Math 563 Lecture Notes
Approximation with orthogonal bases

Spring 2020

The point: The framework for approximation by orthogonal bases ('generalized Fourier series') is set up - the appropriate spaces, general idea and examples using polynomials. The Chebyshev polynomials, which have an important connection to Fourier series, are a notable example to be revisited soon. The structure here is the foundation for essential methods in numerical analysis - Gaussian quadrature (efficient integration), Fourier series (efficient approximation) and more.

Related reading: Details on orthogonal polynomials can be found in Quarteroni, 10.1.

1 General theory

We now consider the problem of **continuous approximation**. For a space of functions on $[a, b]$, we seek a basis $\{\phi_j\}$ such that the first n can be used to approximate the function. Let $w(x) > 0$ be a positive function. For complex-valued functions on $[a, b]$, define the 'weighted' L^2 norm[1]

$$\|f\|_w = \left(\int_a^b w(x)|f(x)|^2 \, dx \right)^{1/2}$$

which has an associated (complex) inner product

$$\langle f, g \rangle_w = \int_a^b f(x)\overline{g(x)}w(x) \, dx. \tag{1}$$

Note that in the case of real functions, the overbar (complex conjugate) can be dropped). We consider approximating functions in the 'weighted L^2' space

$$L^2_w([a, b]) = \{f : [a, b] \to \mathbb{C} \text{ s.t. } \int_a^b |f(x)|^2 w(x) \, dx < \infty\}$$

which includes, in practice, essentially any function of interest. The norm and inner product (1) is well-defined on this space.

[1] Technically, $\|f\|_{2,w}$ or something similar should be written to distinguish from other L^p norms, but the 2 here is implied; the subscript may be dropped entirely if the context is clear.

A **basis** $\{\phi_j\}$ $(j = 1, 2, \cdots)$ for $L^2_w([a,b])$ is a set of functions such that any f in the space can be expressed uniquely as

$$f = \sum_{j=1}^{\infty} c_j \phi_j$$

in the sense that the partial sums $\sum_{j=1}^{N} c_j \phi_j$ converge in norm, i.e.

$$\lim_{N\to\infty} \|f - \sum_{j=1}^{N} c_j \phi_j\|_w = 0.$$

A set of functions $\{f_j\}$ is called **orthogonal** if $\langle f_i, f_j\rangle = 0$ for $i \neq j$.

1.1 Orthogonality

Orthogonal bases are particularly nice, both for theory and numerical approximation. Suppose $\{\phi_j\}$ is an orthogonal basis for $L^2_w([a,b])$ - that is, a basis where $\langle \phi_i, \phi_j\rangle = 0$ for $i \neq j$.

Coefficients: The coeffiicents c_j in the representation

$$f = \sum_{j} c_j \phi_j$$

are easily found by taking the inner product with a basis function to select that component:

$$\langle f, \phi_k\rangle = \sum_{j} c_j \langle \phi_j, \phi_k\rangle \implies c_k = \frac{\langle f, \phi_k\rangle}{\langle \phi_k, \phi_k\rangle}.$$

Best approximation property: The approximation

$$f_N = \sum_{j=1}^{N} c_j \phi_j = \text{first } N \text{ terms of the series for } f$$

is the best approximation to f in the subspace

$$S_N = \text{span}(\phi_1, \cdots, \phi_N)$$

in the sense that

$$g = f_N \text{ minimizes } \|f - g\|_w \text{ for } g \in S_N.$$

Error: We also have have **Parseval's theorem**

$$\|f - \sum_{j=1}^{N} c_j \phi_j\|_w^2 = \sum_{j=N+1}^{\infty} c_j^2 \|\phi_j\|_w^2.$$

Formally, this is proven by writing $\|g\|^2 = \langle g, g\rangle$ and distributing the inner product:

$$\begin{aligned}\|f_N\|^2 &= \langle f - f_N, f - f_N\rangle \\ &= \langle \sum_{j=N+1}^{\infty} c_j\phi_j, \sum_{k=N+1}^{\infty} c_k\phi_k\rangle \\ &= \sum_{j=N+1}^{\infty}\sum_{k=N+1}^{\infty} c_j c_k \langle \phi_j, \phi_k\rangle.\end{aligned}$$

But the inner product is non-zero only when $j = k$, which yields the result (to be rigorous, one has to do some work to prove convergence). Then the 'error' in the N-th approximation is then the sum of the squares of the norms of the omitted terms, e.g. if $c_j \sim C/j^2$ then the error looks like $\sum_{j=N+1}^{\infty} C/j^4 \sim C/N^3$.

1.2 (Continuous) least squares

The properties listed above suggest we can use orthogonal bases to construct good approximations. Suppose $\{\phi_j\}$ is a basis (not necessarily orthogonal) and

$$f \approx \sum_{j=1}^{N} c_j\phi_j \text{ minimizes } \|f - g\|_w \text{ over } S_N.$$

Then the c_j's minimize the L^2 error

$$E(\mathbf{c}) = \|f - \sum_{j=1}^{N} c_j\phi_j\|^2 = \int_a^b (f - \sum_{j=1}^{N} c_j\phi_j)^2 w(x)\, dx.$$

To find the coefficients, note that the minimum occurs at a point where the gradient of E is zero, so the conditions for a critical point are

$$0 = \frac{\partial E}{\partial c_i} = -2\int_a^b (f - \sum_{j=1}^{n} c_j\phi_j)\phi_i w(x)\, dx.$$

It follows that E is minimized when

$$\int_a^b f(x)\phi_i(x)\, dx = \sum_{j=1}^{n}\left(\int_a^b \phi_i\phi_j w(x)\, dx\right) c_j, \quad i = 1, \cdots n.$$

In matrix form, this is a linear system

$$A\mathbf{c} = \mathbf{f}, \quad a_{ij} = \int_a^b \phi_i\phi_j w(x)\, dx = \langle \phi_i, \phi_j\rangle_w, \quad f_i = \int_a^b f(x)\phi_i(x)w(x)\, dx = \langle f, \phi_i\rangle_w.$$

For numerical stability and computational efficiency, we want the matrix A to be as simple as possible. If the basis is **orthogonal** then the equations reduce to a diagonal system since $\langle \phi_i, \phi_j\rangle = 0$ for $i \neq j$ and the solution is just

$$c_i = \frac{\langle f, \phi_i\rangle_w}{\langle \phi_i, \phi_i\rangle_w}$$

Exploiting the structure of the orthogonal basis is our starting point for building good numerical approximations.

1.3 The Gram-schmidt process

Suppose we have a basis $\{f_j\}$ of functions and wish to convert it into an orthogonal basis $\{\phi_j\}$. The **Gram-Schmidt** process does so, ensuring that

$$\phi_j \in \text{span}(f_0, \cdots, f_j).$$

The process is simple: take f_j as the 'starting' function, then subtract off the components of f_j in the direction of the previous ϕ's, so that the result is orthogonal to them. That is, we compute the sequence

$$\begin{aligned}
\phi_0 &= f_0 \\
\phi_1 &= f_1 - \frac{\langle f_1, \phi_0 \rangle}{\langle \phi_0, \phi_0 \rangle} \phi_0 \\
\vdots &= \vdots \\
\phi_j &= f_j - \sum_{k=0}^{j-1} \frac{\langle f_j, \phi_k \rangle}{\langle \phi_k, \phi_k \rangle} \phi_k.
\end{aligned}$$

It is easy to verify that this procedure generates the desired orthogonal basis.

More generally, we can take the 'starting' function f_j to be any function not in the span of $\phi_0, \cdots, \phi_{j-1}$, so it can be chosen to be whatever is most convenient at this step.

Caution (normalization): The norm-squared

$$\|\phi_j\|^2 = \langle \phi_j, \phi_j \rangle$$

can be freely chosen once the basis is constructed by scaling the ϕ's (normalization). For common sets of orthogonal functions, there can be more than one 'standard' choice of normalization (e.g. $\|\phi_j\| = 1$), so one should be careful when using such references.

1.4 The 'three-term' recurrence

When considering polynomial basis, we can simplify the process. We seek an orthogonal basis $\{\phi_j\}$ such that ϕ_j is a polynomial of degree j so that

$$\text{span}(\phi_0, \cdots, \phi_j) = \mathbb{P}_j. \tag{2}$$

One could use the starting basis $1, x, x^2, \cdots$ and then apply Gram-Schmidt.

However, a more judicious choice lets us remove most of the terms in the formula. Suppose $\phi_0, \cdots, \phi_j$ have been constructed with the property (2). Then

$$\begin{aligned}
&\phi_j \text{ is orthogonal to } \phi_0, \cdots, \phi_{j-1} \\
&\quad \implies \phi_j \text{ is orthogonal to } \mathbb{P}_{j-1}.
\end{aligned}$$

Now we take $x\phi_j$ as the starting function for the next basis function. This function is a polynomial of degree $n+1$ and

$$\langle x\phi_j, \phi_k\rangle = \langle \phi_j, x\phi_k\rangle = 0 \text{ if } k \le j-2$$

since $x\phi_k$ has degree $\le j-1$ if $k \le j-2$. Thus $x\phi_j$ is already orthogonal to the previous basis functions except ϕ_{j-1} and ϕ_j, so the Gram-Schmidt formula only has three terms in it:

$$\begin{aligned}\phi_{j+1} &= x\phi_j - \frac{\langle x\phi_j, \phi_j\rangle}{\langle \phi_j, \phi_j\rangle}\phi_j - \frac{\langle x\phi_j, \phi_{j-1}\rangle}{\langle \phi_{j-1}, \phi_{j-1}\rangle}\phi_{j-1} \\ &= (x-\alpha_j)\phi_j + \beta_j\phi_{j-1}\end{aligned}$$

for values α_j, β_j that can be computed. The formula can be simplified a bit further; see the textbook. The point here is that the 'three-term' formula allows the polynomials to be generated in the same number of calculations per step, so they are reasonable to compute (via a computer algebra package - not so reasonable by hand!).

2 Approximation by orthogonal polynomials

2.1 Legendre polynomials

To start, consider $[-1,1]$ and $w(x) = 1$. We use Gram-Schmidt and the three-term recurrence trick to find the basis, which are the **Legendre polynomials.** The first few calculations are as follows:

$$\begin{aligned}\phi_0(x) &= 1 \\ \phi_1(x) &= x - \frac{\langle x, 1\rangle}{\langle 1, 1\rangle}1 = x \\ \phi_2(x) &= x^2 - \frac{\langle x^2, x\rangle}{\langle x, x\rangle}x - \frac{\langle x^2, 1\rangle}{\langle 1, 1\rangle}1 = x^2 - 0\cdot x - \frac{1}{3} = x^2 - 1/3 \\ \phi_3(x) &= x^3 - \frac{1}{3}x - \frac{\langle x\phi_2, \phi_2\rangle}{\langle \phi_2, \phi_2\rangle}\phi_2 - \frac{\langle x\phi_2, x\rangle}{\langle x, x\rangle}x = x^3 - \frac{3}{5}x\end{aligned}$$

and so on. One can obtain a recurrence for the Legendre polynomials by some further work.

Explicitly, the orthogonality relation is that

$$\int_{-1}^{1} \phi_i(x)\phi_j(x)\,dx = \begin{cases} 0 & i \ne j \\ n_j & i = j\end{cases}$$

and one can compute n_j explicitly with some work (for much more detail and a standard reference, see Abramowitz and Stegun). Any function in $L^2[-1,1]$ may then be expressed as a series in terms of this basis as

$$f = \sum_{j=0}^{\infty} c_j\phi_j, \quad c_j = \frac{\langle f, \phi_j\rangle}{\langle \phi_j, \phi_j\rangle} = \frac{1}{n_j}\int_{-1}^{1} f(x)\phi_j(x)\,dx.$$

(Convention): Traditionally, the Legendre polynomials are normalized so that $\phi_j(1) = 1$. If this is done, then they satisfy

$$(j+1)\phi_{j+1} - (2j+1)x\phi_j + j\phi_{j-1} = 0.$$

By this process, $\phi_2 = \frac{1}{2}(3x^2 - 1)$ and $\phi_3 = \frac{1}{2}(5x^3 - 3x)$ and so on.

Example (linear approx.): As a simple application, suppose we wish to construct the best 'least-squares' line to $f(x) = e^x$ in the interval $[0, 1]$. First, change variables to $s \in [-1, 1]$ using $x = (s+1)/2$:

$$g(s) = f((s+1)/2) = e^{(s+1)/2}.$$

By the theory, g has a representation in terms of the Legendre basis:

$$g = \sum_{j=0}^{\infty} c_j \phi_j(s), \quad c_j = \frac{\langle g, \phi_j \rangle}{\langle \phi_j, \phi_j \rangle}.$$

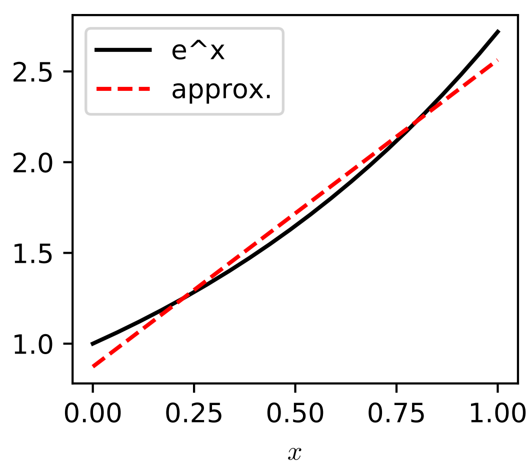
The first two terms are the best approximation in $\mathbb{P}_1$ in the least-squares sense (by the general theory), so we need only calculate c_0 and c_1 (with $\phi_0 = 1$ and $\phi_1 = s$):

$$c_0 = \frac{\int_{-1}^{1} e^{(s+1)/2}\, ds}{\int_{-1}^{1} 1^2\, ds} = e - 1, \quad c_1 = \frac{\int_{-1}^{1} e^{(s+1)/2} s\, ds}{\int_{-1}^{1} s^2\, ds} = 9 - 3e.$$

Converting back to $x \in [0, 1]$ we have the approximation

$$f(x) = g(2x-1) \approx (e-1) + (9-3e)(2x-1).$$

This line minimizes the L^2 error in $[0, 1]$.



2.2 Chebyshev polynomials

For the inner product

$$\langle f, g\rangle = \int_{-1}^{1} \frac{f(x)g(x)}{\sqrt{1-x^2}}\,dx$$

we obtain the **Chebyshev polynomials**. To obtain them, simply transform the integral using $x = \cos\theta$ (so $\theta \in [0, \pi]$); then

$$\langle f, g\rangle = \int_{0}^{\pi} f(\cos\theta)g(\cos(\theta))\,d\theta.$$

We know from Fourier series that the set $\{cos(k\theta)\}$ is orthogonal on $[0, \pi]$ in the L^2 inner product $\int_0^\pi f(\theta)g(\theta)\,d\theta$, from which it follows that the polynomials satisfy

$$T_k(\cos(\theta)) = \cos(k\theta)$$

so they are given by the explicit formula

$$T_j(x) = \cos(j\cos^{-1}(x)). \tag{3}$$

Trigonometric identities guarantee that this formula actually produces polynomials. For example,

$$T_2(\cos(\theta)) = \cos(2\theta) = 2\cos^2(\theta) - 1 \implies T_2 = 2x^2 - 1.$$

The Chebyshev polynomials and the corresponding 'Chebyshev nodes' (the zeros)

$$x_k = \cos((k + 1/2)\pi/j), \quad k = 0, \cdots j - 1. \tag{4}$$

play a key role in numerical analysis due to their close relation to Fourier series, among other nice properties. The three-term recurrence reduces to

$$T_{j+1} = 2xT_j - T_{j-1}, \quad j = 1, 2, \cdots$$

and the first few Chebyshev polynomials are $T_0(x) = 1$ and

$$T_1(x) = x, \quad T_2 = 2x^2 - 1, \quad T_3 = 4x^3 - 3x, \cdots$$

The leading coefficient of $T_j(x)$ is 2^{j-1}, so $2^{1-j}T_j(x)$ is a monic polynomial.

2.2.1 Minimax property

An interesting question is to determine the polynomial that minimizes

$$\max_{x\in[-1,1]} |p(x)| \text{ for monic } p \in \mathbb{P}_j.$$

This is the polynomial of 'least oscillation' - it has the smallest peaks among all monic polynomials of that degree.

Surprisingly, the answer is that the scaled Chebyshev polynomial

$$t_j(x) = 2^{1-j}T_j(x) = x^j + \cdots$$

is this minimizer.

Theorem The monic Chebyshev polynomials t_j have the **minimax property** that

$$t_j(x) \text{ minimizes } \max_{x\in[-1,1]} |p(x)| \text{ for monic } p \in \mathbb{P}_j.$$

Since $|T_j(x)| \le 1$ and can equal 1, the minimum is 2^{1-n}, i.e.

$$\min_{\text{monic } p\in\mathbb{P}_j} \left(\max_{x\in[-1,1]} |p(x)| \right) = 2^{1-n}.$$

Equivalently, the **Chebyshev nodes** (4) (the zeros of $T_j(x)$) minimize

$$\max_{x\in[-1,1]} \left| \prod_{k=0}^{j-1} (x - x_k) \right|$$

among all sets of nodes $x_0, \cdots, x_{j-1}$.

This suggests that Chebyshev polynomials can be used to minimize (or at least come close to doing so) the maximum error.

Interpolation: Incidentally, the minimax property in the second form shows that for interpolation, the Chebyshev nodes do a good job of keeping the Lagrange error under control:

$$\left| \frac{f^{(n+1)}(\eta_x)}{(n+1)!} \prod_{k=0}^{n} (x - x_k) \right| \le \frac{M_n}{(n+1)!} 2^{-n},$$

which explains why they are such a good choice for interpolation.

3 Gaussian quadrature

The structure here provides an elegant way to construct a formula

$$I = \int_a^b f(x)w(x)\,dx \approx \sum_{k=0}^{n} c_k f(x_k) \tag{5}$$

with the highest possible degree of accuracy. Here both the coefficients **and the nodes** are to be chosen. There are $2n+2$ unknowns, suggesting a degree of accuracy of $2n+1$.

- Let $\langle f, g\rangle_w = \int_a^b f(x)g(x)w(x)\,dx$ (the weighted inner product). By the Gram-Schmidt process, there is a sequence $\{\phi_j\}$ of orthogonal polynomials where ϕ_j has degree j.
- ϕ_{n+1} has $n+1$ distinct real zeros $x_0, \cdots, x_n$ in $[a, b]$
- Let $\ell_k(x)$ be the k-th Lagrange basis polynomial for these zeros and let

$$c_k = \int_a^b \ell_k(x)\,dx.$$

The claim is that with this set of x_k's and c_k's, the formula (5) has degree $2n+1$.

Proof. Suppose $f \in \mathbb{P}_{2n+1}$. Since p_{n+1} has degree $n+1$, polynomial division gives

$$f = q(x)p_{n+1}(x) + r(x), \qquad q, r \in \mathbb{P}_n.$$

Plugging this expression into the integral,

$$\begin{aligned} I &= \int_a^b (q(x)p_{n+1}(x) + r(x))w(x)\,dx \\ &= \langle q, p_{n+1}\rangle_w + \int_a^b r(x)w(x)\,dx \\ &= \int_a^b r(x)w(x)\,dx \end{aligned}$$

because p_{n+1} is orthogonal to all polynomials of degree $\leq n$, which includes q.
Now plug the expression into the formula:

$$\begin{aligned} \text{formula} &= \sum_{k=0}^{n} c_k f(x_k) \\ &= \sum_{k=0}^{n} c_k q(x_k) p_{n+1}(x_k) + \sum_{k=0}^{n} c_k r(x_k) \\ &= \sum_{k=0}^{n} c_k r(x_k). \end{aligned}$$

Last, we need to establish that I and the formula are equal. Because $r(x)$ has degree $\leq n$, it is equal to its Lagrange interpolant through the nodes $x_0, \cdots, x_n$, so

$$r(x) = \sum_{k=0}^{n} r(x_k)\ell_k(x).$$

Thus, working from the formula for I,

$$I = \int_a^b r(x)w(x)\,dx = \sum_{k=0}^{n} r(x_k) \int_a^b \ell_k(x)w(x)\,dx = \sum_{k=0}^{n} c_k r(x_k)$$

which establishes equality. To see that the degree of accuracy is exactly $2n+1$, consider

$$f(x) = \prod_{j=0}^{n} (x - x_j)^2.$$

□

Summary (Gaussian quadrature) Let $\{\phi_j\}$ be an orthogonal basis of polynomials in the inner product $\langle f, g\rangle_w \int_a^b f(x)g(x)w(x)\,dx$ and let $x_0, \cdots, x_n$ be the zeros of the polynomial ϕ_{n+1} with Lagrange basis $\{\ell_k(x)\}$. Then

$$I = \int_a^b f(x)w(x)\,dx \approx \sum_{k=0}^{n} c_k f(x_k), \qquad c_k = \int_a^b \ell_k(x)w(x)\,dx, \tag{6}$$

called the **Gaussian quadrature formula** for $w(x)$, has degree of accuracy $2n+1$.

Note that the nodes x_k depend on the degree, so really they should be written $x_{n,k}$ (for $k = 0, \cdots, n$) for ϕ_{n+1}. One can show that, unlike with equally spaced interpolation,

$$\lim_{n\to\infty} |I - \sum_{k=0}^{n} c_k f(x_{n,k})| = 0$$

under reasonable assumptions on f (see textbook for details), and the rate Thus, Gaussian quadrature does well when adding more points to reduce error when function values of f at any point are available.

Example: For example, when $n = 1$ and $w(x) = 1$we have

$$\int_{-1}^{1} f(x)\,dx \approx c_0 f(x_0) + c_1 f(x_1).$$

The nodes are the zeros of $p_2(x) = x^2 - 1/3$, so

$$x_0 = -\frac{1}{\sqrt{3}}, \quad x_1 = \frac{1}{\sqrt{3}}.$$

It is not hard to then compute $c_0 = c_1 = 1$, yielding

$$\int_{-1}^{1} f(x)\,dx = f(-\frac{1}{\sqrt{3}}) + f(\frac{1}{\sqrt{3}})$$

which has degree of accuracy 3.

The value here is that the accuracy is quite high - however, one has to have control over the choice of nodes for the formula to be available.

When $n = 2$, $p_2(x) = x^3 - 3x/5$ so the zeros are at $\pm\sqrt{3/5}$ and 0:

$$\int_{-1}^{1} f(x)\,dx \approx c_0 f(-\sqrt{3/5}) + c_1 f(0) + c_2 f(\sqrt{3/5})$$

and this formula has degree of accuracy 5. The coefficients c_k are computed from the the formula (6); while the algebra is messy, they can be computed in advance.

3.1 Gauss-Lobatto quadrature

In a slight variation, we instead include the endpoints a and b in the integration.[2] The claim is that the formula (supposing $w(x) = 1$ for simplicity)

$$I = \int_a^b f(x)\,dx \approx \sum_{k=0}^{n} c_k f(y_k)$$

where

$$y_1, y_2, \cdots y_{n-1} = \text{zeros of } p_n' \text{ in } (a,b), \quad x_0' = a, \;\; y_n = b,$$

$$c_i = \int_a^b \ell_i(x)\,dx, \quad \ell_i(x) = \text{Lagrange basis poly. for } y_0, y_1, \cdots, y_n$$

is exact for all $f \in \mathbb{P}_{2n-1}$. That is, it has a degree of accuracy two less than Gaussian quadrature ($2n-1$ vs. $2n+1$). The Lobatto version is used when the endpoints are needed in the approximation (and as a building block for other methods that need the endpoints).

The existence of the zeros y_k is clear (it can be shown that the polynomial p_n has n distinct real zeros, the nodes we used for Gaussian quadrature). The coefficient formula c_j has the same derivation as before.

To show the degree of accuracy,suppose $f \in \mathbb{P}_{2n-1}$ and use polynomial division to write (note that p_n' has degree $n-1$)

$$f(x) = q(x)p_n'(x) + r(x), \quad q, r \in \mathbb{P}_n.$$

Then after an integration by parts,

$$\int_a^b f(x)\,dx = q(x)p_n(x)\Big|_a^b - \int_a^b q'(x)p_n(x)\,dx + \int_a^b r(x)\,dx$$
$$= q(b)p_n(b) - q(a)p_n(a) + \sum_{k=0}^{n} c_k r(x_k)$$

since q' has degree $n-1$ so it is orthogonal to p_n. Now note that

$$r(x_k) = f(x_k) \text{ if } 1 \le k \le n-1$$

but is not equal when $k = 0$ or $k = n$, so

$$\int_a^b f(x)\,dx = q(b)p_n(b) - q(a)p_n(a) + \sum_{k=0}^{n} c_k f(x_k) - q(a)p_n'(a)c_0 - q(b)p_n'(b)c_n$$

$$\int_a^b f(x)\,dx = \sum_{k=0}^{n} c_k f(x_k) + q(b)(p_n(b) - p_n'(b)c_n) - q(a)(p_n(a) + p_n'(a)c_0)$$

The boundary terms can be shown to vanish using the identities

$$0 = \int_a^b p_n(x)\,dx, \quad p_n(b) - p_n(a) = \int_a^b p_n'(x)\,dx$$

noting that p_n is orthogonal to $p_0 = 1$ (left as an exercise).

[2] Adapted from Trangenstein, *Scientific Computing* lecture notes, 2011.

3.2 Example: Laplace transform (Laguerre)

The Laplace transform is defined as

$$F(s) = \mathcal{L}[f(t)] = \int_0^\infty f(t)e^{-st}\,dt.$$

Often, we need to evaluate this transform at many different points s given a function $f(t)$. This integral is over an **infinite** interval, which can be handled using Gaussian quadrature. First, scale out the s with $x = st$ to get

$$F(s) = \frac{1}{s}\int_0^\infty f(x/s)e^{-x}\,dx.$$

Now to be efficient, consider Gaussian quadrature in $[0, \infty)$ with weight $w(x) = e^{-x}$; the inner product on $L^2_w([0, \infty))$ is

$$\langle f, g\rangle = \int_0^\infty f(x)g(x)e^{-x}\,dx.$$

Start with $p_0(x) = 1$ and then (using $1, x, x^2$ for simplicity here)

$$p_1(x) = x - \frac{\langle x, 1\rangle}{\langle 1, 1\rangle}1 = x - \int_0^\infty xe^{-x}\,dx = x - 1,$$

$$p_2(x) = x^2 - \frac{\langle x^2, x-1\rangle}{\langle x-1, x-1\rangle}(x-1) - \frac{\langle x^2, 1\rangle}{\langle 1, 1\rangle}1 = x^2 - 4x + 2$$

and so on. With two points, the nodes are

$$x_0 = 2 - \sqrt{2}, \quad x_1 = 2 + \sqrt{2}$$

and the coefficients are

$$c_0 = \int_0^\infty \frac{x - x_1}{x_0 - x_1}e^{-x}\,dx = -\frac{1}{2\sqrt{2}}\int_0^\infty (x - x_1)e^{-x}\,dx = \frac{1}{4}(2 + \sqrt{2}) \approx 0.85355$$

$$c_1 = \int_0^\infty \frac{x - x_0}{x_1 - x_0}e^{-x}\,dx \approx 0.146447$$

so a quick to compute approximation with degree of accuracy 3 is

$$\int_0^\infty g(x)e^{-x}\,dx \approx c_0 g(x_0) + c_1 g(x_1).$$

One can, of course, go to a much higher degree if more accuracy is needed; the weights c_i and nodes x_i are not pleasant to compute, but this can be done in advance to high accuracy and then stored as hard-coded values in the algorithm.

3.3 Singular integrals, briefly

There are many strategies for computing singular integrals. A few starting ideas are presented here. Take, for example, the integral

$$I = \int_0^1 \frac{\sin x}{x^{3/2}}\,dx.$$

Option 1 (brute force): We can use an open Newton-Cotes formula, which avoids the singularity at the endpoint $x = 0$. However, the singularity means convergence results may not apply.

Option 2 (local approximation): The trick here is to use an asymptotic approximation (from theory) near the singularity. In this case, we can just use a Taylor series. Let

$$f(x) = \frac{\sin x}{x^{3/2}}.$$

Then expand the series for $\sin x$ to get

$$f(x) = \sum_{n=0}^{\infty} \frac{(-1)^{n+1}}{(2n+1)!} x^{2n-1/2}$$

Now split the integral into a 'small' singular region and a 'large' good region:

$$I = \int_0^{\epsilon} f(x)\,dx + \int_{\epsilon}^1 f(x)\,dx = I_{1,\epsilon} + I_{2,\epsilon}.$$

For the good region, just use any normal method; the integrand is not singular. (Note that an adaptive method is suggested, since one probably needs higher accuracy near $x = \epsilon$.

For the bad region, integrate the power series term by term analytically:

$$\int_0^{\epsilon} f(x)\,dx = \sum_{n=0}^{\infty} \frac{(-1)^{n+1}}{(2n+1)!(2n+1/2)} \epsilon^{2n+1/2}$$

We now choose ϵ small and enough terms of the sum to get the desired accuracy.

Option 3: Gaussian quadrature. In some cases, one can use Gaussian quadrature, putting the singularity into the weight function. Here we write

$$I = \int_0^1 \frac{\sin x/x}{x^{1/2}}\,dx, \quad \langle f, g\rangle = \int_0^1 \frac{f(x)g(x)}{x^{1/2}}\,dx.$$

The proceed by obtaining the orthogonal polynomials and their zeros. This approach can be useful if we can do the calculation of the weights/nodes in advance.

Option 4: Transform! There are a number of tricks to transform a singular integral into a non-singular one. These methods are of varying complexity and tend to be problem dependent (exception: the rather general double exponential rule). The details will not be pursued here.