_{k+1}^{(j)}}, \quad \beta_{k+1} = \frac{\sum_j \beta_0^{(j)} \boldsymbol{\zeta}_{k+1}^{(j)\mathrm{T}} \boldsymbol{\zeta}_{k+1}^{(j)}}{\sum_j \beta_0^{(j)} \boldsymbol{\zeta}_k^{(j)\mathrm{T}} \boldsymbol{\zeta}_k^{(j)}}.\end{aligned}$$

In Matlab, this is implemented in the `OPQ` routine

```
ab=r_multidomain_sti(N,abmd)
```

where `abmd` is the array containing the (α, β)-coefficients of the measures $\mathrm{d}\lambda^{(j)}$.

Example 8. Example 1, revisited.

This is the case of two identical intervals $[-1, 1]$ and two measures $\mathrm{d}\lambda^{(j)}$ on $[-1, 1]$, one a multiple c of the Legendre measure, the other the Chebyshev measure. This was solved in Example 1 by a 2-component discretization method. The solution by the 2-domain algorithm of this subsection, in Matlab, looks as follows:

```
ab1=r_jacobi(N); ab1(1,2)=2*c;
ab2=r_jacobi(N,-.5);
abmd=[ab1 ab2];
ab=r_multidomain_sti(N,abmd)
```

It produces results identical with those produced by the method of Example 1.

16.2 Solution by the modified Chebyshev algorithm

The quadrature procedure used in §16.1 to compute inner products can equally be applied to compute the first $2n$ modified moments of $\mathrm{d}\lambda$,

$$m_k = \sum_j \int_{c_j}^{d_j} p_k(t)\mathrm{d}\lambda_j(t) = \sum_j \beta_0^{(j)} \boldsymbol{e}_1^{\mathrm{T}} p_k(\boldsymbol{J}^{(j)})\boldsymbol{e}_1. \tag{16.4}$$

The relevant vectors are now

$$\boldsymbol{z}_k^{(j)} := p_k(\boldsymbol{J}^{(j)})\boldsymbol{e}_1, \quad \boldsymbol{e}_1 = [1, 0, \ldots, 0]^{\mathrm{T}},$$

and the computation proceeds as in

Algorithm 8 Modified moments for polynomials orthogonal on several intervals

initialization

$$\boldsymbol{z}_0^{(j)} = \boldsymbol{e}_1,\ \boldsymbol{z}_{-1}^{(j)} = 0 \text{ (all } j), \quad m_0 = \sum_j \beta_0^{(j)}.$$

continuation: for $k = 0, 1, \ldots, 2n-2$ do

$$\boldsymbol{z}_{k+1}^{(j)} = (\boldsymbol{J}^{(j)} - a_k \boldsymbol{I})\boldsymbol{z}_k^{(j)} - b_k \boldsymbol{z}_{k-1}^{(j)} \quad (\text{all } j),$$
$$m_{k+1} = \sum_j \beta_0^{(j)} \boldsymbol{z}_{k+1}^{(j)\mathrm{T}} \boldsymbol{e}_1.$$

With these moments at hand, we can apply Algorithm 1 to obtain the desired recurrence coefficients. This is done in the `OPQ` routine

```
ab=r_multidomain_cheb(N,abmd,abmm)
```

The input array `abmd` has the same meaning as in the routine of §16.1, and `abmm` is a (`2N`×2) array of the recurrence coefficients a_k, b_k generating the polynomials p_k.

Applied to Example 8, the Matlab program, using Legendre moments (p_k the monic Legendre polynomials), is as follows:

```
abm=r_jacobi(2*N-1);
ab1=abm(1:N,:); ab1(1,2)=2*c;
ab2=r_jacobi(N,-.5);
abmd=[ab1 ab2];
ab=r_multidomain_cheb(N,abmd,abm)
```

It produces results identical with those obtained in §16.1, but takes about three times as long to run.

17 Quadrature estimates of matrix functionals

The problem to be considered here is to find lower and upper bounds for the quadratic form

$$\boldsymbol{u}^{\mathrm{T}} f(\boldsymbol{A})\boldsymbol{u}, \quad \boldsymbol{u} \in \mathbb{R}^N, \ \|\boldsymbol{u}\| = 1, \tag{17.1}$$

where $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ is a symmetric, positive definite matrix, f a smooth function (for which $f(\boldsymbol{A})$ makes sense), and $\boldsymbol{u}$ a given vector. While this looks more like a linear algebra problem, it can actually be solved, for functions f

with derivatives of constant sign, by applying Gauss-type quadrature rules. The connecting link is provided by the *spectral resolution* of $\boldsymbol{A}$,

$$\boldsymbol{A}\boldsymbol{V} = \boldsymbol{V}\boldsymbol{\Lambda},\ \boldsymbol{\Lambda} = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_N),\ \boldsymbol{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_N], \tag{17.2}$$

where λ_k are the eigenvalues of $\boldsymbol{A}$ (which for simplicity are assumed distinct), and $\boldsymbol{v}_k$ the normalized eigenvectors of $\boldsymbol{A}$. If we put

$$\boldsymbol{u} = \sum_{k=1}^{N} \rho_k \boldsymbol{v}_k = \boldsymbol{V}\boldsymbol{\rho},\ \boldsymbol{\rho} = [\rho_1, \rho_2, \ldots, \rho_N]^{\mathrm{T}}, \tag{17.3}$$

and again for simplicity assume $\rho_k \neq 0$, all k, then

$$\begin{aligned} \boldsymbol{u}^{\mathrm{T}} f(\boldsymbol{A})\boldsymbol{u} &= \boldsymbol{\rho}^{\mathrm{T}}\boldsymbol{V}^{\mathrm{T}}\boldsymbol{V} f(\boldsymbol{\Lambda})\boldsymbol{V}^{\mathrm{T}}\boldsymbol{V}\boldsymbol{\rho} = \boldsymbol{\rho}^{\mathrm{T}} f(\boldsymbol{\Lambda})\boldsymbol{\rho}, \\ &= \sum_{k=1}^{N} \rho_k^2 f(\lambda_k) =: \int_{\mathbb{R}_+} f(t)\mathrm{d}\rho_N(t). \end{aligned} \tag{17.4}$$

This shows how the matrix functional is related to an integral relative to a discrete positive measure. Now we know from (11.10) and (11.15) how Gauss–Radau or Gauss–Lobatto rules (and for that matter also ordinary Gauss rules, in view of $R_n^G = [f^{(2n)}(\tau)/(2n)!] \int_a^b [\pi_n(t;\mathrm{d}\lambda)]^2 \mathrm{d}\lambda(t)$, $a < \tau < b$) can be applied to obtain two-sided bounds for (17.4) when some derivative of f has constant sign. To generate these quadrature rules, we need the orthogonal polynomials for the measure $\mathrm{d}\rho_N$, and for these the Jacobi matrix $\boldsymbol{J}_N(\mathrm{d}\rho_N)$. The latter, in principle, could be computed by Lanczos's algorithm of §3.2. However, in the present application this would require knowledge of the eigenvalues λ_k and expansion coefficients ρ_k, which are too expensive to compute. Fortunately, there is an alternative way to implement Lanczos's algorithm that works directly with the matrix $\boldsymbol{A}$ and requires only multiplications of $\boldsymbol{A}$ into vectors and the computation of inner products.

17.1 Lanczos algorithm

Let ρ_k be as in (17.4) and $h_0 = \sum_{k=1}^{N} \rho_k \boldsymbol{v}_k$ $(= \boldsymbol{u})$, $\|h_0\| = 1$, as in (17.3).

Algorithm 9 Lanczos algorithm

initialization:

$$\boldsymbol{h}_0 \text{ prescribed with } \|\boldsymbol{h}_0\| = 1, \quad \boldsymbol{h}_{-1} = \boldsymbol{0}.$$

continuation: for $j = 0, 1, \ldots, N-1$ do

$$\begin{aligned}
\alpha_j &= \boldsymbol{h}_j^{\mathrm{T}} \boldsymbol{A} \boldsymbol{h}_j, \\
\tilde{\boldsymbol{h}}_{j+1} &= (\boldsymbol{A} - \alpha_j \boldsymbol{I}) \boldsymbol{h}_j - \gamma_j \boldsymbol{h}_{j-1}, \\
\gamma_{j+1} &= \|\tilde{\boldsymbol{h}}_{j+1}\|, \\
\boldsymbol{h}_{j+1} &= \tilde{\boldsymbol{h}}_{j+1} / \gamma_{j+1}.
\end{aligned}$$

While γ_0 in Algorithm 9 can be arbitrary (it multiplies $\boldsymbol{h}_{-1} = \boldsymbol{0}$), it is convenient to define $\gamma_0 = 1$. The vectors $\boldsymbol{h}_0, \boldsymbol{h}_1, \ldots, \boldsymbol{h}_N$ generated by Algorithm 9 are called *Lanczos vectors*. It can be shown that α_k generated by the Lanczos algorithm is precisely $\alpha_k(\mathrm{d}\rho_N)$, and $\gamma_k = \sqrt{\beta_k(\mathrm{d}\rho_N)}$, for $k = 0, 1, 2, \ldots, N-1$. This provides us with the Jacobi matrix $\boldsymbol{J}_N(\mathrm{d}\rho_N)$. It is true that the algorithm becomes unstable as j approaches N, but in the applications of interest here, only small values of j are needed.

17.2 Examples

Example 9. Error bounds for linear algebraic systems.

Consider the system

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \quad \boldsymbol{A} \text{ symmetric, positive definite.} \tag{17.5}$$

Given an approximation $\boldsymbol{x}^* \approx \boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$ to the exact solution $\boldsymbol{x}$, and the residual vector $\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^*$, we have $\boldsymbol{x} - \boldsymbol{x}^* = \boldsymbol{A}^{-1}\boldsymbol{b} + \boldsymbol{A}^{-1}(\boldsymbol{r} - \boldsymbol{b}) = \boldsymbol{A}^{-1}\boldsymbol{r}$, thus

$$\|\boldsymbol{x} - \boldsymbol{x}^*\|^2 = (\boldsymbol{A}^{-1}\boldsymbol{r})^{\mathrm{T}} \boldsymbol{A}^{-1}\boldsymbol{r} = \boldsymbol{r}^{\mathrm{T}} \boldsymbol{A}^{-2} \boldsymbol{r},$$

and therefore

$$\|\boldsymbol{x} - \boldsymbol{x}^*\|^2 = \|\boldsymbol{r}\|^2 \cdot \boldsymbol{u}^{\mathrm{T}} f(\boldsymbol{A}) \boldsymbol{u}, \tag{17.6}$$

where $\boldsymbol{u} = \boldsymbol{r}/\|\boldsymbol{r}\|$ and $f(t) = t^{-2}$. All derivatives of f are here of constant sign on $\mathbb{R}_+$,

$$f^{(2n)}(t) > 0, \quad f^{(2n+1)}(t) < 0 \quad \text{for } t \in \mathbb{R}_+. \tag{17.7}$$

By (17.4), we now have

$$\|\boldsymbol{x} - \boldsymbol{x}^*\| = \|\boldsymbol{r}\|^2 \int_{\mathbb{R}_+} t^{-2} \mathrm{d}\rho_N(t). \tag{17.8}$$

The n-point Gauss quadrature rule applied to the integral on the right of (17.8), by the first inequality in (17.7), yields a *lower bound* of $\|\boldsymbol{x} - \boldsymbol{x}^*\|$, without having to know the exact support interval of $\mathrm{d}\rho_N$. If, on the other hand, we know that the support of $\mathrm{d}\rho_N$ is contained in some interval $[a, b]$, $0 < a < b$, we can get a lower bound also from the right-handed $(n+1)$-point Gauss–Radau formula, and *upper bounds* from the left-handed $(n+1)$-point Gauss–Radau formula on $[a, b]$, or from the $(n+2)$-point Gauss–Lobatto formula on $[a, b]$.

Example 10. Diagonal elements of $\boldsymbol{A}^{-1}$.

Here, trivially

$$\boldsymbol{u}^{\mathrm{T}} f(\boldsymbol{A})\boldsymbol{u} = \boldsymbol{e}_i^{\mathrm{T}} \boldsymbol{A}^{-1} \boldsymbol{e}_i, \tag{17.9}$$

where $f(t) = t^{-1}$ and $\boldsymbol{e}_i$ is the ith coordinate vector. Using n-point Gauss quadrature in (17.4), with $n < N$, yields

$$(\boldsymbol{A}^{-1})_{ii} = \int_{\mathbb{R}_+} t^{-1}\, \mathrm{d}\rho_N(t) > \boldsymbol{e}_1^{\mathrm{T}} \boldsymbol{J}_n^{-1} \boldsymbol{e}_1, \quad \boldsymbol{e}_1^{\mathrm{T}} = [1, 0, \ldots, 0] \in \mathbb{R}^n. \tag{17.10}$$

Suppose we take $n = 2$ steps of Algorithm 9 to compute

$$\boldsymbol{J}_2 = \begin{bmatrix} \alpha_0 & \gamma_1 \\ \gamma_1 & \alpha_1 \end{bmatrix}.$$

We get

$$\begin{aligned} &\alpha_0 = a_{ii}, \\ &\tilde{\boldsymbol{h}}_1 = (\boldsymbol{A} - \alpha_0 \boldsymbol{I})\boldsymbol{e}_i = [a_{1i}, \ldots, a_{i-1,i}, 0, a_{i+1,i}, \ldots, a_{Ni}]^{\mathrm{T}}, \\ &\gamma_1 = \sqrt{\sum_{k \neq i} a_{ki}^2} =: s_i, \\ &\boldsymbol{h}_1 = \tilde{\boldsymbol{h}}_1 / s_i, \\ &\alpha_1 = \frac{1}{s_i^2} \tilde{\boldsymbol{h}}_1^{\mathrm{T}} \boldsymbol{A} \tilde{\boldsymbol{h}}_1 = \frac{1}{s_i^2} \sum_{k \neq i} \sum_{\ell \neq i} a_{k\ell} a_{ki} a_{\ell i}. \end{aligned} \tag{17.11}$$

But

$$\boldsymbol{J}_2^{-1} = \frac{1}{\alpha_0 \alpha_1 - \gamma_1^2} \begin{bmatrix} \alpha_1 & -\gamma_1 \\ -\gamma_1 & \alpha_0 \end{bmatrix}, \quad \boldsymbol{e}_1^{\mathrm{T}} \boldsymbol{J}_2^{-1} \boldsymbol{e}_1 = \frac{\alpha_1}{\alpha_0 \alpha_1 - \gamma_1^2},$$

so that by (17.10) with $n = 2$, and (17.11),

$$(\boldsymbol{A}^{-1})_{ii} > \frac{\sum_{k\neq i}\sum_{\ell\neq i} a_{k\ell}a_{ki}a_{\ell i}}{a_{ii}\sum_{k\neq i}\sum_{\ell\neq i} a_{k\ell}a_{ki}a_{\ell i} - \left(\sum_{k\neq i} a_{ki}^2\right)^2}\,. \tag{17.12}$$

Simpler bounds, both lower and upper, can be obtained by the 2-point Gauss-Radau and Gauss-Lobatto formulae, which however require knowledge of an interval $[a, b]$, $0 < a < b$, containing the spectrum of $\boldsymbol{A}$.

Exercises to Part II (Stars indicate more advanced exercises.)

1. Prove (11.5).

2. Prove that complex zeros of the Sobolev orthogonal polynomials of Example 6 must be purely imaginary.

3*. Circle theorems for quadrature weights (cf. P.J. Davis and P. Rabinowitz, J. Math. Anal. Appl. 2 (1961), 428–437).

   (a) Gauss–Jacobi quadrature

   Let $w(t) = (1-t)^\alpha(1+t)^\beta$ be the Jacobi weight function. It is known (Sz75, eqn (15.3.10)) that the nodes τ_ν and weights λ_ν of the n-point Gauss–Jacobi quadrature formula satisfy

   $$\lambda_\nu \sim \frac{\pi}{n}\, w(\tau_\nu)\sqrt{1-\tau_\nu^2}, \quad n \to \infty,$$

   for τ_ν on any compact interval contained in $(-1,1)$. Thus, suitably normalized weights, plotted against the nodes, lie asymptotically on the unit circle. Use Matlab to demonstrate this graphically.

   (b) Gauss quadrature for the logarithmic weight function $w(t) = t^\alpha \ln(1/t)$ on $[0,1]$ (cf. Ga04, Example 2.27).

   Try, numerically, to find a circle theorem in this case also, and experiment with different values of the parameter $\alpha > -1$. (Use the `OPQ` routine `r_jaclog.m` to generate the recurrence coefficients of the orthogonal polynomials for the weight function w.)

   (c) Gauss–Kronrod quadrature.

   With w as in (a), the analogous result for the $2n+1$ nodes τ_ν and weights λ_ν of the $(2n+1)$-point Gauss–Kronrod formula is expected to be

   $$\lambda_\nu \sim \frac{\pi}{2n}\, w(\tau_\nu)\sqrt{1-\tau_\nu^2}, \quad n \to \infty.$$

   That this indeed is the case, when $\alpha, \beta \in [0, \frac{5}{2})$, follows from Theorem 2 in F. Peherstorfer and K. Petras, Numer. Math. 95 (2003), 689–706. Use Matlab to illustrate this graphically.

   (d) Experiment with the Gauss–Kronrod formula for the logarithmic weight function of (b), when $\alpha = 0$.

4. Discrete orthogonality.

   Let $\pi_k(\,\cdot\,;\mathrm{d}\lambda)$, $k = 0, 1, 2, \ldots$, be the orthogonal polynomials relative to an absolutely continuous measure. Show that for each $N \geq 2$, the first N of them are orthogonal with respect to the discrete inner product

$$(p, q)_N = \sum_{\nu=0}^{N-1} \lambda_\nu^G p(\tau_\nu^G) q(\tau_\nu^G),$$

   where τ_ν^G, λ_ν^G are the nodes and weights of the N-point Gauss formula for $\mathrm{d}\lambda$. Moreover, $\|\pi_k\|_N^2 = \|\pi_k\|_{\mathrm{d}\lambda}^2$ for $k \leq n-1$.

5. (a) Consider the Cauchy integral

$$\rho_n(z) = \rho_n(z;\mathrm{d}\lambda) = \int_a^b \frac{\pi_n(t;\mathrm{d}\lambda)}{z-t}\,\mathrm{d}\lambda(t),$$

   where $[a, b]$ is the support of $\mathrm{d}\lambda$. Show that

$$\rho_n(z) = O(z^{-n-1}) \quad \text{as } z \to \infty.$$

   {*Hint*: Expand the integral defining $\rho_n(z)$ in descending powers of z.}

   (b) Show that

$$\int_a^b \frac{\mathrm{d}\lambda(t)}{z-t} - \frac{\sigma_n(z)}{\pi_n(z)} = \frac{\rho_n(z)}{\pi_n(z)} = O(z^{-2n-1}) \quad \text{as } z \to \infty.$$

   {*Hint*: Use (5.8).}

   (c) Consider the partial fraction decomposition

$$\frac{\sigma_n(z)}{\pi_n(z)} = \sum_{\nu=1}^{n} \frac{\lambda_\nu}{z - \tau_\nu^G}$$

   of $\sigma_n(z)/\pi_n(z)$ in (5.8). Use (b) to show that $\lambda_\nu = \lambda_\nu^G$ are the weights of the n-point Gaussian quadrature formula for $\mathrm{d}\lambda$. In particular, show that

$$\lambda_\nu^G = \frac{\sigma_n(\tau_\nu^G)}{\pi_n'(\tau_\nu^G)}\,.$$

(d) Discuss what happens if $z \to x$, $x \in (a, b)$.

6. Characterize the nodes τ_ν^b in (11.9) as zeros of an orthogonal polynomial of degree n, and identify the appropriate Gauss–Radau matrix for (11.9).

7. Prove (11.10). {*Hint*: Use the fact that both formulae (11.6) and (11.9) are interpolatory.}

8. (a) Prove the first formula in (11.11). {*Hint*: Use the relation between the Jacobi polynomials $P_k = P_k^{(\alpha,\beta)}$ customarily defined and the monic Jacobi polynomials $\pi_k = \pi_k^{(\alpha,\beta)}$, expressed by $P_k(t) = 2^{-k}\binom{2k+\alpha+\beta}{k}\pi_k(t)$. You also need $P_k(-1) = (-1)^k\binom{k+\beta}{k}$ and the β-coefficient for Jacobi polynomials, $\beta_n^J = 4n(n+\alpha)(n+\beta)(n+\alpha+\beta)/(2n+\alpha+\beta)^2(2n+\alpha+\beta+1)(2n+\alpha+\beta-1)$.}

   (b) Prove the second formula in (11.11). {*Hint*: With $\pi_k^{(\alpha)}$ and $L_k^{(\alpha)}$ denoting the monic resp. conventional generalized Laguerre polynomials, use $L_k^{(\alpha)}(t) = \big((-1)^k/k!\big)\,\pi_k^{(\alpha)}(t)$. You also need $L_k^{(\alpha)}(0) = \binom{k+\alpha}{k}$, and $\beta_n^{\mathrm{L}} = n(n+\alpha)$.}

9. Prove (11.16). {*Hint*: With notation as in the hint to Exercise 8(a), use $P_k(1) = \binom{k+\alpha}{k}$ in addition to the information provided there.}

10. The (left-handed) *generalized Gauss–Radau formula* is

$$\int_a^\infty f(t)\,\mathrm{d}\lambda(t) = \sum_{\rho=0}^{r-1} \lambda_0^{(\rho)} f^{(\rho)}(a) + \sum_{\nu=1}^{n} \lambda_\nu^R f(\tau_\nu^R) + R_{n,r}^R(f),$$

where $r > 1$ is the multiplicity of the end point $\tau_0 = a$, and $R_{n,r}^R(f) = 0$ for $f \in \mathbb{P}_{2n-1+r}$. Let $\mathrm{d}\lambda^{[r]}(t) = (t-a)^r \mathrm{d}\lambda(t)$ and $\tau_\nu^{[r]}$, $\lambda^{[r]}$, $\nu = 1, 2, \ldots, n$, be the nodes and weights of the n-point Gauss formula for $\mathrm{d}\lambda^{[r]}$.

(a) Show that

$$\tau_\nu^R = \tau_\nu^{[r]}, \quad \lambda_\nu^R = \frac{\lambda_\nu^{[r]}}{(\tau_\nu^R - a)^r}, \quad \nu = 1, 2, \ldots, n.$$

(b) Show hat not only the internal weights λ_ν^R are all positive (why?), but also the boundary weights λ_0, λ_0' if $r = 2$.

11. The *generalized Gauss–Lobatto formula* is

$$\int_a^b f(t)\,\mathrm{d}\lambda(t) = \sum_{\rho=0}^{r-1} \lambda_0^{(\rho)} f^{(\rho)}(a) + \sum_{\nu=1}^{n} \lambda_\nu^L f(\tau_\nu^L) + \sum_{\rho=0}^{r-1} (-1)^\rho \lambda_{n+1}^{(\rho)} f^{(\rho)}(b) + R_{n,r}^L(f),$$

where $r > 1$ is the multiplicity of the end points $\tau_0 = a$, $\tau_{n+1} = b$, and $R_{n,r}^L(f) = 0$ for $\mathbb{P}_{2n-1+2r}$. Let $\mathrm{d}\lambda^{[r]}(t) = [(t-a)(b-t)]^r \mathrm{d}\lambda(t)$ and $\tau_\nu^{[r]}$, $\lambda_\nu^{[r]}$, $\nu = 1, 2, \ldots, n$, be the nodes and weights of the n-point Gauss formula for $\mathrm{d}\lambda^{[r]}$.

(a) Show that

$$\tau_\nu^L = \tau_\nu^{[r]}, \quad \lambda_\nu^L = \frac{\lambda_\nu^{[r]}}{[(\tau_\nu^L - a)(b - \tau_\nu^L)]^r}, \quad \nu = 1, 2, \ldots, n.$$

(b) Show that not only the internal weights λ_ν^L are all positive (why?), but also the boundary weights λ_0, λ_0' and λ_{n+1}, λ_{n+1}'.

(c) Show that $\lambda_0^{(\rho)} = \lambda_{n+1}^{(\rho)}$, $\rho = 0, 1, \ldots, r-1$, if the measure $\mathrm{d}\lambda$ is symmetric.

12*. Generalized Gauss-Radau quadrature.

(a) Write a Matlab routine `gradau.m` for generating the generalized Gauss-Radau quadrature rule of Exercise 10 for a measure $\mathrm{d}\lambda$ on $[a, \infty]$, having a fixed node a of multiplicity r, $r > 1$. {*Hint*: To compute the boundary weights, set up an (upper triangular) system of linear equations by applying the formula in turn with $\pi_n^2(t)$, $(t-a)\pi_n^2(t), \ldots, (t-a)^{r-1}\pi_n^2(t)$, where $\pi_n(t) = \prod_{\nu=1}^n (t - \tau_\nu^R)$.}

(b) Check your routine against the known formulae with $r = 2$ for the Legendre and Chebyshev measures (see Ga04, Examples 3.10 and 3.11). Devise and implement a check that works for arbitrary $r \geq 2$ and, in particular, for $r = 1$.

(c) Use your routine to explore positivity of the boundary weights and see whether you can come up with any conjectures.

13*. Generalized Gauss-Lobatto quadrature.

(a) Write a Matlab routine `globatto.m` for generating the generalized Gauss-Lobatto rule of Exercise 11 for a measure $d\lambda$ on $[a, b]$, having fixed nodes at a and b of multiplicity r, $r > 1$. For simplicity, start with the case $r \geq 2$ even; then indicate the changes necessary to deal with odd values of r. {*Hint*: Similar to the hint in Exercise 12(a).}

(b) Check your routine against the known formulae with $r = 2$ for the Legendre and Chebyshev measures (see Ga04, Examples 3.13 and 3.14). Devise and implement a check that works for arbitrary $r \geq 2$ and, in particular, for $r = 1$.

(c) Explore the positivity of the boundary weights $\lambda_0^{(\rho)}$ and the quantities $\lambda_{n+1}^{(\rho)}$ in the quadrature formula.

14. Show that the monic Stieltjes polynomial π_{n+1}^K in (12.3) exists uniquely.

15. (a) Let $d\lambda$ be a positive measure. Use approximation theory to show that the minimum of $\int_{\mathbb{R}} |\pi(t)|^p d\lambda(t)$, $1 < p < \infty$, extended over all monic polynomials π of degree n is uniquely determined.

(b) Show that the minimizer of the extremal problem in (a), when $p = 2s + 2$, $s \geq 0$ an integer, is the s-orthogonal polynomial $\pi = \pi_{n,s}$. {*Hint*: Differentiate the integral partially with respect to the variable coefficients of π.}

16. (a) Show that r in (13.2) has to be odd.

(b) Show that in (13.2) with r as in (13.3), one cannot have $k > n$.

17. Derive (14.8) and (14.9).

18. Derive (15.4) from (15.3) and explain he meaning of $R_n(f; x)$. {*Hint*: Use Exercise 5(c) and (5.8).}

19. Show that $p_n(f; t)$ in (15.5) is

$$p_n(f;t) = \frac{\pi_n(t)}{\pi_n(x)}\, f(x) + \sum_{\nu=1}^{n} \frac{(t-x)\pi_n(t)}{(t-\tau_\nu^G)(\tau_\nu^G - x)\pi_n'(\tau_\nu^G)}\, f(\tau_\nu^G),$$

and thus prove (15.5). {*Hint*: Use Exercise 5(c).}

20. Derive (15.7) and (15.8). {*Hint*: For $k < n$, use Gauss quadrature, and for $k = n$ insert the expression for $p_n(f;t)$ from Exercise 19 into the formula for a_n in (15.6). Also use the fact that the elementary Lagrange interpolation polynomials sum up to 1.}

21. Derive (15.9).

22. Prove (15.11). {*Hint*: Use Exercise 4.}

23. (a) Prove that the Lanczos vectors are mutually orthonormal.

    (b) Show that the vectors $\{\boldsymbol{h}_j\}_{j=0}^n$, $n < N$, form an orthonormal basis of the *Krylov space*

$$\mathcal{K}_n(\boldsymbol{A}, \boldsymbol{h}_0) = \operatorname{span}(\boldsymbol{h}_0, \boldsymbol{A}\boldsymbol{h}_0, \ldots, \boldsymbol{A}^n\boldsymbol{h}_0).$$

    (c) Prove that

$$\boldsymbol{h}_j = p_j(\boldsymbol{A})\boldsymbol{h}_0, \quad j = 0, 1, \ldots, N,$$

    where p_j is a polynomial of degree j satisfying the three-term recurrence relation

$$\gamma_{j+1}p_{j+1}(\lambda) = (\lambda - \alpha_j)p_j(\lambda) - \gamma_j p_{j-1}(\lambda), \qquad j = 0, 1, \ldots, N-1,$$
$$p_0(\lambda) = 1, \quad p_{-1}(\lambda) = 0.$$

    {*Hint*: Use mathematical induction.}

24. Prove that the polynomial p_k of Exercise 23(c) is equal to the orthonormal polynomial $\tilde{\pi}_k(\,\cdot\,; \mathrm{d}\rho_N)$. {*Hint*: Use the spectral resolution of $\boldsymbol{A}$ and Exercises 23(a) and (c).}

25. Derive the bounds for $(\boldsymbol{A}^{-1})_{ii}$ hinted at in the last sentence of Example 10.

PART III APPROXIMATION

18 Polynomial least squares approximation

18.1 Classical least squares problem

We are given N data points (t_k, f_k), $k = 1, 2, \ldots, N$, and wish to find a polynomial $\hat{\pi}_n$ of degree $\leq n$, $n < N$, such that a weighted average of the squared errors $[p(t_k) - f_k]^2$ is as small as possible among all polynomials p of degree n,

$$\sum_{k=1}^{N} w_k[\hat{p}_n(t_k) - f_k]^2 \leq \sum_{k=1}^{N} w_k[p(t_k) - f_k]^2 \quad \text{for all } p \in \mathbb{P}_n. \tag{18.1}$$

Here, $w_k > 0$ are positive weights, which allow placing more emphasis on data points that are reliable, and less emphasis on others, by choosing them larger resp. smaller. If the quality of the data is uniformly the same, then equal weights, say $w_k = 1$, are appropriate.

The problem as formulated suggests a discrete N-point measure

$$\mathrm{d}\lambda_N(t) = \sum_{k=1}^{N} w_k \delta(t - t_k), \quad \delta = \text{Dirac delta function}, \tag{18.2}$$

in terms of which the problem can be written in the compact form

$$\|\hat{p}_n - f\|^2_{\mathrm{d}\lambda_N} \leq \|p - f\|^2_{\mathrm{d}\lambda_N} \quad \text{for all } p \in \mathbb{P}_n. \tag{18.3}$$

The polynomials $\pi_k(\,\cdot\,) = \pi_k(\,\cdot\,; \mathrm{d}\lambda_N)$ orthogonal (not necessarily monic) with respect to the discrete measure (18.2) provide an easy solution: one writes

$$p(t) = \sum_{i=0}^{n} c_i \pi_i(t), \quad n < N, \tag{18.4}$$

and obtains for the squared error, using the orthogonality of π_k,

(18.5)

$$\begin{aligned} E_n^2 &= \left(\sum_{i=0}^{n} c_i \pi_i - f, \sum_{j=0}^{n} c_j \pi_j - f \right) = \sum_{i,j=0}^{n} c_i c_j (\pi_i, \pi_j) - 2 \sum_{i=0}^{n} c_i (f, \pi_i) + \|f\|^2 \\ &= \sum_{i=0}^{n} \left(\|\pi_i\| c_i - \frac{(f, \pi_i)}{\|\pi_i\|} \right)^2 + \|f\|^2 - \sum_{i=0}^{n} \frac{(f, \pi_i)^2}{\|\pi_i\|^2}\,. \end{aligned}$$

(All norms and inner products are understood to be relative to the measure $\mathrm{d}\lambda_N$.) Evidently, the minimum is attained for $c_i = \hat{c}_i(f)$, where

$$\hat{c}_i(f) = \frac{(f, \pi_i)}{\|\pi_i\|^2}, \quad i = 0, 1, \ldots, n, \tag{18.6}$$

are the "Fourier coefficients" of f relative to the orthogonal system $\pi_0, \pi_1, \ldots, \pi_{N-1}$. Thus,

$$\hat{p}_n(t) = \sum_{i=0}^{n} \hat{c}_i(f)\pi_i(t; \mathrm{d}\lambda_N). \tag{18.7}$$

In Matlab, the procedure is implemented in the `OPQ` routine

```
[phat,c]=least_squares(n,f,xw,ab,d)
```

The given function values f_k are input through the `N`×1 array `f`, the abscissae t_k and weights w_k through the `N`×2 array `xw`, and the measure $\mathrm{d}\lambda_N$ through the (`N+1`)×2 array `ab` of recurrence coefficients. The 1×(`n+1`) array `d` is the vector of leading coefficients of the orthogonal polynomials. The procedure returns as output the `N`×(`n+1`) array `phat` of the values $\hat{p}_\nu(t_k)$, $0 \le \nu \le n$, $1 \le k \le N$, and the (`n+1`)×1 array `c` of the Fourier coefficients.

Example 11. Equally weighted least squares approximation on $N = 10$ equally spaced points on $[-1, 1]$.

Matlab program:

```
N=10; k=(1:N)'; d=ones(1,N);
xw(k,1)=-1+2*(k-1)/(N-1); xw(:,2)=2/N;
ab=r_hahn(N-1); ab(:,1)=-1+2*ab(:,1)/(N-1);
ab(:,2)=(2/(N-1))^2*ab(:,2); ab(1,2)=2;
[phat,c]=least_squares(N-1,f,xw,ab,d);
```

Demo#5 The program is applied to the function $f(t) = \ln(2+t)$ on $[-1, 1]$, and selected least squares errors $\hat{E}_n$ are compared in the table below with maximum errors E_n^∞ (taken over 100 equally spaced points on $[-1, 1]$).

n	$\hat{E}_n$	E_n^∞
0	4.88(–01)	6.37(–01)
3	2.96(–03)	3.49(–03)
6	2.07(–05)	7.06(–05)
9	1.74(–16)	3.44(–06)

If $n = N - 1$, the least squares error $\hat{E}_{N-1}$ is zero, since the N data points can be interpolated exactly by a polynomial of degree $\leq N - 1$. This is confirmed in the first tabular entry for $n = 9$. The infinity errors are only slightly larger than the least squares errors, except for $n = 9$.

18.2 Constrained least squares approximation

It is sometimes desirable to impose constraints on the least squares approximation, for example to insist that at certain points s_j the error should be exactly zero. Thus, the polynomial $p \in \mathbb{P}_n$ is subject to the constraints

$$p(s_j) = f_j, \quad j = 1, 2, \ldots, m; \quad m \leq n, \tag{18.8}$$

but otherwise is freely variable. For simplicity we assume that none of the s_j equals one of the support points t_k. (Otherwise, the procedure to be described requires some simple modifications.)

In order to solve the constrained least squares problem, let

$$p_m(f;t) = p_m(f; s_1, \ldots, s_m; t), \quad \sigma_m(t) = \prod_{j=1}^{m}(t - s_j) \tag{18.9}$$

be respectively the polynomial of degree $m-1$ interpolating f at the points s_j and the constraint polynomial of degree m. We then write

$$p(t) = p_m(f;t) + \sigma_m(t)q(t). \tag{18.10}$$

This clearly satisfies the constraints (18.8), and q is a polynomial of degree $n - m$ that can be freely varied. The problem is to minimize the squared error

$$\| f - p_m(f; \cdot) - \sigma_m q \|^2_{\mathrm{d}\lambda_N} = \int_{\mathbb{R}} \left[\frac{f(t) - p_m(f;t)}{\sigma_m(t)} - q(t) \right]^2 \sigma_m^2(t) \mathrm{d}\lambda_N$$

over all polynomials q of degree $n-m$. This is an *unconstrained* least squares problem, but for a new function f^* and a new measure $\mathrm{d}\lambda^*_N$,

$$\text{minimize :} \quad \| f^* - q \|_{\mathrm{d}\lambda^*_N}, \quad q \in \mathbb{P}_{n-m}, \tag{18.11}$$

where

$$f^*(t) = \frac{f(t) - p_m(f;t)}{\sigma_m(t)}, \quad \mathrm{d}\lambda^*_N(t) = \sigma_m^2(t)\mathrm{d}\lambda_N(t). \tag{18.12}$$

If $\hat{q}_{n-m}$ is the solution of (18.11), then

$$\hat{p}_n(t) = p_m(f;t) + \sigma_m(t)\hat{q}_{n-m}(t) \tag{18.13}$$

is the solution of the constrained least squares problem. The function f^*, incidentally, can be given the form of a divided difference,

$$f^*(t) = [s_1, s_2, \ldots, s_m, t\,]f, \quad t \in \operatorname{supp} \mathrm{d}\lambda_N^*,$$

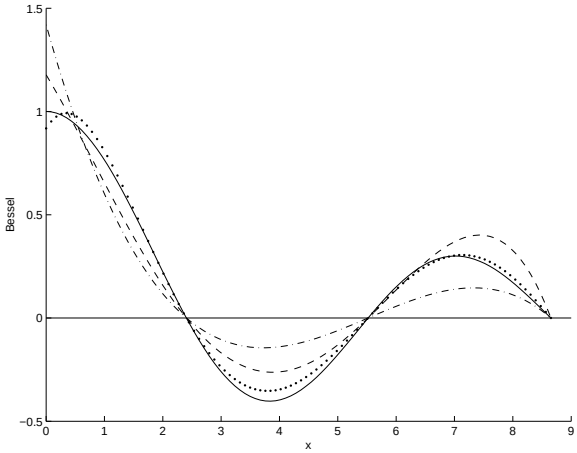
as follows from the theory of interpolation. Note also that the discrete orthogonal polynomials $\pi_k(\,\cdot\,;\mathrm{d}\lambda_N^*)$ needed to solve (18.11) can be obtained from the polynomials $\pi_k(\,\cdot\,;\mathrm{d}\lambda_N)$ by m modifications of the measure $\mathrm{d}\lambda_N(t)$ by linear factors $t - s_j$.

Example 12. Bessel function $J_0(t)$ for $0 \le t \le j_{0,3}$.

Here, $j_{0,3}$ is the third positive zero of J_0. A natural constraint is to reproduce the first three zeros of J_0 exactly, that is, $m = 3$ and

$$s_1 = j_{0,1}, \quad s_2 = j_{0,2}, \quad s_3 = j_{0.3}.$$

Demo#6 The constrained least squares approximations of degrees $n = 3, 4, 5$ (that is, $n - m = 0, 1, 2$) using $N = 51$ equally spaced points on $[0, j_{0,3}]$ (end points included) are shown in the figure below. The solid curve

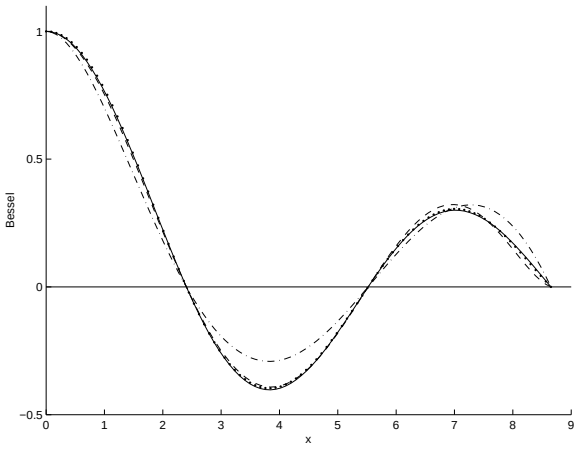


represents the exact function, the dashdotted, dashed, an dotted curves the approximants for $n = 3, 4$, and 5, respectively. The approximations are not particularly satisfactory and show spurious behavior near $t = 0$.

Example 13. Same as Example 12, but with two additional constraints

$$p(0) = 1, \quad p'(0) = 0.$$

Demo#7 Derivative constraints, as the one in Example 13, can be incorporated similarly as before. In this example, the added constraints are designed to remove the spurious behavior near $t = 0$; they also improve considerably the overall accuracy, as is shown in the next figure.



18.3 Least squares approximation in Sobolev spaces

The task now is to approximate simultaneously functions and some of their first derivatives. More precisely, we want to minimize

$$\sum_{\sigma=0}^{s} \sum_{k=1}^{N} w_k^{(\sigma)} [p^{(\sigma)}(t_k) - f_k^{(\sigma)}]^2$$

over all polynomials $p \in \mathbb{P}_n$, where $f_k^{(\sigma)}$, $\sigma = 0, 1, \ldots, s$, are given function and derivative values, and $w_k^{(\sigma)} > 0$ appropriate weights for each derivative. These are often chosen such that

$$w_k^{(\sigma)} = \gamma_\sigma w_k, \quad \gamma_\sigma > 0, \quad k = 1, 2, \ldots, N,$$

in terms of one set of positive weights w_k. Evidently,, the problem, analogously to (18.3), can be written in terms of the Sobolev inner product and

norm

$$(u,v)_S = \sum_{\sigma=0}^{s}\sum_{k=1}^{N} w_k^{(\sigma)} u^{(\sigma)}(t_k)v^{(\sigma)}(t_k), \quad \|u\|_S = \sqrt{(u,u)_S} \tag{18.14}$$

in the compact form

$$\text{minimize}: \quad \|p-f\|_S^2 \text{ for all } p \in \mathbb{P}_n. \tag{18.15}$$

The solution is entirely analogous to the one provided in §18.1,

$$\hat{p}_n(t) = \sum_{i=0}^{n} \hat{c}_i(f)\pi_i(t), \quad \hat{c}_i(f) = \frac{(f,\pi_i)_S}{\|\pi_i\|_S^2}, \tag{18.16}$$

where $\{\pi_i\}$ are the orthogonal polynomials of Sobolev type. In Matlab, the procedure is

```
[phat,c]=least_squares_sob(n,f,xw,B)
```

The input parameter `f` is now an `N`×$(s+1)$ array containing the `N` values of the given function and its first s derivatives at the points t_k. The abscissae t_k and the weights $w_k^{(\sigma)}$ of the Sobolev inner product are input via the `N`×$(s+1)$ array `xw`. (The routine determines s automatically from the size of the array `xw`.) The user also has to provide the `N`×`N` upper triangular array of the recurrence coefficients for the Sobolev orthogonal polynomials, which for $s = 1$ can be generated by the routine `chebyshev_sob.m` and for arbitrary s by the routine `stieltjes_sob.m`. The output `phat` is an array of dimension (`n+1`)×(`N`s) containing the `N` values of the derivative of order σ of the `n`th degree approximant $\hat{p}_n$ in positions (`n+1`,σ:$s+1$:`N`s) of the array `phat`. The Fourier coefficients $\hat{c}_i$ are output in the (`n+1`)×1 vector `c`.

Example 14. The complementary error function on $[0,2]$.

This is the function

$$f(t) = e^{t^2}\operatorname{erfc} t = \frac{2}{\sqrt{\pi}}\, e^{t^2}\int_t^\infty e^{-u^2}\,\mathrm{d}u, \quad 0 \le t \le 2,$$

whose derivatives are easily calculated.

Demo#8 The routine `least_squares_sob.m` is applied to the function f of Example 14 with $s = 2$ and `N`=5 equally spaced points t_k on $[0,2]$. All weights are chosen to be equal, $w_k^{(\sigma)} = 1/N$ for $\sigma = 0,1,2$. The table below, in the top half, shows selected results for the Sobolev least squares error $\hat{E}_n$

s	n	$\hat{E}_n$	$E_{n,0}^{\infty}$	$E_{n,1}^{\infty}$	$E_{n,2}^{\infty}$
2	0	1.153(+00)	4.759(–01)	1.128(+00)	2.000(+00)
	2	7.356(–01)	8.812(–02)	2.860(–01)	1.411(+00)
	4	1.196(–01)	1.810(–02)	5.434(–02)	1.960(–01)
	9	2.178(–05)	4.710(–06)	3.011(–05)	3.159(–04)
	14	3.653(–16)	1.130(–09)	1.111(–08)	1.966(–07)
0	0	2.674(–01)	4.759(–01)	1.128(+00)	2.000(+00)
	2	2.245(–02)	3.865(–02)	3.612(–01)	1.590(+00)
	4	1.053(–16)	3.516(–03)	5.160(–02)	4.956(–01)

and the maximum errors $E_{n,0}^{\infty}$, $E_{n,1}^{\infty}$, $E_{n,2}^{\infty}$ (over 100 equally spaced points on $[0, 2]$) for the function and its first two derivatives. In the bottom half are shown the analogous results for ordinary least squares approximation ($s = 0$) when $n \leq N - 1$. (It makes no sense to consider $n > N - 1$.) Note that the Sobolev least squares error $\hat{E}_{3N-1}$ is essentially zero, reflecting the fact that the Hermite interpolation polynomial of degree $3N - 1$ interpolates the data exactly. In contrast, $\hat{E}_n = 0$ for $n \geq N - 1$ in the case of ordinary least squares.

As expected, the table shows rather convincingly that Sobolev least squares approximation approximates the derivatives decidedly better than ordinary least squares approximation, when applicable, and even the function itself when n is sufficiently large.

19 Moment-preserving spline approximation

There are various types of approximation: those that control the maximum pointwise error; those that control some average error (like least squares error); and those, often motivated by physical considerations, that try to preserve the moments of the given function, or at least as many of the first moments as possible. It is this last type of approximation that we now wish to study. We begin with piecewise constant approximation on the whole real line $\mathbb{R}_+$, then proceed to spline approximation on $\mathbb{R}_+$, and end with spline approximation on a compact interval.

19.1 Piecewise constant approximation on $\mathbb{R}_+$

The piecewise constant approximants to be considered are

$$s_n(t) = \sum_{\nu=1}^{n} a_\nu H(t_\nu - t), \quad t \in \mathbb{R}_+, \tag{19.1}$$

where $a_\nu \in \mathbb{R}$, $0 < t_1 < t_2 < \cdots < t_n$, and H is the Heaviside function

$$H(u) = \begin{cases} 1 & \text{if } u \geq 0, \\ 0 & \text{otherwise.} \end{cases}$$

The problem is, for given $f \in C^1(\mathbb{R}_+)$, to find, if possible, the a_ν and t_ν such that

$$\int_0^\infty s_n(t) t^j \mathrm{d}t = \mu_j, \quad j = 0, 1, \ldots, 2n-1, \tag{19.2}$$

where

$$\mu_j = \int_0^\infty f(t) t^j \mathrm{d}t, \quad j = 0, 1, \ldots, 2n-1, \tag{19.3}$$

are the *moments* of f, assumed to exist.

The solution can be formulated in terms of Gauss quadrature relative to the measure

$$\mathrm{d}\lambda(t) = -t f'(t) \mathrm{d}t \quad \text{on } \mathbb{R}_+. \tag{19.4}$$

Indeed, if $f(t) = o(t^{-2n})$ as $t \to \infty$, then the problem has a unique solution if and only if $\mathrm{d}\lambda$ in (19.4) admits an n-point Gauss quadrature formula

$$\int_0^\infty g(t) \mathrm{d}\lambda(t) = \sum_{\nu=1}^{n} \lambda_\nu^G g(\tau_\nu^G), \quad g \in \mathbb{P}_{2n-1}, \tag{19.5}$$

satisfying $0 < \tau_1^G < \tau_2^G < \cdots < \tau_n^G$. If that is the case, then the desired knots t_ν and coefficients a_ν are given by

$$t_\nu = \tau_\nu^G, \quad a_\nu = \frac{\lambda_\nu^G}{\tau_\nu^G}, \quad \nu = 1, 2, \ldots, n. \tag{19.6}$$

A Gauss formula (19.5) always exists if $f' < 0$ on $\mathbb{R}_+$, that is, $\mathrm{d}\lambda(t) \geq 0$.

For the proof, we use integration by parts,

$$\int_0^T f(t)t^j \mathrm{d}t = \frac{1}{j+1}\, t^{j+1} f(t)\Big|_0^T - \frac{1}{j+1} \int_0^T f'(t)t^{j+1}\mathrm{d}t, \quad j \leq 2n-1,$$

and let $T \to \infty$. The integrated part on the right goes to zero by assumption on f, and the left-hand side converges to the jth moment of f, again by assumption. Therefore, the last term on the right also converges, and since $-tf'(t) = \mathrm{d}\lambda(t)$, one finds

$$\mu_j = \frac{1}{j+1} \int_0^\infty t^j \mathrm{d}\lambda(t), \quad j = 0, 1, \ldots, 2n-1.$$

This shows in particular that the first $2n$ moments of $\mathrm{d}\lambda$ exist, and therefore, if $\mathrm{d}\lambda \geq 0$, also the Gauss formula (19.5).

On the other hand, the approximant s_n has moments

$$\int_0^\infty s_n(t)t^j \mathrm{d}t = \sum_{\nu=1}^n a_\nu \int_0^{t_\nu} t^j \mathrm{d}t = \frac{1}{j+1} \sum_{\nu=1}^n a_\nu t_\nu^{j+1},$$

so that the first $2n$ moments μ_j of f are preserved if and only if

$$\sum_{\nu=1}^n (a_\nu t_\nu) t_\nu^j = \int_0^\infty t^j \mathrm{d}\lambda(t), \quad j = 0, 1, \ldots, 2n-1.$$

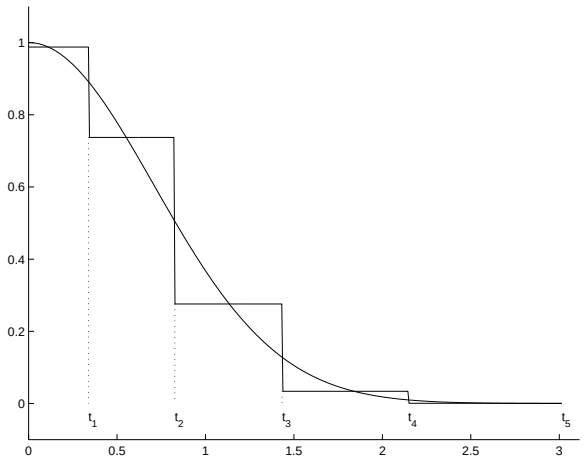
This is equivalent to saying that the knots t_ν are the Gauss nodes in (19.5), and $a_\nu t_\nu$ the corresponding weights.

Example 15. Maxwell distribution $f(t) = \mathrm{e}^{-t^2}$ on $\mathbb{R}_+$.

Here,

$$\mathrm{d}\lambda(t) = 2t^2 \mathrm{e}^{-t^2}\mathrm{d}t \quad \text{on } \mathbb{R}_+,$$

which is a positive measure obtained (up to the factor 2) by twice modifying the half-range Hermite measure by a linear factor t. The first $n+2$ recurrence coefficients of the half-range Hermite measure can be computed by a discretization method. Applying to these recurrence coefficients twice the routine `chri1.m`, with zero shift, then yields the recurrence coefficients $\alpha_k(\mathrm{d}\lambda)$, $\beta_k(\mathrm{d}\lambda)$, $k \leq n-1$, and hence the required n-point Gauss quadrature rule (19.5) for $\mathrm{d}\lambda$. The result for $n = 5$ is depicted in the figure below.



19.2 Spline approximation on $\mathbb{R}_+$

The approximant s_n of §19.1 can be interpreted as a spline function of degree 0. We now consider spline functions $s_{n,m}$ of degree $m > 0$,

$$s_{n,m}(t) = \sum_{\nu=1}^{n} a_\nu (t_\nu - t)_+^m, \quad t \in \mathbb{R}_+, \tag{19.7}$$

where u_+^m is the truncated power $u_+^m = u^m$ if $u \geq 0$, and $u_+^m = 0$ if $u < 0$. Given the first $2n$ moments (19.3) of f, the problem again is to determine $a_\nu \in \mathbb{R}$ and $0 < t_1 < t_2 < \cdots < t_n$ such that

$$\int_0^\infty s_{n,m}(t) t^j \mathrm{d}t = \mu_j, \quad j = 0, 1, \ldots, 2n-1. \tag{19.8}$$

By a reasoning similar to the one in §19.1, but more complicated, involving m integrations by part, one proves that for $f \in C^{m+1}(\mathbb{R}_+)$ and satisfying $f^{(\mu)}(t) = o(t^{-2n-\mu})$ as $t \to \infty$, $\mu = 0, 1, \ldots, m$, the problem has a unique solution if and only if the measure

$$\mathrm{d}\lambda^{[m]}(t) = \frac{(-1)^{m+1}}{m!} t^{m+1} f^{(m+1)}(t) \mathrm{d}t \quad \text{on } \mathbb{R}_+ \tag{19.9}$$

admits an n-point Gauss quadrature formula

$$\int_0^\infty g(t)\, \mathrm{d}\lambda^{[m]}(t) = \sum_{\nu=1}^{n} \lambda_\nu^G g(\tau_\nu^G) \quad \text{for all } g \in \mathbb{P}_{2n-1} \tag{19.10}$$

satisfying $0 < \tau_1^G < \tau_2^G < \cdots < \tau_n^G$. If that is the case, the knots t_ν and coefficients a_ν are given by

$$t_\nu = \tau_\nu^G, \quad a_\nu = \frac{\lambda_\nu^G}{[\tau_\nu^G]^{m+1}}, \quad \nu = 1, 2, \ldots, n. \tag{19.11}$$

Note that $\mathrm{d}\lambda^{[m]}$ in (19.9) is a positive measure, for each $m \geq 0$, and hence (19.10) exists, if f is completely monotonic on $\mathbb{R}_+$, that is, $(-1)^\mu f^{(\mu)}(t) > 0$, $t \in \mathbb{R}_+$, for $\mu = 0, 1, 2, \ldots$.

Example 16. Maxwell distribution $f(t) = \mathrm{e}^{-t^2}$ on $\mathbb{R}_+$, revisited.

We now have

$$\mathrm{d}\lambda^{[m]}(t) = \frac{1}{m!}\, t^{m+1} H_{m+1}(t) \mathrm{e}^{-t^2}\, \mathrm{d}t \quad \text{on } \mathbb{R}_+,$$

where H_{m+1} is the Hermite polynomial of degree $m+1$. Here, $\mathrm{d}\lambda^{[m]}$ if $m > 0$ is no longer of constant sign on $\mathbb{R}_+$, and hence the existence of the Gauss rule (19.10) is in doubt. Numerical exploration, using discretization methods, yields the situation shown in the table below, where a dash indicates the presence of a negative Gauss node τ_ν^G, and an asterisk the presence of a pair

n	$m=1$	$m=2$	$m=3$	n	$m=1$	$m=2$	$m=3$
1	6.9(–2)	1.8(–1)	2.6(–1)	11	—	1.1(–3)	1.1(–4)
2	8.2(–2)	—	2.3(–1)	12	—	—	*
3	—	1.1(–2)	2.5(–3)	13	7.8(–3)	6.7(–4)	*
4	3.5(–2)	6.7(–3)	2.2(–3)	14	8.3(–3)	5.6(–4)	8.1(–5)
5	2.6(–2)	—	1.6(–3)	15	7.7(–3)	—	7.1(–5)
6	2.2(–2)	3.1(–3)	*	16	—	4.9(–4)	7.8(–5)
7	—	2.4(–3)	*	17	—	3.8(–4)	3.8(–5)
8	1.4(–2)	—	3.4(–4)	18	5.5(–3)	3.8(–4)	*
9	1.1(–2)	1.7(–3)	2.5(–4)	19	5.3(–3)	—	*
10	9.0(–3)	1.1(–3)	—	20	5.4(–3)	3.1(–4)	*

of conjugate complex Gauss nodes. In all cases computed, there were never more than one negative Gauss node, or more than one pair of complex nodes. The numbers in the table represent the maximum errors $\|s_{n,m} - f\|_\infty$, the maximum being taken over 100 equally spaced points on $[0, \tau_n^G]$.

19.3 Spline approximation on a compact interval

The problem on a compact interval, say $[0,1]$, is a bit more involved than the problem on $\mathbb{R}_+$. For one, the spline function $s_{n,m}$ may now include a polynomial p of degree m, which was absent before since no moment of p exists on $\mathbb{R}_+$ unless $p \equiv 0$. Thus, the spline approximant has now the form

$$s_{n,m}(t) = p(t) + \sum_{\nu=1}^{n} a_\nu (t_\nu - t)_+^m, \quad p \in \mathbb{P}_m, \quad 0 \le t \le 1, \tag{19.12}$$

where $a_\nu \in \mathbb{R}$ and $0 < t_1 < t_2 < \cdots < t_n < 1$. There are two problems of interest:

Problem I. Find $s_{n,m}$ such that

$$\int_0^1 s_{n,m}(t) t^j \mathrm{d}t = \mu_j, \quad j = 0, 1, \ldots, 2n+m. \tag{19.13}$$

Since we have $m+1$ additional parameters at our disposal (the coefficients of p), we can impose $m+1$ additionl moment conditions.

Problem II. Rather than matching more moments, we use the added degree of freedom to impose $m+1$ "boundary conditions" at the end point $t = 1$. More precisely, we want to find $s_{n,m}$ such that

$$\int_0^1 s_{n,m}(t) t^j \mathrm{d}t = \mu_j, \quad j = 0, 1, \ldots, 2n-1 \tag{19.14}$$

and

$$s_{n,m}^{(\mu)}(1) = f^{(\mu)}(1), \quad \mu = 0, 1, \ldots, m. \tag{19.15}$$

It is still true that a solution can be given in terms of quadrature formulae, but they are now respectively generalized Gauss–Lobatto and generalized Gauss–Radau formulae relative to the measure[3]

$$\mathrm{d}\lambda^{[m]}(t) = \frac{(-1)^{m+1}}{m!} f^{(m+1)}(t) \mathrm{d}t \quad \text{on } [0,1]. \tag{19.16}$$

[3]See M. Frontini, W. Gautschi, and G.V. Milovanović, "Moment-preserving spline approximation on finite intervals", Numer. Math. 50 (1987), 503–518; also M. Frontini, W. Gautschi, G.V. Milovanović, "Moment-preserving spline approximation on finite intervals and Turán quadratures", Facta Univ. Ser. Math. Inform. 4 (1989), 45–56.

Problem I, in fact, has a unique solution if and only if the *generalized Gauss–Lobatto formula*

$$\int_0^1 g(t)\mathrm{d}\lambda^{[m]}(t) = \sum_{\mu=0}^{m}[\lambda_0^{(\mu)}g^{(\mu)}(0) + (-1)^\mu\lambda_{n+1}^{(\mu)}g^{(\mu)}(1)] + \sum_{\nu=1}^{n}\lambda_\nu^L g(\tau_\nu^L), \quad g \in \mathbb{P}_{2n+2m+1}, \tag{19.17}$$

exists with $0 < \tau_1^L < \cdots < \tau_n^L < 1$. In this case,

$$t_\nu = \tau_\nu^L, \quad a_\nu = \lambda_\nu^L, \quad \nu = 1, 2, \ldots . n, \tag{19.18}$$

and p is uniquely determined by

$$p^{(\mu)}(1) = f^{(\mu)}(1) + (-1)^m m!\lambda_{n+1}^{(m-\mu)}, \quad \mu = 0, 1, \ldots, m. \tag{19.19}$$

Similarly, Problem II has a unique solution if and only if the *generalized Gauss–Radau formula*

$$\int_0^1 g(t)\mathrm{d}\lambda^{[m]}(t) = \sum_{\mu=0}^{m}\lambda_0^{(\mu)}g^{(\mu)}(0) + \sum_{\nu=1}^{n}\lambda_\nu^R g(\tau_\nu^R), \quad g \in \mathbb{P}_{2n+m}, \tag{19.20}$$

exists with $0 < \tau_1^R < \cdots < \tau_n^R < 1$. Then

$$t_\nu = \tau_\nu^R, \quad a_\nu = \lambda_\nu^R, \quad \nu = 1, 2, \ldots . n, \tag{19.21}$$

and (trivially)

$$p(t) = \sum_{\mu=0}^{m}\frac{f^{(\mu)}(1)}{\mu!}(t-1)^\mu. \tag{19.22}$$

In both cases, complete monotonicity of f implies $\mathrm{d}\lambda \geq 0$ and the existence of the respective quadrature formulae. For their construction, see Exercises 12 and 13 of Part II.

20 Slowly convergent series

Standard techniques of accelerating the convergence of slowly convergent series are based on linear or nonlinear sequence transformations: the sequence

of partial sums is transformed somehow into a new sequence that converges to the same limit, but a lot faster. Here we folllow another approach, more in the spirit of these lectures: the sum of the series is represented as a definite integral; a sequence of quadrature rules is then applied to this integral which, when properly chosen, will produce a sequence of approximations that converges quickly to the desired sum.

An easy way (and certainly not the only one) to obtain an inegral representation presents itself when the general term of the series, or part thereof, is expressible in terms of the Laplace transform (or some other integral transform) of a known function. Several instances of this will now be described.

20.1 Series generated by a Laplace transform

The series

$$S = \sum_{k=1}^{\infty} a_k \tag{20.1}$$

to be considered first has terms a_k that are the Laplace transform

$$(\mathcal{L}f)(s) = \int_0^{\infty} \mathrm{e}^{-st} f(t)\mathrm{d}t$$

of some known function f evaluated at $s = k$,

$$a_k = (\mathcal{L}f)(k), \quad k = 1, 2, 3, \ldots . \tag{20.2}$$

In this case,

$$\begin{aligned} S &= \sum_{k=1}^{\infty} \int_0^{\infty} \mathrm{e}^{-kt} f(t)\mathrm{d}t \\ &= \int_0^{\infty} \sum_{k=1}^{\infty} \mathrm{e}^{-(k-1)t} \cdot \mathrm{e}^{-t} f(t)\mathrm{d}t \\ &= \int_0^{\infty} \frac{1}{1-\mathrm{e}^{-t}}\, \mathrm{e}^{-t} f(t)\mathrm{d}t \end{aligned}$$

that is,

$$S = \int_0^{\infty} \frac{t}{1-\mathrm{e}^{-t}}\, \frac{f(t)}{t}\, \mathrm{e}^{-t}\mathrm{d}t. \tag{20.3}$$

There are at least three different approaches to evaluate this integral numerically: one is Gauss–Laguerre quadrature of $(t/(1-\mathrm{e}^{-t}))f(t)/t$ with $\mathrm{d}\lambda(t)=\mathrm{e}^{-t}\mathrm{d}t$ on $\mathbb{R}_+$; another is rational/polynomial Gauss–Laguerre quadrature of the same function; and a third Gauss–Einstein quadrature of the function $f(t)/t$ with $\mathrm{d}\lambda(t)=t\mathrm{d}t/(\mathrm{e}^t-1)$ on $\mathbb{R}_+$. In the last method, the weight function $t/(\mathrm{e}^t-1)$ is widely used in solid state physics, where it is named after Einstein (coming from the Einstein-Bose distribution). It is also, incidentally, the generating function of the Bernoulli polynomials.

Example 17. The Theodorus constant

$$S=\sum_{k=1}^{\infty}\frac{1}{k^{3/2}+k^{1/2}}=1.860025\ldots\,.$$

This is a universal constant introduced by P.J. Davis (1993) in connection with a spiral attributed to the ancient mathematician Theodorus of Cyrene.

Here we note that

$$\frac{1}{s^{3/2}+s^{1/2}}=s^{-1/2}\frac{1}{s+1}=\left(\mathcal{L}\frac{1}{\sqrt{\pi t}}*\mathrm{e}^{-t}\right)(s),$$

where the star stands for convolution. A simple computation yields (20.2) with

$$f(t)=\frac{2}{\sqrt{\pi}}\,F(\sqrt{t}),$$

where

$$F(x)=\mathrm{e}^{-x^2}\int_0^x\mathrm{e}^{t^2}\mathrm{d}t$$

is Dawson's integral.

Demo#9 To make $f(t)$ regular at $t=0$, we divide by $\sqrt{t}$ and write

$$\begin{aligned}S&=\frac{2}{\sqrt{\pi}}\int_0^{\infty}\frac{t}{1-\mathrm{e}^{-t}}\,\frac{F(\sqrt{t})}{\sqrt{t}}\,t^{-1/2}\mathrm{e}^{-t}\mathrm{d}t\\&=\frac{2}{\sqrt{\pi}}\int_0^{\infty}\frac{F(\sqrt{t})}{\sqrt{t}}\,t^{-1/2}\frac{t}{\mathrm{e}^t-1}\,\mathrm{d}t.\end{aligned}$$

To the first integral we apply Gauss–Laguerre quadrature with $\mathrm{d}\lambda(t)=t^{-1/2}\mathrm{e}^{-t}\mathrm{d}t$ on $\mathbb{R}_+$, or rational Gauss–Laguerre with the same $\mathrm{d}\lambda$, and to the second integral Gauss–Einstein quadrature (modified by the factor $t^{-1/2}$). The errors committed in these quadrature methods are shown in the table below.

n	Gauss-Laguerre	rational Gauss-Laguerre	Gauss-Einstein
1	9.6799(–03)	1.5635(–02)	1.3610(–01)
4	5.5952(–06)	1.1893(–08)	2.1735(–04)
7	4.0004(–08)	5.9689(–16)	3.3459(–07)
10	5.9256(–10)		5.0254(–10)
15	8.2683(–12)		9.4308(–15)
20	8.9175(–14)		4.7751(–16)
	timing: 10.8	timing: 8.78	timing: 10.4

The clear winner is rational Gauss–Laguerre, both in terms of accuracy and run time.

Example 18. The Hardy–Littlewood function

$$H(x) = \sum_{k=1}^{\infty} \frac{1}{k} \sin \frac{x}{k}, \quad x > 0.$$

It can be shown that

$$a_k := \frac{1}{k} \sin \frac{x}{k} = (\mathcal{L} f(t; x))(k),$$

where

$$f(t; x) = \frac{1}{2\mathrm{i}} \left[I_0(2\sqrt{\mathrm{i}xt}) - I_0(2\sqrt{-\mathrm{i}xt}) \right]$$

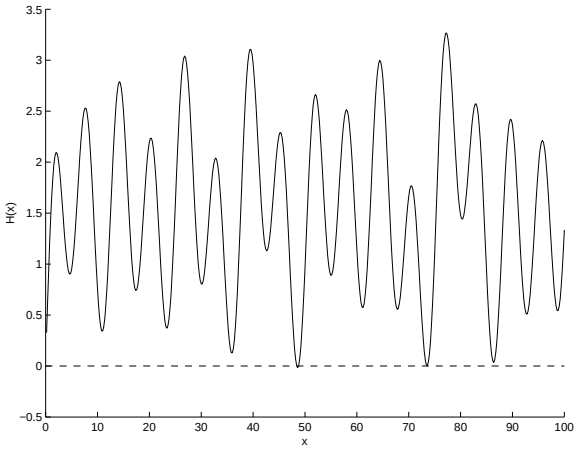
and I_0 is the modified Bessel function. This gives rise to the two integral representations

$$H(x) = \int_0^{\infty} \frac{t}{1 - \mathrm{e}^{-t}} \frac{f(t; x)}{t} \mathrm{e}^{-t} \mathrm{d}t = \int_0^{\infty} \frac{f(t; x)}{t} \frac{t}{\mathrm{e}^t - 1} \mathrm{d}t.$$

Among the three quadrature methods, Gauss–Einstein performs best, but all suffer from internal cancellation of terms in the quadrature sum. The problem becomes more prominent as the number n of terms increases. In this case, other methods can be applied, using the Euler-Maclaurin formula.[4]

The figure below shows the behavior of $H(x)$ in the range $0 \le x \le 100$.

[4]See W. Gautschi, "The Hardy-Littlewood function: an exercise in slowly convergent series", J. Comput. Appl. Math., to appear.



20.2 "Alternating" series generated by a Laplace transform

These are series in which the general terms are Laplace transforms with alternating signs of some function f, that is, series (20.1) with

$$a_k = (-1)^{k-1}(\mathcal{L}\, f)(k), \quad k = 1, 2, 3, \ldots . \tag{20.4}$$

An elementary computation similar to the one carried out in §20.1 will show that

$$S = \int_0^\infty \frac{1}{1+\mathrm{e}^{-t}}\, f(t)\mathrm{e}^{-t}\mathrm{d}t = \int_0^\infty f(t)\frac{1}{\mathrm{e}^t+1}\,\mathrm{d}t. \tag{20.5}$$

We can again choose between three quadrature methods: Gauss–Laguerre quadrature of the function $f(t)/(1+\mathrm{e}^{-t})$ with $\mathrm{d}\lambda(t) = \mathrm{e}^{-t}\mathrm{d}t$, rational/polynomial Gauss–Laguerre of the same function, and Gauss-Fermi quadrature of $f(t)$ with $\mathrm{d}\lambda(t) = \mathrm{d}t/(\mathrm{e}^t+1)$ involving the Fermi function $1/(\mathrm{e}^t+1)$ (also used in solid state physics).

Example 19. The series

$$S = \sum_{k=1}^{\infty} \frac{(-1)^{k-1}}{k}\,\mathrm{e}^{-1/k}.$$

One can show that the function f in question here is $f(t) = J_0(2\sqrt{t})$, with J_0 the Bessel function of order zero. Errors obtained by the three quadrature methods are displayed in the table below, showing the clear superiority of Gauss–Fermi quadrature.

n	Gauss-Laguerre	rational Gauss-Laguerre	Gauss-Fermi
1	1.6961(–01)	1.0310(–01)	5.6994(–01)
4	4.4754(–03)	4.6605(–05)	9.6454(–07)
7	1.7468(–04)	1.8274(–09)	9.1529(–15)
10	3.7891(–06)	1.5729(–13)	2.8163(–16)
15	2.6569(–07)	1.5490(–15)	
20	8.6155(–09)		
40	1.8066(–13)		
	timing: 12.7	timing: 19.5	timing: 4.95

20.3 Series generated by the derivative of a Laplace transform

These are series (20.1) in which

$$a_k = -\frac{\mathrm{d}}{\mathrm{d}s}(\mathcal{L}f)(s)\bigg|_{s=k}, \quad k = 1, 2, 3, \ldots . \tag{20.6}$$

In this case one finds

$$S = \int_0^\infty \frac{t}{1 - \mathrm{e}^{-t}}\, f(t)\mathrm{e}^{-t}\mathrm{d}t = \int_0^\infty f(t)\frac{t}{\mathrm{e}^t - 1}\,\mathrm{d}t, \tag{20.7}$$

and Gauss–Laguerre, rational/polynomial Gauss–Laguerre, and Gauss–Einstein quadrature are again options as in §20.1.

Example 20. The series

$$S = \sum_{k=1}^{\infty} (\tfrac{3}{2} + 1)k^{-2}(k+1)^{-3/2}.$$

The relevant function f is calculated to be

$$f(t) = \frac{\operatorname{erf}\sqrt{t}}{\sqrt{t}} \cdot t^{1/2},$$

where erf is the error function $\operatorname{erf} x = (2/\sqrt{\pi}) \int_0^x \mathrm{e}^{-t^2}\mathrm{d}t$. Numerical results analogous to those in the two previous tables are shown below.

n	Gauss-Laguerre	rational Gauss-Laguerre	Gauss-Einstein
1	4.0125(–03)	5.1071(–02)	8.1715(–02)
4	1.5108(–05)	4.5309(–08)	1.6872(–04)
7	4.6576(–08)	1.3226(–13)	3.1571(–07)
10	3.0433(–09)	1.2087(–15)	5.4661(–10)
15	4.3126(–11)		1.2605(–14)
20	7.6664(–14)		
30	3.4533(–16)		
	timing: 6.50	timing: 10.8	timing: 1.58

The run time is best for Gauss–Einstein quadrature, though the error is worse than for the closest competitor, rational Gauss–Laguerre.

20.4 Slowly convergent series occurring in plate contact problems

The series of interest here is

$$R_p(z) = \sum_{k=0}^{\infty} \frac{z^{2k+1}}{(2k+1)^p}, \quad z \in \mathbb{C},\ |z| \le 1,\ p = 2 \text{ or } 3. \tag{20.8}$$

Rather than expressing the whole general term of the series as a Laplace transform, we do this only for the coefficient,

$$\frac{1}{(k+\frac{1}{2})^p} = (\mathcal{L}f)(k), \quad f(t) = \frac{1}{(p-1)!}\, t^{p-1}\mathrm{e}^{-t/2}. \tag{20.9}$$

Then

$$\begin{aligned} R_p(z) &= \frac{z}{2^p} \sum_{k=0}^{\infty} \frac{z^{2k}}{(k+\frac{1}{2})^p} \\ &= \frac{z}{2^p} \sum_{k=0}^{\infty} z^{2k} \int_0^{\infty} \mathrm{e}^{-kt} \cdot \frac{t^{p-1}\mathrm{e}^{-t/2}}{(p-1)!}\, \mathrm{d}t \\ &= \frac{z}{2^p(p-1)!} \int_0^{\infty} \sum_{k=0}^{\infty} (z^2\mathrm{e}^{-t})^k \cdot t^{p-1}\mathrm{e}^{-t/2}\mathrm{d}t \\ &= \frac{z}{2^p(p-1)!} \int_0^{\infty} \frac{1}{1-z^2\mathrm{e}^{-t}}\, t^{p-1}\mathrm{e}^{-t/2}\mathrm{d}t, \end{aligned}$$

that is,

$$R_p(z) = \frac{z}{2^p(p-1)!} \int_0^{\infty} \frac{t^{p-1}\mathrm{e}^{t/2}}{\mathrm{e}^t - z^2}\, \mathrm{d}t, \quad z^{-2} \in \mathbb{C}\backslash[0,1]. \tag{20.10}$$

The case $z = 1$ can be treated directly by using the connection with the zeta function, $R_p(1) = (1 - 2^{-p})\zeta(p)$. Assume therefore $z \neq 1$. When $|z|$ is close to 1, the integrand in (20.10) is rather ill-behaved near $t = 0$, exhibiting a steep boundary layer. We try to circumvent this by making the change of variables $\mathrm{e}^{-t} \mapsto t$ to obtain

$$R_p(z) = \frac{1}{2^p(p-1)!z} \int_0^1 \frac{t^{-1/2}[\ln(1/t)]^{p-1}}{z^{-2} - t}\, \mathrm{d}t.$$

This expresses $R_p(z)$ as a Cauchy integral of the measure

$$\mathrm{d}\lambda^{[p]}(t) = t^{-1/2}[\ln(1/t)]^{p-1}\mathrm{d}t.$$

Since by assumption, z^{-2} lies outside the interval $[0, 1]$, the integral can be evaluated by the continued fraction algorithm, once sufficiently many recurrence coefficients for $\mathrm{d}\lambda^{[p]}$ have been precomputed. For the latter, the modified Chebyshev algorithm is quite effective. The first 100 coefficients are available for $p = 2$ and $p = 3$ in the `OPQ` files `absqm1log1` and `absqm1log2` to 25 resp. 20 decimal digits.

Example 21.

$$R_p(x),\ p = 2 \text{ and } 3,\ x = .8, .9, .95, .99, .999 \text{ and } 1.0.$$

Numerical results are shown in he table below and are accurate to all digits

x	$R_2(x)$	$R_3(x)$
.8	0.87728809392147	0.82248858052014
.9	1.02593895111111	0.93414857586540
.95	1.11409957792905	0.99191543992243
.99	1.20207566477686	1.03957223187364
.999	1.22939819733	1.05056774973
1.000	1.233625	1.051795

shown. Full acuracy cannot be achieved for $x \geq .999$ using only 100 recurrence coefficients of $\mathrm{d}\lambda^{[p]}$.

Example 22.

$$R_p(e^{\mathrm{i}\alpha}),\ p = 2 \text{ and } 3,\ \alpha = \omega\pi/2,\ \omega = .2, .1, .05, .01, .001 \text{ and } 0.0.$$

Numerical results are shown in the table below.

p	ω	$\mathrm{Re}(R_p(z))$	$\mathrm{Im}(R_p(z))$
2	.2	0.98696044010894	0.44740227008596
3		0.96915102126252	0.34882061265337
2	0.1	1.11033049512255	0.27830297928558
3		1.02685555765937	0.18409976778928
2	0.05	1.17201552262936	0.16639152396897
3		1.04449441539672	0.09447224926029
2	0.01	1.22136354463481	0.04592009281744
3		1.05140829197388	0.01928202831056
2	0.001	1.232466849	0.006400460
3		1.051794454	0.001936923
2	0.000	1.2336	0.0000
3		1.0518	0.0000

Here, too, full accuracy is not attainable for $\omega \leq 0.001$ with only 100 recurrence coefficients. Curiously, the continued fraction algorithm seems to converge also when $z = 1$, albeit slowly.

20.5 Series involving ratios of hyperbolic functions

More of a challenge are series of the type

$$T_p(x;b) = \sum_{k=0}^{\infty} \frac{1}{(2k+1)^p} \frac{\cosh(2k+1)k}{\cosh(2k+1)b}, \quad 0 \leq x \leq b,\ b > 0,\ p = 2, 3, \tag{20.11}$$

which also occur in plate contact problems. Here, we first expand the ratio of hyperbolic cosines into an infinite series,

$$\begin{aligned} &\frac{\cosh(2k+1)x}{\cosh(2k+1)b} \\ &\qquad = \sum_{n=0}^{\infty} (-1)^n \left\{ \mathrm{e}^{-(2k+1)[(2n+1)b-x]} + \mathrm{e}^{-(2k+1)[(2n+1)b+x]} \right\}, \end{aligned} \tag{20.12}$$

insert this in (20.11) and apply the Laplace transform technique of §20.4. This yields, after an elementary computation (using an interchange of the summations over k and n),

$$T_p(x,b) = \frac{1}{2^p (p-1)!} \sum_{n=0}^{\infty} (-1)^n \mathrm{e}^{(2n+1)b} [\varphi_n(-x) + \varphi_n(x)], \tag{20.13}$$

where

$$\varphi_n(s) = \mathrm{e}^s \int_0^1 \frac{\mathrm{d}\lambda^{[p]}(t)}{\mathrm{e}^{2[(2n+1)b+s]} - t}\,, \quad -b \le s \le b. \tag{20.14}$$

The integral on the right is again amenable to the continued fraction algorithm for $\mathrm{d}\lambda^{[p]}$, which for large n converges almost instantaneously. Convergence of the series (20.13) is geometric with ratio e^{-b}.

Exercises to Part III (Stars indicate more advanced exercises.)

1. With $\pi_0, \pi_1, \ldots, \pi_{N-1}$ denoting the discrete orthogonal polynomials relative to the measure $\mathrm{d}\lambda_N$, and $\hat{c}_i(f)$ the Fourier coefficients of f with respect to these orthogonal polynomials, show that

$$\sum_{i=0}^{n} |\hat{c}_i(f)|^2 \|\pi_i\|^2 \leq \|f\|^2, \quad n < N,$$

with equality holding for $n = N - 1$.

2. Prove the following alternative form for the Fourier coefficients,

$$\hat{c}_i(f) = \frac{1}{\|\pi_i\|^2} \left(f - \sum_{j=0}^{i-1} \hat{c}_j(f)\pi_j, \pi_i \right), \quad i = 0, 1, \ldots, n,$$

and discuss its possible advantages over the original form.

3. Discuss the modifications required in the constrained least squares approximation when ν $(0 \leq \nu \leq m)$ of the points s_j are equal to one of the support points t_k.

4. What are $p_m(f; \cdot)$, f^*, and σ_m in Example 12?

5. Calculate the first and second derivative of the complementary error function of Example 14.

6*. Prove the unique solvability of the problem (19.8) under the conditions stated in (19.9)–(19.10), and, in the affirmative case, derive (19.11).

7. Derive the measure $\mathrm{d}\lambda^{[m]}$ for the Maxwell distribution of Example 16.

8. Derive the formula for f in Example 17.

9. Derive the formula for f in Example 18.

10. Derive (20.5).

11. Derive the formula for f in Example 19.

12. Derive (20.7).

13. Derive the formula for f in Example 20.

14. Supply the details for deriving (20.13).