1 Orthogonal polynomial methods for non-periodic functions

We learned that smooth periodic functions and their derivatives can be well approximated by trigonometric interpolants:

$$f(x) \approx p(x) = \sum_{k=-(n-1)/2}^{(n-1)/2} \tilde{c}_k \mathrm{e}^{\mathrm{i}kx}$$

and there are fast algorithms for mapping function values to coeffients and vice versa and for differentiation. We saw how Fourier approximation methods could be used to compute an approximate solution to a PDE.

Here we'll learn that smooth non-periodic functions and their derivatives can be accurately and stably approximated by expansions in orthogonal polynomials (OPs),

$$f(x) \approx \sum_{k=0}^{n} c_k p_k(x).$$

and there are fast algorithms for computing with OPs. We'll also learn how OPs can be used to approximate solutions to PDEs.

1.1 The Runge phenomenon and its resolution via OPs

Suppose we want to approximate the function $f(x) = \frac{1}{1+25x^2}$ on the interval $x \in [-1, 1]$. We know that a trigonometric interpolant will not converge very fast to f. Instead we try to approximate f with a Lagrange interpolating polynomial. Recall from Lecture 1 that one representation of the unique polynomial $p(x)$ of degree $\leq n$ that interpolates f at $x_0, \ldots, x_{n+1}$ ($p(x_j) = f(x_j)$, $j = 0, \ldots, n$) is as follows

$$p(x) = \sum_{j=0}^{n} \ell_j(x) f(x_j),$$

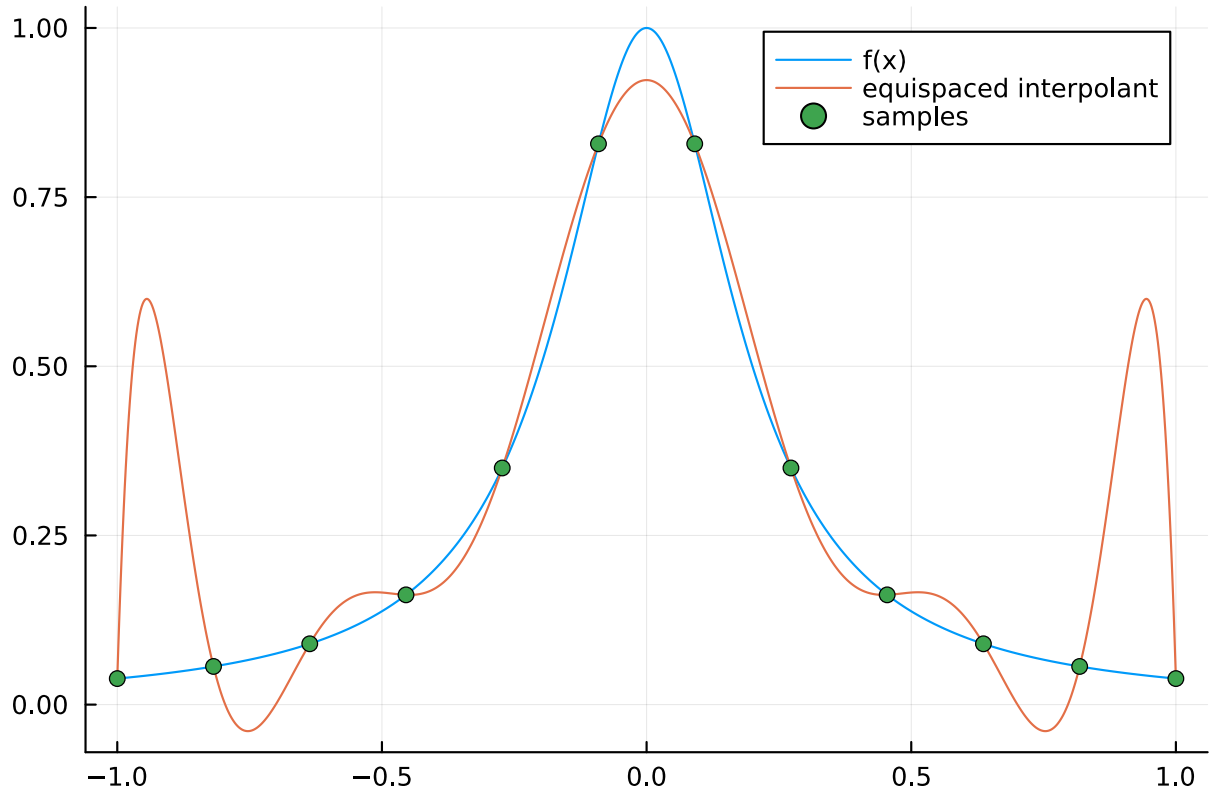
where

$$\ell_j(x) = \prod_{\substack{i=0 \\ i \neq j}}^{n} \frac{x - x_i}{x_j - x_i}.$$

First, we let the interpolation nodes x_j, $j = 0, \ldots, n$ be equally spaced on the interval $[-1, 1]$. Instead of using the formula for the Lagrange interpolating polynomial, we'll use ApproxFun.jl to construct the polynomial interpolant:

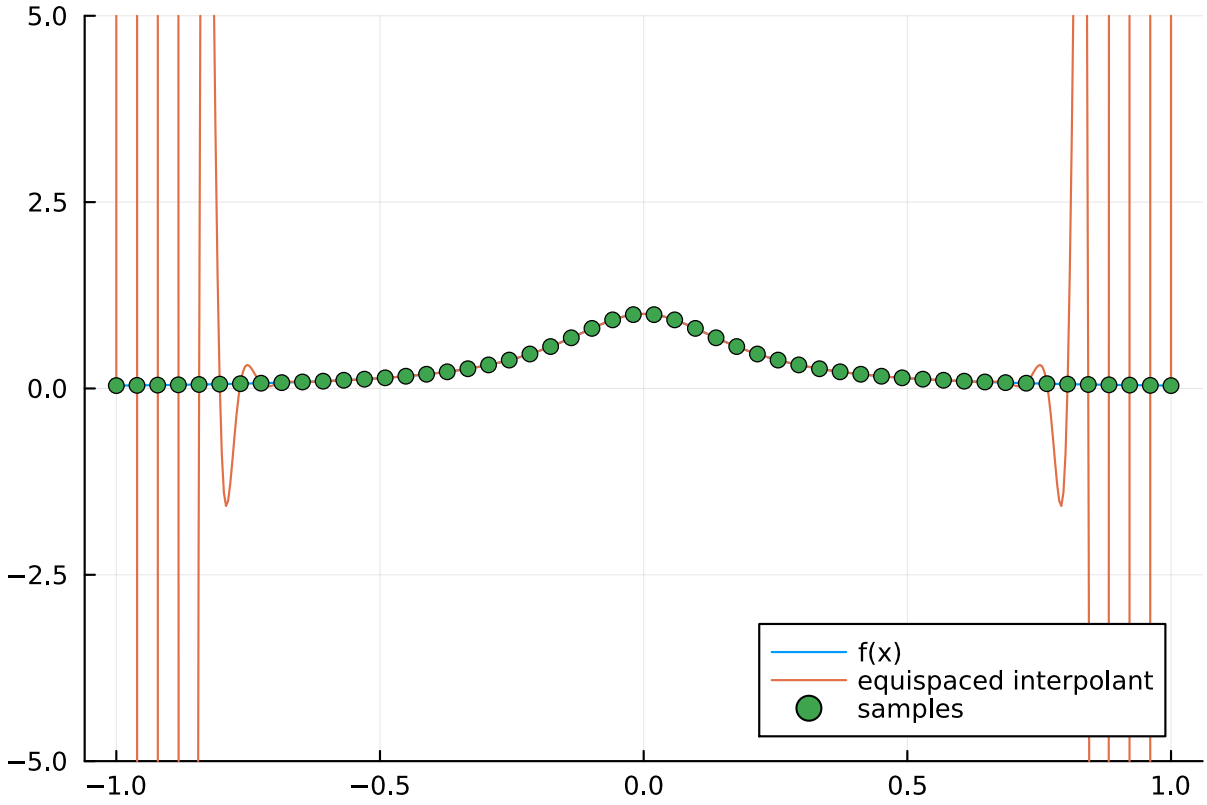
```
using ApproxFun, Plots

function equi_interp(f,n)
# Construct a polynomial interpolant of f at n+1 equispaced points on [-1, 1]
S = Chebyshev()
xk = range(-1,stop=1,length=n+1); # n equally spaced points
v = f.(xk);
V = Array{Float64}(undef,n+1,n+1); # Create a Vandermonde matrix by evaluating the basis
at the grid
```

```
for k = 1:n+1
    V[:,k] = Fun(S,[zeros(k-1);1]).(xk)
end
p_n = Fun(S,V\v)
p_n
end;

f = x -> 1/(25x^2 + 1)
n = 11
nodes = range(-1,1;length=n+1)
xx = range(-1,1;length=1001)  # plotting grid
p = equi_interp(f,n)
plot(xx,f.(xx);label="f(x)")
plot!(xx,p.(xx);label="equispaced interpolant")
scatter!(nodes,p.(nodes);label="samples")
```



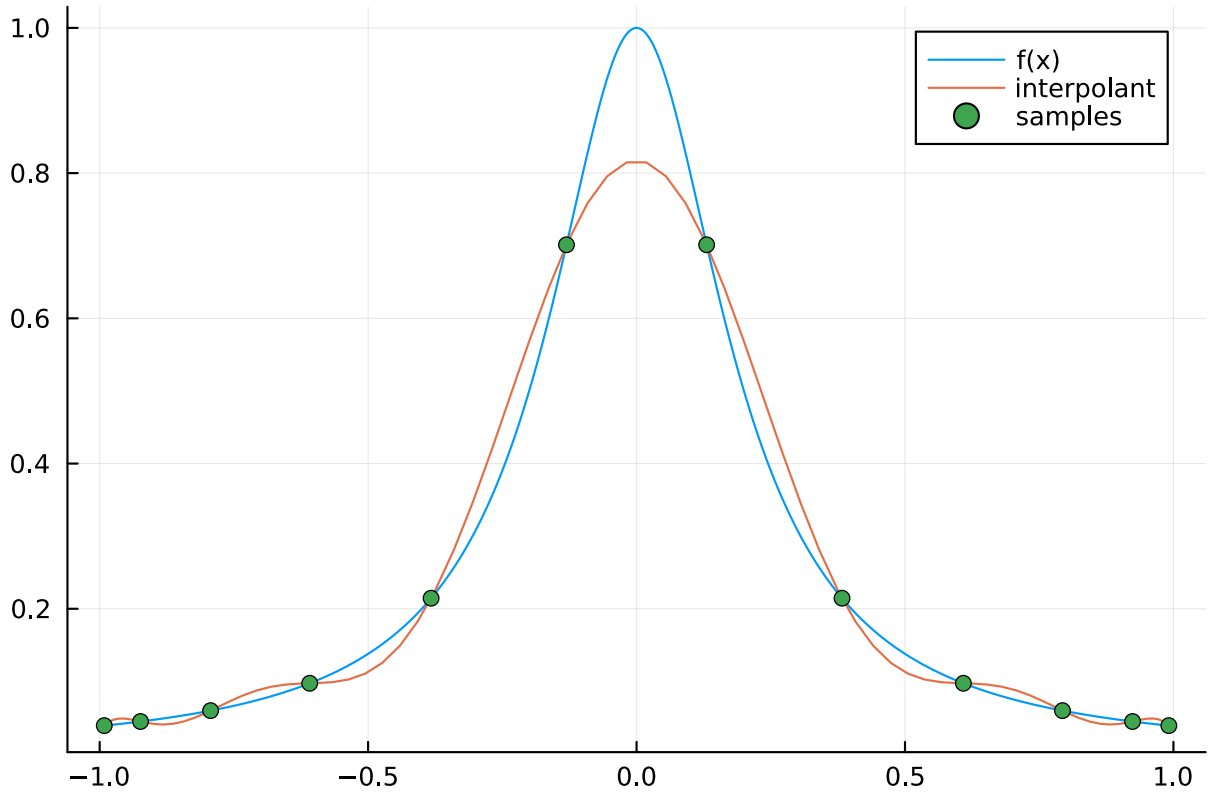
The oscillations at the ends of the interval become larger as we increase n:

```
n = 51
nodes = range(-1,stop=1,length=n+1)
xx = range(-1,1;length=501)  # plotting grid
p = equi_interp(f,n)
plot(xx,f.(xx);label="f(x)",ylims=(-5,5))
plot!(xx,p.(xx);label="equispaced interpolant")
scatter!(nodes,p.(nodes);label="samples")
```



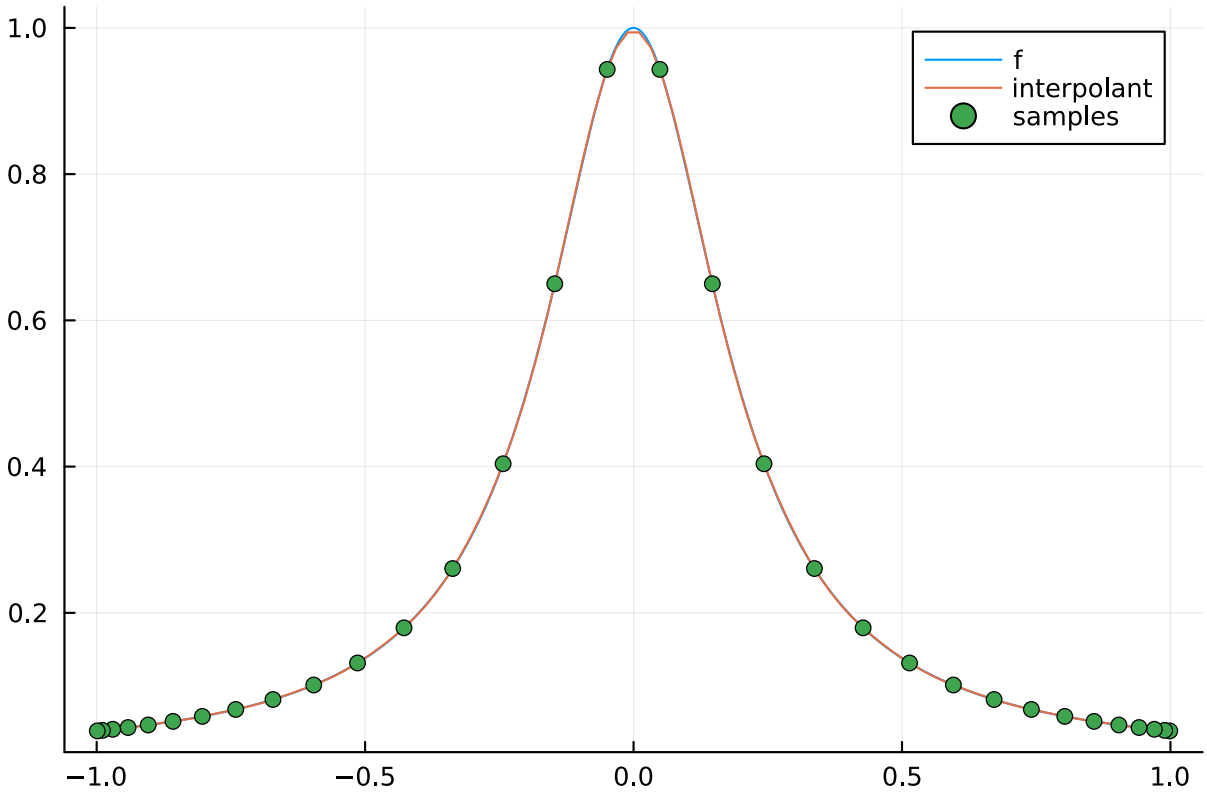
These oscillations of the interpolant through equally spaced nodes is known as the *Runge phenomenon*. Equally spaced interpolation nodes is clearly not a good choice for polynomial interpolation. Maybe the oscillations at the endpoints can be suppressed if we choose nodes that are clustered at the ends of the interval?

```
n = 11
S = Chebyshev()
p_n = Fun(f,S,n+1)
xn = points(S,n+1)
plot(xx,f.(xx);label="f(x)")
plot!(p_n;label="interpolant")
scatter!(xn,p_n.(xn);label="samples")
```



In the above figure, we interpolated $f(x)$ at the roots of an orthogonal polynomial (OP), which we'll learn about later. The interpolants at the roots of the OP converge exponentially fast to f on $[-1, 1]$ as n increases.

```
n = 31
S = Chebyshev()
p_n = Fun(f,S,n+1)
xn = points(S,n+1)
plot(xx,f.(xx);label="f")
plot!(p_n;label="interpolant")
scatter!(xn,p_n.(xn);label="samples")
```



A deep and quantitative understanding of the reasons why equispaced interpolation failed for the above function and why interpolants through clustered points converged fast is a beautiful and advanced topic that requires tools from complex analysis and potential theory.

We now introduce orthogonal polynomials (OPs). These are **fundamental** for computational mathematics, with applications in

1. Function approximation
2. Quadrature (calculating integrals)
3. Solving differential equations
4. Spectral analysis of Schrödinger operators

We will investigate the properties of *general* OPs, in this lecture:

1. Definition of orthogonal polynomials
2. Three-term recurrence relationships
3. Function approximation with orthogonal polynomials
4. Construction of orthogonal polynomials via GramSchmidt process

In addition to numerics, OPs play a very important role in many mathematical areas including functional analysis, integrable systems, singular integral equations, complex analysis, and random matrix theory.

1. General properties of OPs: we define orthogonal polynomials, three-term recurrences and Jacobi operators

2. Classical OPs: we define Chebyshev, Legendre, Jacobi, Laguerre, and Hermite.

Remark (advanced) Going beyond what we discuss here are many other beautiful properties of orthogonal polynomials that are useful in computation, analysis, representation theory, quantum mechanics, etc. These include sparse recurrence relationships for derivatives, that they are eigenfunctions of differential equations, their asymptotics, generating functions, etc.

1.2 Definition of orthogonal polynomials

Let $p_0(x), p_1(x), p_2(x), \ldots$ be a sequence of polynomials such that $p_n(x)$ is exactly of degree n, that is,

$$p_n(x) = k_n x^n + O(x^{n-1})$$

where $k_n \neq 0$.

Let $w(x)$ be a continuous weight function on a (possibly infinite) interval (a, b): that is $w(x) \geq 0$ for all $a < x < b$. This induces an inner product

$$\langle f, g \rangle := \int_a^b f(x)g(x)w(x)\mathrm{d}x$$

We say that $\{p_0, p_1, \ldots\}$ are *orthogonal with respect to the weight* w if

$$\langle p_n, p_m \rangle = 0 \qquad \text{for } n \neq m.$$

Because w is continuous, we have

$$\|p_n\|^2 = \langle p_n, p_n \rangle > 0.$$

Orthogonal polymomials are not unique: we can multiply each p_n by a different nonzero constant $\tilde{p}_n(x) = c_n p_n(x)$, and $\tilde{p}_n$ will be orthogonal w.r.t. w. However, if we specify k_n, this is sufficient to uniquely define them:

Proposition (Uniqueness of OPs I) Given a non-zero k_n, there is a unique polynomial p_n orthogonal w.r.t. w to all lower degree polynomials.

Proof Suppose $r_n(x) = k_n x^n + O(x^{n-1})$ is another OP w.r.t. w. We want to show $p_n - r_n$ is zero. But this is a polynomial of degree $< n$, hence

$$p_n(x) - r_n(x) = \sum_{k=0}^{n-1} c_k p_k(x)$$

But we have for $k \leq n - 1$

$$\langle p_k, p_k \rangle c_k = \langle p_n - r_n, p_k \rangle = \langle p_n, p_k \rangle - \langle r_n, p_k \rangle = 0 - 0 = 0$$

which shows all c_k are zero.

■

Corollary (Uniqueness of OPs I) If q_n and p_n are orthogonal w.r.t. w to all lower degree polynomials, then $q_n(x) = Cp_n(x)$ for some constant C.

1.2.1 Monic orthogonal polynomials

If $k_n = 1$, that is,

$$p_n(x) = x^n + O(x^{n-1})$$

then we refer to the orthogonal polynomials as monic.

Monic OPs are unique as we have specified k_n.

1.2.2 Orthonormal polynomials

If $\|p_n\| = 1$, then we refer to the orthogonal polynomials as orthonormal w.r.t. w. We will usually use q_n when they are orthonormal. Note it's not unique: we can multiply by ± 1 without changing the norm.

Remark The classical OPs are neither monic nor orthonormal (apart from one case). Many people make the mistake of using orthonormal polynomials for computations. But there is a good reason to use classical OPs: their properties result in rational formulae, whereas orthonormal polynomials require square roots. This makes a performance difference.

1.3 Orthogonal polynomials: definitions and properties

The set of polynomials of degree $\leq n$ with real coefficients, $\mathbb{P}_n$, is a linear space (or vector space) of dimension $n+1$. The set of monomials of degree $\leq n$, $\{1, x, x^2, \ldots, x^n\}$, is a basis for the space $\mathbb{P}_n$, meaning that all polynomials of degree $\leq n$ can be expressed as linear combinations of monomials. For example, the Lagrange interpolating polynomial of degree $\leq n$ can be expressed in the monomial basis. However, it is more efficient and stable to perform computations in OP bases as oposed to the monomial basis.

Definition (graded polynomial basis) A set of polynomials $\{p_0(x), p_1(x), \ldots\}$ is *graded* if p_n is precisely degree n: i.e.,

$$p_n(x) = k_n x^n + k_n^{(n-1)} x^{n-1} + \cdots + k_n^{(1)} x + k_n^{(0)}$$

with $k_n \neq 0$.

Note that if the p_n are graded then $\{p_0(x), \ldots, p_n(x)\}$ form a basis for all polynomials of degree n. For example, the Lagrange interpolating polynomial can be expressed in any graded polynomial basis.

Definition (orthogonal polynomials) Given an (integrable) *weight* $w(x) > 0$ for $x \in (a, b)$, associated with which is the inner product

$$\langle f, g \rangle = \int_a^b f(x) g(x) w(x) \mathrm{d}x$$

a graded polynomial basis $\{p_0(x), p_1(x), \ldots\}$ is a set of *orthogonal polynomials (OPs)* if

$$\langle p_n, p_m \rangle = 0$$

whenever $n \neq m$.

Note in the above

$$h_n := \langle p_n, p_n \rangle = \|p_n\|^2 = \int_a^b p_n(x)^2 w(x) \mathrm{d}x > 0.$$

Multiplying any orthogonal polynomial by a constant necessarily is also an orthogonal polynomial. We have two standard normalisations:

Definition (orthonormal polynomials) A set of orthogonal polynomials $\{q_0(x), q_1(x), \ldots\}$ are *orthonormal* if $\|q_n\| = 1$.

Definition (monic orthogonal polynomials) A set of orthogonal polynomials $\{p_0(x), p_1(x), \ldots\}$ are *monic* if $k_n = 1$.

Proposition (existence) Given a weight $w(x)$, monic orthogonal polynomials exist.

Proof

Existence follows immediately from the GramSchmidt procedure. That is, define $p_0(x) := 1$ and

$$p_n(x) := x^n - \sum_{k=0}^{n-1} \frac{\langle x^n, p_k \rangle}{\|p_k\|^2} p_k(x)$$

■

1.4 Function approximation with orthogonal polynomials

A basic usage of orthogonal polynomials is for polynomial approximation. Suppose $f(x)$ is a degree $n-1$ polynomial. Since $\{p_0(x), \ldots, p_{n-1}(x)\}$ span all degree $n-1$ polynomials, we know that

$$f(x) = \sum_{k=0}^{n-1} f_k p_k(x)$$

where

$$f_k = \frac{\langle f, p_k \rangle}{\langle p_k, p_k \rangle}$$

Sometimes, we want to incorporate the weight into the approximation. This is typically one of two forms, depending on the application:

$$f(x) = w(x) \sum_{k=0}^{\infty} f_k p_k(x)$$

or

$$f(x) = \sqrt{w(x)} \sum_{k=0}^{\infty} f_k p_k(x)$$

The $w(x)p_k(x)$ or $\sqrt{w(x)}p_k(x)$ are called weighted polynomials.

We are primarly concerned with the usage of orthogonal polynomials in approximating functions. First we observe the following:

Proposition (expansion) If $r(x)$ is a degree n polynomial, $\{p_n\}$ are orthogonal and $\{q_n\}$ are orthonormal then

$$\begin{aligned} r(x) &= \sum_{k=0}^{n} \frac{\langle p_k, r\rangle}{\|p_k\|^2} p_k(x) \\ &= \sum_{k=0}^{n} \langle q_k, r\rangle q_k(x) \end{aligned}$$

Proof Because $\{p_0, \ldots, p_n\}$ are a basis of polynomials we can write

$$r(x) = \sum_{k=0}^{n} r_k p_k(x)$$

for constants $r_k \in \mathbb{R}$. By linearity we have

$$\langle p_m, r\rangle = \sum_{k=0}^{n} r_k \langle p_m, p_k\rangle = r_m \langle p_m, p_m\rangle$$

for $m = 0, \ldots, n$. ■

Corollary (zero inner product) If a degree n polynomial r satisfies

$$0 = \langle p_0, r\rangle = \ldots = \langle p_n, r\rangle$$

then $r = 0$.

Corollary (uniqueness) Monic orthogonal polynomials are unique.

Proof If $\widetilde{p}_n(x)$ and $p_n(x)$ are both monic orthogonal polynomials then $r(x) = \widetilde{p}_n(x) - p_n(x)$ is degree $n-1$ but satisfies

$$\langle r, p_k\rangle = \langle \widetilde{p}_n, p_k\rangle - \langle p_n, p_k\rangle = 0$$

for $k = 0, \ldots, n-1$. Note $\langle \widetilde{p}_n, p_k\rangle = 0$ can be seen by expanding

$$p_k(x) = \sum_{j=0}^{k} c_j \widetilde{p}_j(x).$$

■

OPs are uniquely defined (up to a constant) by the property that they are orthogonal to all lower degree polynomials.

Proposition (orthogonal to lower degree) Given a weight $w(x)$, a polynomial p of precisely degree n satisfies

$$\langle p, r \rangle = 0$$

for all degree $m < n$ polynomials r if and only if $p(x) = cp_n(x)$ where $p_n(x)$ are the monic orthogonal polynomials. Therefore an orthogonal polynomial is uniquely defined by k_n.

Proof ($\Leftarrow$) As $\{p_0, \ldots, p_n\}$ are a basis of all polynomials of degree n, we can write

$$r(x) = \sum_{k=0}^{m} a_k p_k(x)$$

Thus by linearity of inner products we have

$$\langle cp_n, \sum_{k=0}^{m} a_k p_k \rangle = \sum_{k=0}^{m} c a_k \langle p_n, p_k \rangle = 0.$$

($\Rightarrow$) Now for

$$p(x) = cx^n + O(x^{n-1})$$

consider $p(x) - cp_n(x)$ which is of degree $n-1$. It satisfies for $k \leq n-1$

$$\langle p_k, p - cp_n \rangle = \langle p_k, p \rangle - c\langle p_k, p_n \rangle = 0.$$

Thus it is zero, i.e., $p(x) = cp_n(x)$.

■

A consequence of this is that orthonormal polynomials are always a constant multiple of orthogonal polynomials.

1.5 Jacobi matrices and three-term recurrences for general orthogonal polynomials

1.5.1 Three-term recurrence relationships

A central theme: if you know the Jacobi matrix / three-term recurrence, you know the polynomials. This is the **best** way to evaluate expansions in orthogonal polynomials: even for cases where we have explicit formulae (e.g. Chebyshev polynomials $T_n(x) = \cos n \arccos x$), using the recurrence avoids evaluating trigonometric functions.

Every family of orthogonal polynomials has a three-term recurrence relationship:

Theorem (three-term recurrence) Suppose $\{p_n(x)\}$ are a family of orthogonal polynomials w.r.t. a weight $w(x)$. Then there exists constants a_n, $b_n \neq 0$ and c_n such that

$$\begin{aligned} xp_0(x) &= a_0 p_0(x) + b_0 p_1(x) \\ xp_n(x) &= c_n p_{n-1}(x) + a_n p_n(x) + b_n p_{n+1}(x) \end{aligned}$$

Proof The first part follows since $p_0(x)$ and $p_1(x)$ span all degree 1 polynomials.

The second part follows essentially because multiplication by x is "self-adjoint", that is,

$$\langle xf, g\rangle = \int_a^b xf(x)g(x)w(x)\mathrm{d}x = \langle f, xg\rangle$$

Therefore, if f_m is a degree $m < n-1$ polynomial, we have

$$\langle xp_n, f_m\rangle = \langle p_n, xf_m\rangle = 0.$$

In particular, if we write

$$xp_n(x) = \sum_{k=0}^{n+1} C_k p_k(x)$$

then

$$C_k = \frac{\langle xp_n, p_k\rangle}{\|p_k\|^2} = 0$$

if $k < n-1$. ■

Monic polynomials clearly have $b_n = 1$. Orthonormal polynomials have an even simpler form:

Theorem (orthonormal three-term recurrence) Suppose $\{q_n(x)\}$ are a family of orthonormal polynomials w.r.t. a weight $w(x)$. Then there exists constants a_n and b_n such that

$$\begin{aligned} xq_0(x) &= a_0 q_0(x) + b_0 q_1(x) \\ xq_n(x) &= b_{n-1}q_{n-1}(x) + a_n q_n(x) + b_n q_{n+1}(x) \end{aligned}$$

Proof Follows again by self-adjointness of multiplication by x:

$$c_n = \langle xq_n, q_{n-1}\rangle = \langle q_n, xq_{n-1}\rangle = \langle xq_{n-1}, q_n\rangle = b_{n-1}$$

■

Corollary (symmetric three-term recurrence implies orthonormality) Suppose $\{p_n(x)\}$ are a family of orthogonal polynomials w.r.t. a weight $w(x)$ such that

$$\begin{aligned} xp_0(x) &= a_0 p_0(x) + b_0 p_1(x) \\ xp_n(x) &= b_{n-1}p_{n-1}(x) + a_n p_n(x) + b_n p_{n+1}(x) \end{aligned}$$

with $b_n \neq 0$. Suppose further that $\|p_0\| = 1$. Then p_n must be orthonormal.

Proof This follows from

$$b_n = \frac{\langle xp_n, p_{n+1}\rangle}{\|p_{n+1}\|^2} = \frac{\langle xp_{n+1}, p_n\rangle}{\|p_{n+1}\|^2} = b_n \frac{\|p_n\|^2}{\|p_{n+1}\|^2}$$

which implies

$$\|p_{n+1}\|^2 = \|p_n\|^2 = \cdots = \|p_0\|^2 = 1$$

■

Remark We can scale $w(x)$ by a constant without changing the orthogonality properties, hence we can make $\|p_0\| = 1$ by changing the weight.

Remark This is beyond the scope of this course, but satisfying a three-term recurrence like this such that coefficients are bounded with $p_0(x) = 1$ is sufficient to show that there exists a $w(x)$ (or more accurately, a Borel measure) such that $p_n(x)$ are orthogonal w.r.t. w. The relationship between the coefficients a_n, b_n and the $w(x)$ is an object of study in spectral theory, particularly when the coefficients are periodic, quasi-periodic or random.

1.5.2 3-term recurrence

The most *fundamental* property of orthogonal polynomials is their three-term recurrence.

Theorem (3-term recurrence, 2nd form) If $\{p_n\}$ are OPs then there exist real constants $a_n, b_n \neq 0, c_{n-1} \neq 0$ such that

$$\begin{aligned} xp_0(x) &= a_0 p_0(x) + b_0 p_1(x) \\ xp_n(x) &= c_{n-1} p_{n-1}(x) + a_n p_n(x) + b_n p_{n+1}(x) \end{aligned}$$

Proof The $n = 0$ case is immediate since $\{p_0, p_1\}$ are a basis of degree 1 polynomials. The $n > 0$ case follows from

$$\langle xp_n, p_k \rangle = \langle p_n, xp_k \rangle = 0$$

for $k < n - 1$ as xp_k is of degree $k + 1 < n$.

Note that

$$b_n = \frac{\langle p_{n+1}, xp_n \rangle}{\|p_{n+1}\|^2} \neq 0$$

since $xp_n = k_n x^{n+1} + O(x^n)$ is precisely degree n. Further,

$$c_{n-1} = \frac{\langle p_{n-1}, xp_n \rangle}{\|p_{n-1}\|^2} = \frac{\langle p_n, xp_{n-1} \rangle}{\|p_{n-1}\|^2} = b_{n-1} \frac{\|p_n\|^2}{\|p_{n-1}\|^2} \neq 0.$$

■

Clearly if p_n is monic then so is xp_n which leads to the following:

Corollary (monic 3-term recurrence) If $\{p_n\}$ are monic then $b_n = 1$.

Example What are the monic OPs $p_0(x), \ldots, p_3(x)$ with respect to $w(x) = 1$ on $[0, 1]$? We can construct these using GramSchmidt, but exploiting the 3-term recurrence to reduce the computational cost. We have $p_0(x) = 1$, which we see is orthogonal:

$$\|p_0\|^2 = \langle p_0, p_0 \rangle = \int_0^1 \mathrm{dx} = 1.$$

We know from the 3-term recurrence that

$$xp_0(x) = a_0 p_0(x) + p_1(x)$$

where

$$a_0 = \frac{\langle p_0, xp_0\rangle}{\|p_0\|^2} = \int_0^1 x\mathrm{d}x = 1/2.$$

Thus

$$\begin{aligned} p_1(x) &= xp_0(x) - a_0 p_0(x) = x - 1/2 \\ \|p_1\|^2 &= \int_0^1 (x^2 - x + 1/4)\mathrm{d}x = 1/12 \end{aligned}$$

From

$$xp_1(x) = c_0 p_0(x) + a_1 p_1(x) + p_2(x)$$

we have

$$\begin{aligned} c_0 &= \frac{\langle p_0, xp_1\rangle}{\|p_0\|^2} = \int_0^1 (x^2 - x/2)\mathrm{d}x = 1/12 \\ a_1 &= \frac{\langle p_1, xp_1\rangle}{\|p_1\|^2} = 12\int_0^1 (x^3 - x^2 + x/4)\mathrm{d}x = 1/2 \\ p_2(x) &= xp_1(x) - c_0 - a_1 p_1(x) = x^2 - x + 1/6 \\ \|p_2\|^2 &= \int_0^1 (x^4 - 2x^3 + 4x^2/3 - x/3 + 1/36)\mathrm{d}x = \frac{1}{180} \end{aligned}$$

Finally, from

$$xp_2(x) = c_1 p_1(x) + a_2 p_2(x) + p_3(x)$$

we have

$$\begin{aligned} c_1 &= \frac{\langle p_1, xp_2\rangle}{\|p_1\|^2} = 12\int_0^1 (x^4 - 3x^3/2 + 2x^2/3 - x/12)\mathrm{d}x = 1/15 \\ a_2 &= \frac{\langle p_2, xp_2\rangle}{\|p_2\|^2} = 180\int_0^1 (x^5 - 2x^4 + 4x^3/3 - x^2/3 + x/36)\mathrm{d}x = 1/2 \\ p_3(x) &= xp_2(x) - c_1 p_1(x) - a_2 p_2(x) = x^3 - x^2 + x/6 - x/15 + 1/30 - x^2/2 + x/2 - 1/12 \\ &= x^3 - 3x^2/2 + 3x/5 - 1/20 \end{aligned}$$

1.6 Jacobi matrices and multiplication by x

We can rewrite the three-term recurrence as

$$x \begin{pmatrix} p_0(x) \\ p_1(x) \\ p_2(x) \\ \vdots \end{pmatrix} = J \begin{pmatrix} p_0(x) \\ p_1(x) \\ p_2(x) \\ \vdots \end{pmatrix}$$

where J is a Jacobi matrix, an infinite-dimensional tridiagonal matrix:

$$J = \begin{pmatrix} a_0 & b_0 & & & \\ c_1 & a_1 & b_1 & & \\ & c_2 & a_2 & b_2 & \\ & & c_3 & a_3 & \ddots \\ & & & \ddots & \ddots \end{pmatrix}$$

When the polynomials are monic, we have 1 on the superdiagonal. When we have an orthonormal basis, then J is symmetric:

$$J = \begin{pmatrix} a_0 & b_0 & & & \\ b_0 & a_1 & b_1 & & \\ & b_1 & a_2 & b_2 & \\ & & b_2 & a_3 & \ddots \\ & & & \ddots & \ddots \end{pmatrix}$$

Given a polynomial expansion, J tells us how to multiply by x in coefficient space, that is, if

$$f(x) = \sum_{k=0}^{\infty} f_k p_k(x) = (p_0(x), p_1(x), \ldots) \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \end{pmatrix}$$

then (provided the relevant sums converge absolutely and uniformly)

$$xf(x) = x(p_0(x), p_1(x), \ldots) \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \end{pmatrix} = \left(J \begin{pmatrix} p_0(x) \\ p_1(x) \\ p_2(x) \\ \vdots \end{pmatrix} \right)^{\top} \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \end{pmatrix} = (p_0(x), p_1(x), \ldots) X \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \end{pmatrix}$$

where $X := J^{\top}$.

1.6.1 Jacobi matrix

The three-term recurrence can also be interpreted as a matrix known as the Jacobi matrix:

Corollary (Jacobi matrix) For

$$P(x) := [p_0(x)|p_1(x)|\cdots]$$

then we have

$$xP(x) = P(x) \underbrace{\begin{bmatrix} a_0 & c_0 & & \\ b_0 & a_1 & c_1 & \\ & b_1 & a_2 & \ddots \\ & & \ddots & \ddots \end{bmatrix}}_{X}$$

More generally, for any polynomial $a(x)$ we have

$$a(x)P(x) = P(x)a(X).$$

For the special cases of orthonormal and monic polynomials we have extra structure:

Corollary (orthonormal 3-term recurrence) If $\{q_n\}$ are orthonormal then its recurrence coefficients satisfy $c_n = b_n$. That is, the Jacobi matrix is symmetric:

$$X = \begin{bmatrix} a_0 & b_0 & & \\ b_0 & a_1 & b_1 & \\ & b_1 & a_2 & \ddots \\ & & \ddots & \ddots \end{bmatrix}$$

Proof

$$b_n = \langle xq_n, q_{n+1} \rangle = \langle q_n, xq_{n+1} \rangle = c_{n-1}.$$

■

Remark Typically the Jacobi matrix is the transpose $J := X^\top$. If the basis are orthonormal then X is symmetric and they are the same.

Remark (advanced) If you are worried about multiplication of infinite matrices/vectors note it is well-defined by the standard definition because it is banded. It can also be defined in terms of functional analysis where one considers these as linear operators (functions of functions) between vector spaces.

Remark (advanced) Every integrable weight generates a family of orthonormal polynomials, which in turn generates a symmetric Jacobi matrix. There is a "Spectral Theorem for Jacobi matrices" that says one can go the other way: every tridiagonal symmetric matrix with bounded entries is a Jacobi matrix for some integrable weight with compact support. This is an example of what Barry Simon calls a "Gem of spectral theory", that is.

Example (uniform weight Jacobi matrix) Consider the monic orthogonal polynomials $p_0(x), p_1(x), \ldots, p_3(x)$ for $w(x) = 1$ on $[0, 1]$ constructed above. We can write the 3-term recurrence coefficients we have computed above as the Jacobi matrix:

$$x[p_0(x)|p_1(x)|\cdots] = [p_0(x)|p_1(x)|\cdots] \underbrace{\begin{bmatrix} 1/2 & 1/12 & & \\ 1 & 1/2 & 1/15 & \\ & 1 & 1/2 & \ddots \\ & & \ddots & \ddots \end{bmatrix}}_{X}$$

We can compute the orthonormal polynomials, using

$$\|p_3\|^2 = \int_0^1 (x^6 - 3x^5 + 69x^4/20 - 19x^3/10 + 51x^2/100 - 3x/50 + 1/400)\mathrm{d}x = \frac{1}{2800}$$

as:

$$\begin{aligned}
q_0(x) &= p_0(x)\\
q_1(x) &= \sqrt{12}p_1(x) = \sqrt{3}(2x-1)\\
q_2(x) &= \sqrt{180}p_2(x) = \sqrt{5}(6x^2 - 6x + 1)\\
q_3(x) &= \sqrt{2800}p_3(x) = \sqrt{7}(20x^3 - 30x^2 + 12x - 1)
\end{aligned}$$

which have the Jacobi matrix

$$\begin{aligned}
x[q_0(x)|q_1(x)|\cdots] &= x[p_0(x)|p_1(x)|\cdots]\underbrace{\begin{bmatrix} 1 & & & & \\ & 2\sqrt{3} & & & \\ & & 6\sqrt{5} & & \\ & & & 20\sqrt{7} & \\ & & & & \ddots \end{bmatrix}}_{D}\\
&= [q_0(x)|q_1(x)|\cdots]D^{-1}XD = \begin{bmatrix} 1/2 & 1/\sqrt{12} & & \\ 1/\sqrt{12} & 1/2 & 1/\sqrt{15} & \\ & 1/\sqrt{15} & 1/2 & \ddots \\ & & \ddots & \ddots \end{bmatrix}
\end{aligned}$$

which is indeed symmetric. The problem sheet explores a more elegant way of doing this.

Example (expansion) Consider expanding a low degree polynomial like $f(x) = x^2$ in $p_n(x)$. We have

$$\langle p_0, f\rangle = \int_0^1 x^2\mathrm{d}x = 1/3\langle p_1, f\rangle = \int_0^1 x^2(x-1/2)\mathrm{d}x = 1/12\langle p_2, f\rangle = \int_0^1 x^2(x^2-x+1/6)\mathrm{d}x = 1/180$$

Thus we have:

$$f(x) = \frac{p_0(x)}{3} + p_1(x) + p_2(x) = [p_0(x)|p_1(x)|p_2(x)|\cdots]\begin{bmatrix} 1/3 \\ 1 \\ 1 \\ 0 \\ \vdots \end{bmatrix}$$

We multiply (using that $b_2 = 1$ for monic OPs) to deduce:

$$
\begin{aligned}
xf(x) &= x[p_0(x)|p_1(x)|p_2(x)|\cdots]\begin{bmatrix}1/3\\1\\1\\0\\\vdots\end{bmatrix}\\
&= [p_0(x)|p_1(x)|p_2(x)|\cdots]X\begin{bmatrix}1/3\\1\\1\\0\\\vdots\end{bmatrix} = [p_0(x)|p_1(x)|p_2(x)|\cdots]\begin{bmatrix}1/4\\9/10\\3/2\\1\\0\\\vdots\end{bmatrix}\\
&= \frac{p_0(x)}{4} + \frac{9p_1(x)}{10} + \frac{3p_2(x)}{2} + p_3(x)
\end{aligned}
$$

1.6.2 Evaluating polynomials

We can use the three-term recurrence to construct the polynomials. One way to express this is in the language of linear algebra. Suppose we are given $p_0(x) = k_0$ (where $k_0 = 1$ is pretty much always the case in practice). This can be written in matrix form as

$$
(1, 0, 0, 0, 0, \ldots)\begin{pmatrix}p_0(x)\\p_1(x)\\p_2(x)\\\vdots\end{pmatrix} = k_0
$$

We can combine this with the Jacobi operator to get

$$
\underbrace{\begin{pmatrix}
1 & & & & & \\
a_0 - x & b_0 & & & & \\
c_1 & a_1 - x & b_1 & & & \\
& c_2 & a_2 - x & b_2 & & \\
& & c_3 & a_3 - x & b_3 & \\
& & & \ddots & \ddots & \ddots
\end{pmatrix}}_{L_x}\begin{pmatrix}p_0(x)\\p_1(x)\\p_2(x)\\\vdots\end{pmatrix} = \begin{pmatrix}k_0\\0\\0\\\vdots\end{pmatrix}
$$

For x fixed, this is a lower triangular system, that is, the polynomials equal

$$
k_0 L_x^{-1}\mathbf{e}_0
$$

This can be solved via forward recurrence:

$$
\begin{aligned}
p_0(x) &= k_0 \\
p_1(x) &= \frac{(x-a_0)p_0(x)}{b_0} \\
p_2(x) &= \frac{(x-a_1)p_0(x) - c_1 p_0(x)}{b_1} \\
p_3(x) &= \frac{(x-a_2)p_1(x) - c_2 p_1(x)}{b_2} \\
&\vdots
\end{aligned}
$$

We can use this to evaluate functions as well:

$$
f(x) = (p_0(x), p_1(x), \ldots) \begin{pmatrix} f_0 \\ f_1 \\ \vdots \end{pmatrix} = k_0 \mathbf{e}_0^\top L_x^{-\top} \begin{pmatrix} f_0 \\ f_1 \\ \vdots \end{pmatrix}
$$

when f is a degree $n-1$ polynomial, this becomes a problem of inverting an upper triangular matrix, that is, we want to solve the $n \times n$ system

$$
\underbrace{\begin{pmatrix}
1 & a_0 - x & c_1 & & & \\
& b_0 & a_1 - x & c_2 & & \\
& & b_1 & a_2 - x & \ddots & \\
& & & b_2 & \ddots & c_{n-2} \\
& & & & \ddots & a_{n-2} - x \\
& & & & & b_{n-2}
\end{pmatrix}}_{L_x^\top}
\begin{pmatrix} \gamma_0 \\ \vdots \\ \gamma_{n-1} \end{pmatrix}
$$

so that $f(x)/k_0 = \gamma_0$. We can solve this via back-substitution:

$$
\begin{aligned}
\gamma_{n-1} &= \frac{f_{n-1}}{b_{n-2}} \\
\gamma_{n-2} &= \frac{f_{n-2} - (a_{n-2} - x)\gamma_{n-1}}{b_{n-3}} \\
\gamma_{n-3} &= \frac{f_{n-3} - (a_{n-3} - x)\gamma_{n-2} - c_{n-2}\gamma_{n-1}}{b_{n-4}} \\
&\vdots \\
\gamma_1 &= \frac{f_1 - (a_1 - x)\gamma_2 - c_2\gamma_3}{b_0} \\
\gamma_0 &= f_0 - (a_0 - x)\gamma_1 - c_1\gamma_2
\end{aligned}
$$

We give examples of these algorithms applied to Chebyshev polynomials in the next lecture.

1.7 GramSchmidt algorithm

In general we don't have nice formulae for OPs but we can always construct them via the GramSchmidt process:

Proposition (GramSchmidt) Define

$$\begin{aligned} p_0(x) &= 1 \\ q_0(x) &= \frac{1}{\|p_0\|} \\ p_{n+1}(x) &= xq_n(x) - \sum_{k=0}^{n} \langle xq_n, q_k\rangle q_k(x) \\ q_{n+1}(x) &= \frac{p_{n+1}(x)}{\|p_{n+1}\|} \end{aligned}$$

Then $q_0(x), q_1(x), \ldots$ are orthonormal w.r.t. w.

Proof By linearity we have

$$\langle p_{n+1}, q_j\rangle = \left\langle xq_n - \sum_{k=0}^{n} \langle xq_n, q_k\rangle q_k, q_j \right\rangle = \langle xq_n, q_j\rangle - \langle xq_n, q_j\rangle\langle q_j, q_j\rangle = 0$$

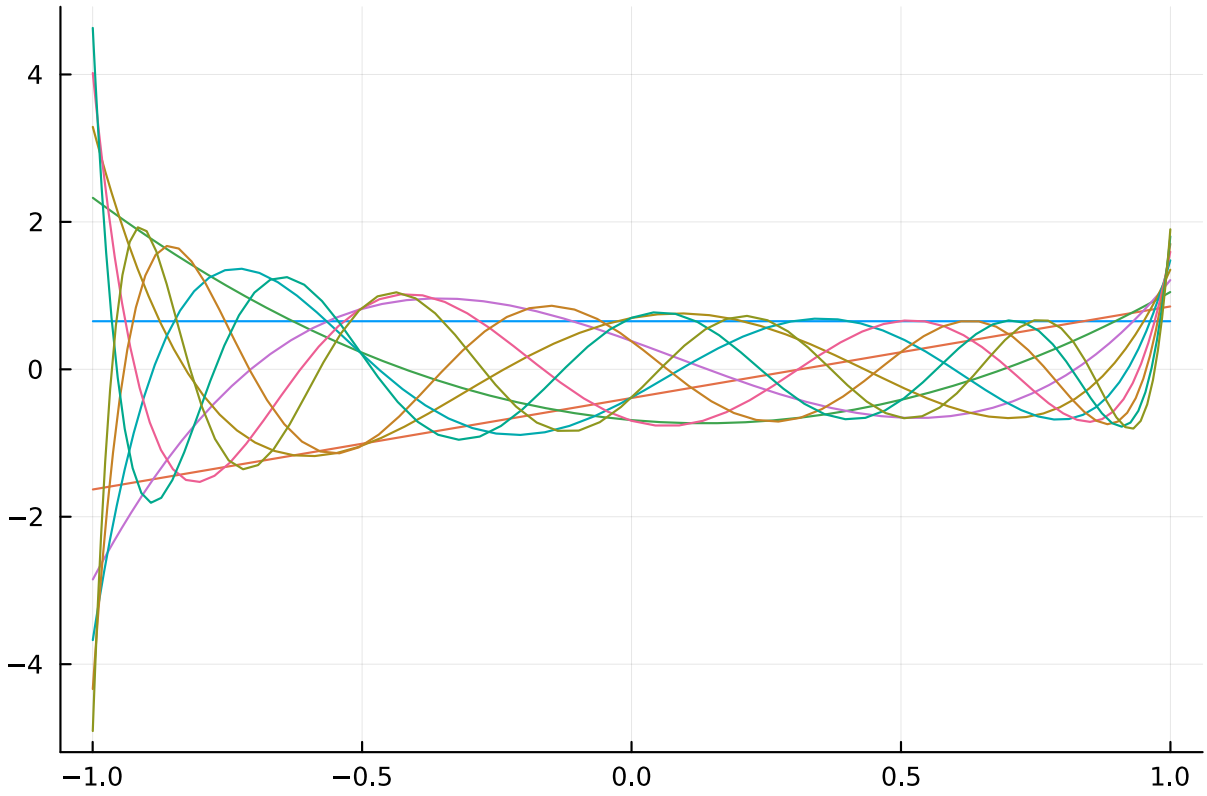
Thus p_{n+1} is orthogonal to all lower degree polynomials. So is q_{n+1}, since it is a constant multiple of p_{n+1}.

■

Let's make our own family of OPs:

```
using ApproxFun, Plots
x = Fun()
w = exp(x)
ip = (f,g) -> sum(f*g*w)
nrm = f    -> sqrt(ip(f,f))
n = 10
q = Array{Fun}(undef,n)
p = Array{Fun}(undef,n)
p[1] = Fun(1, -1 .. 1 )
q[1] = p[1]/nrm(p[1])

for k=1:n-1
    p[k+1] = x*q[k]
    for j=1:k
        p[k+1] -= ip(p[k+1],q[j])*q[j]
    end
    q[k+1] = p[k+1]/nrm(p[k+1])
end

@show sum(q[2]*q[4]*w)

p = plot(; legend=false)
for k=1:10
    plot!(q[k])
end
p
```

```
sum(q[2] * q[4] * w) = -9.920449878242366e-17
```



The three-term recurrence means we can simplify GramSchmidt, and calculate the recurrence coefficients at the same time:

Proposition (GramSchmidt) Define

$$
\begin{aligned}
p_0(x) &= 1 \\
q_0(x) &= \frac{1}{\|p_0\|} \\
a_n &= \langle x q_n, q_n \rangle \\
b_{n-1} &= \langle x q_n, q_{n-1} \rangle \\
p_{n+1}(x) &= x q_n(x) - a_n q_n(x) - b_{n-1} q_{n-1}(x) \\
q_{n+1}(x) &= \frac{p_{n+1}(x)}{\|p_{n+1}\|}
\end{aligned}
$$

Then $q_0(x), q_1(x), \ldots$ are orthonormal w.r.t. w.

Remark This can be made a bit more efficient by using $\|p_{n+1}\|$ to calculate b_n.

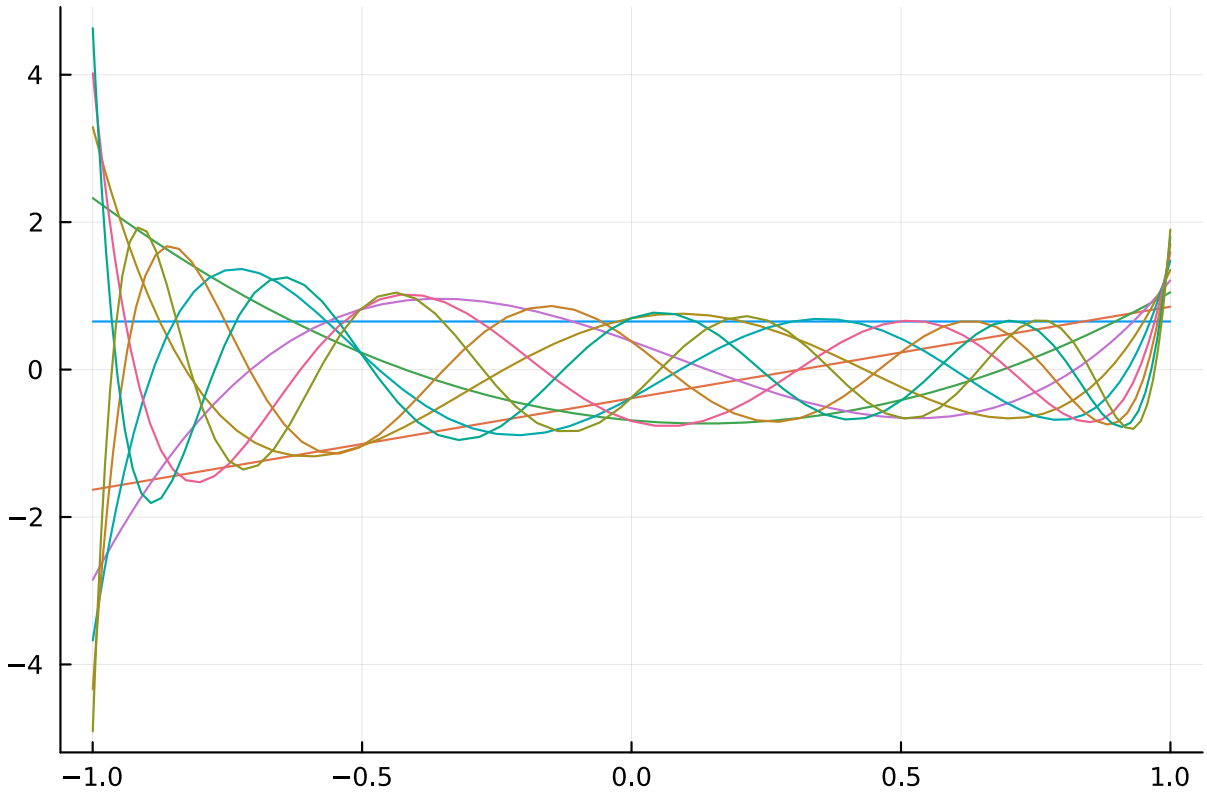
```
x = Fun()
w = exp(x)
ip = (f,g) -> sum(f*g*w)
nrm = f    -> sqrt(ip(f,f))
n = 10
q = Array{Fun}(undef, n)
p = Array{Fun}(undef, n)
a = zeros(n)
b = zeros(n)
p[1] = Fun(1, -1 .. 1 )
q[1] = p[1]/nrm(p[1])
```

```
p[2] = x*q[1]
a[1] = ip(p[2],q[1])
p[2] -= a[1]*q[1]
q[2] = p[2]/nrm(p[2])

for k=2:n-1
    p[k+1] = x*q[k]
    b[k-1] =ip(p[k+1],q[k-1])
    a[k] = ip(p[k+1],q[k])
    p[k+1] = p[k+1] - a[k]q[k] - b[k-1]q[k-1]
    q[k+1] = p[k+1]/nrm(p[k+1])
end

ip(q[5],q[2]) # shows orthogonality (to numerical accuracy)
```

```
8.118505867571457e-16
```

Here we see a plot of the first 10 polynomials:

```
p = plot(; legend=false)
for k=1:10
    plot!(q[k])
end
p
```



1.8 2. Classical orthogonal polynomials

Classical orthogonal polynomials are special families of orthogonal polynomials with a number of beautiful properties, for example

1. Their derivatives are also OPs
2. They are eigenfunctions of simple differential operators

As stated above orthogonal polynomials are uniquely defined by the weight $w(x)$ and the constant k_n. We consider:

1. Chebyshev polynomials (1st kind) $T_n(x)$: $w(x) = 1/\sqrt{1-x^2}$ on $[-1, 1]$.
2. Chebyshev polynomials (2nd kind) $U_n(x)$: $\sqrt{1-x^2}$ on $[-1, 1]$.
3. Legendre polynomials $P_n(x)$: $w(x) = 1$ on $[-1, 1]$.
4. Ultrapsherical polynomials (my fav!): $C_n^{(\lambda)}(x)$: $w(x) = (1-x^2)^{\lambda-1/2}$ on $[-1, 1]$, $\lambda \neq 0$, $\lambda > -1/2$.
5. Jacobi polynomials: $P_n^{(a,b)}(x)$: $w(x) = (1-x)^a(1+x)^b$ on $[-1, 1]$, $a, b > -1$.
6. Laguerre polynomials: $L_n(x)$: $w(x) = \exp(-x)$ on $[0, \infty)$.
7. Hermite polynomials $H_n(x)$: $w(x) = \exp(-x^2)$ on $(-\infty, \infty)$.

In the notes we will discuss properties of $T_n(x)$ and $P_n(x)$.

1.8.1 Chebyshev

Definition (Chebyshev polynomials, 1st kind) $T_n(x)$ are orthogonal with respect to $1/\sqrt{1-x^2}$ and satisfy:

$$\begin{aligned} T_0(x) &= 1, \\ T_n(x) &= 2^{n-1}x^n + O(x^{n-1}) \end{aligned}$$

Definition (Chebyshev polynomials, 2nd kind) $U_n(x)$ are orthogonal with respect to $\sqrt{1-x^2}$.

$$U_n(x) = 2^n x^n + O(x^{n-1})$$

Theorem (Chebyshev T are cos)

$$T_n(x) = \cos n \operatorname{acos} x$$

In other words

$$T_n(\cos(\theta)) = \cos n\theta.$$

Proof

We need to show that $p_n(x) := \cos n \operatorname{acos} x$ are

1. graded polynomials

2. orthogonal w.r.t. $1/\sqrt{1-x^2}$ on $[-1,1]$, and
3. have the right normalisation constant $k_n = 2^{n-1}$ for $n = 1, \ldots$.

Property (2) follows under a change of variables:

$$\langle p_n, p_m \rangle = \int_{-1}^{1} \frac{p_n(x)p_m(x)}{\sqrt{1-x^2}}\mathrm{d}x = \int_{-\pi}^{\pi} \frac{\cos(n\theta)\cos(m\theta)}{\sqrt{1-\cos^2\theta}}\sin\theta\mathrm{d}\theta = \int_{-\pi}^{\pi} \cos(n\theta)\cos(m\theta)\mathrm{d}x = 0$$

if $n \neq m$.

To see that they are graded we use the fact that

$$xp_n(x) = \cos\theta\cos n\theta = \frac{\cos(n-1)\theta + \cos(n+1)\theta}{2} = \frac{p_{n-1}(x) + p_{n+1}(x)}{2}$$

In other words $p_{n+1}(x) = 2xp_n(x) - p_{n-1}(x)$ for $n \geq 1$. Since each time we multiply by $2x$ and $p_0(x) = 1$, $p_1(x) = x$ we have

$$p_{n+1}(x) = (2x)^n + O(x^{n-1})$$

which completes the proof.

■

Buried in the proof is the 3-term recurrence:

Corollary

$$\begin{aligned} xT_0(x) &= T_1(x) \\ xT_n(x) &= \frac{T_{n-1}(x) + T_{n+1}(x)}{2} \end{aligned}$$

Chebyshev polynomials are particularly powerful as their expansions are cosine series in disguise: for

$$f(x) = \sum_{k=0}^{\infty} \check{f}_k T_k(x)$$

we have

$$f(\cos\theta) = \sum_{k=0}^{\infty} \check{f}_k \cos k\theta.$$

Thus the coefficients can be recovered fast using FFT-based techniques as discussed in the problem sheet.

In the problem sheet we will also show the following:

Theorem (Chebyshev U are sin) For $x = \cos\theta$,

$$U_n(x) = \frac{\sin(n+1)\theta}{\sin\theta}$$

which satisfy:

$$xU_0(x) = U_1(x)/2$$
$$xU_n(x) = \frac{U_{n-1}(x)}{2} + \frac{U_{n+1}(x)}{2}.$$

1.8.2 Legendre

Definition (Legendre) Legendre polynomials $P_n(x)$ are orthogonal polynomials with respect to $w(x) = 1$ on $[-1, 1]$, with

$$k_n = \frac{1}{2^n}\binom{2n}{n} = \frac{(2n)!}{2^n(n!)^2}$$

The reason for this complicated normalisation constant is both historical and that it leads to simpler formulae for recurrence relationships.

Classical orthogonal polynomials have *Rodriguez formulae*, defining orthogonal polynomials as high order derivatives of simple functions. In this case we have:

Lemma (Legendre Rodriguez formula)

$$P_n(x) = \frac{1}{(-2)^n n!}\frac{\mathrm{d}^n}{\mathrm{d}x^n}(1-x^2)^n$$

Proof We need to verify:

1. graded polynomials
2. orthogonal to all lower degree polynomials on $[-1, 1]$, and
3. have the right normalisation constant $k_n = \frac{1}{2^n}\binom{2n}{n}$.

(1) follows since its a degree n polynomial (the n-th derivative of a degree $2n$ polynomial). (2) follows by integration by parts. Note that $(1-x^2)^n$ and its first $n-1$ derivatives vanish at ± 1. If r_m is a degree $m < n$ polynomial we have:

$$\int_{-1}^{1}\frac{\mathrm{d}^n}{\mathrm{d}x^n}(1-x^2)^n r_m(x)\mathrm{d}x = -\int_{-1}^{1}\frac{\mathrm{d}^{n-1}}{\mathrm{d}x^{n-1}}(1-x^2)^n r_m'(x)\mathrm{d}x = \cdots = (-)^n\int_{-1}^{1}(1-x^2)r_m^{(n)}(x)\mathrm{d}x = 0.$$

(3) follows since:

$$\begin{aligned}\frac{\mathrm{d}^n}{\mathrm{d}x^n}[(-)^n x^{2n} + O(x^{2n-1})] &= (-)^n 2n\frac{\mathrm{d}^{n-1}}{\mathrm{d}x^{n-1}}x^{2n-1} + O(x^{2n-1})] \\ &= (-)^n 2n(2n-1)\frac{\mathrm{d}^{n-2}}{\mathrm{d}x^{n-2}}x^{2n-2} + O(x^{2n-2})] = \cdots \\ &= (-)^n 2n(2n-1)\cdots(n+1)x^n + O(x^{n-1}) = (-)^n\frac{(2n)!}{n!}x^n + O(x^{n-1})\end{aligned}$$

■

This allows us to determine the coefficients $k_n^{(\lambda)}$ which are useful in proofs. In particular we will use $k_n^{(1)}$:

Lemma (Legendre monomial coefficients)

$$\begin{aligned}
P_0(x) &= 1 \\
P_1(x) &= x \\
P_n(x) &= \underbrace{\frac{(2n)!}{2^n(n!)^2}}_{k_n} x^n - \underbrace{\frac{(2n-2)!}{2^n(n-2)!(n-1)!}}_{k_n^{(2)}} x^{n-2} + O(x^{n-4})
\end{aligned}$$

(Here the $O(x^{n-4})$ is as $x \to \infty$, which implies that the term is a polynomial of degree $\leq n-4$. For $n = 2, 3$ the $O(x^{n-4})$ term is therefore precisely zero.)

Proof

The $n = 0$ and 1 case are immediate. For the other case we expand $(1 - x^2)^n$ to get:

$$\begin{aligned}
(-)^n \frac{\mathrm{d}^n}{\mathrm{d}x^n}(1-x^2)^n &= \frac{\mathrm{d}^n}{\mathrm{d}x^n}[x^{2n} - nx^{2n-2} + O(x^{2n-4})] \\
&= (2n)\cdots(2n-n+1)x^n - n(2n-2)\cdots(2n-2-n+1)x^{n-2} + O(x^{n-4}) \\
&= \frac{(2n)!}{n!}x^n - \frac{n(2n-2)!}{(n-2)!}x^{n-2} + O(x^{n-4})
\end{aligned}$$

Multiplying through by $\frac{1}{2^n(n!)}$ completes the derivation.

■

Theorem (Legendre 3-term recurrence)

$$\begin{aligned}
xP_0(x) &= P_1(x) \\
(2n+1)xP_n(x) &= nP_{n-1}(x) + (n+1)P_{n+1}(x)
\end{aligned}$$

Proof The $n = 0$ case is immediate (since $w(x) = w(-x)$ $a_n = 0$, from PS8). For the other cases we match terms:

$$(2n+1)xP_n(x) - nP_{n-1}(x) - (n+1)P_{n+1}(x) = [(2n+1)k_n - (n+1)k_{n+1}]x^{n+1} + [(2n+1)k_n^{(1)} - nk_{n-1} - (n+1)k_{n+1}^{(1)}$$

Using the expressions for k_n and $k_n^{(1)}$ above we have (leaving the manipulations as an exercise):

$$(2n+1)k_n - (n+1)k_{n+1} = \frac{(2n+1)!}{2^n(n!)^2} - (n+1)\frac{(2n+2}{2(n+1)((n-}$$

$$(2n+1)k_n^{(1)} - nk_{n-1} - (n+1)k_{n+1}^{(1)} = -(2n+1)\frac{(2n-2)!}{2^n(n-2)!(n-1)!} - n\frac{(2n-2)!}{2^{n-1}((n-1)!)^2} + (n+1)\frac{(2n}{2^{n+1}(n-}$$

Thus

$$(2n+1)xP_n(x) - nP_{n-1}(x) - (n+1)P_{n+1}(x) = O(x^{n-3})$$

But as it is orthogonal to $P_k(x)$ for $0 \le k \le n-3$ it must be zero. ■

We will also investigate the properties of *classical* OPs. A good reference is Digital Library of Mathematical Functions, Chapter 18.

This lecture we discuss

1. Hermite, Laguerre, and Jacobi polynomials
2. Legendre, Chebyshev, and ultraspherical polynomials
3. Explicit construction for Chebyshev polynomials

1.9 Definition of classical orthogonal polynomials

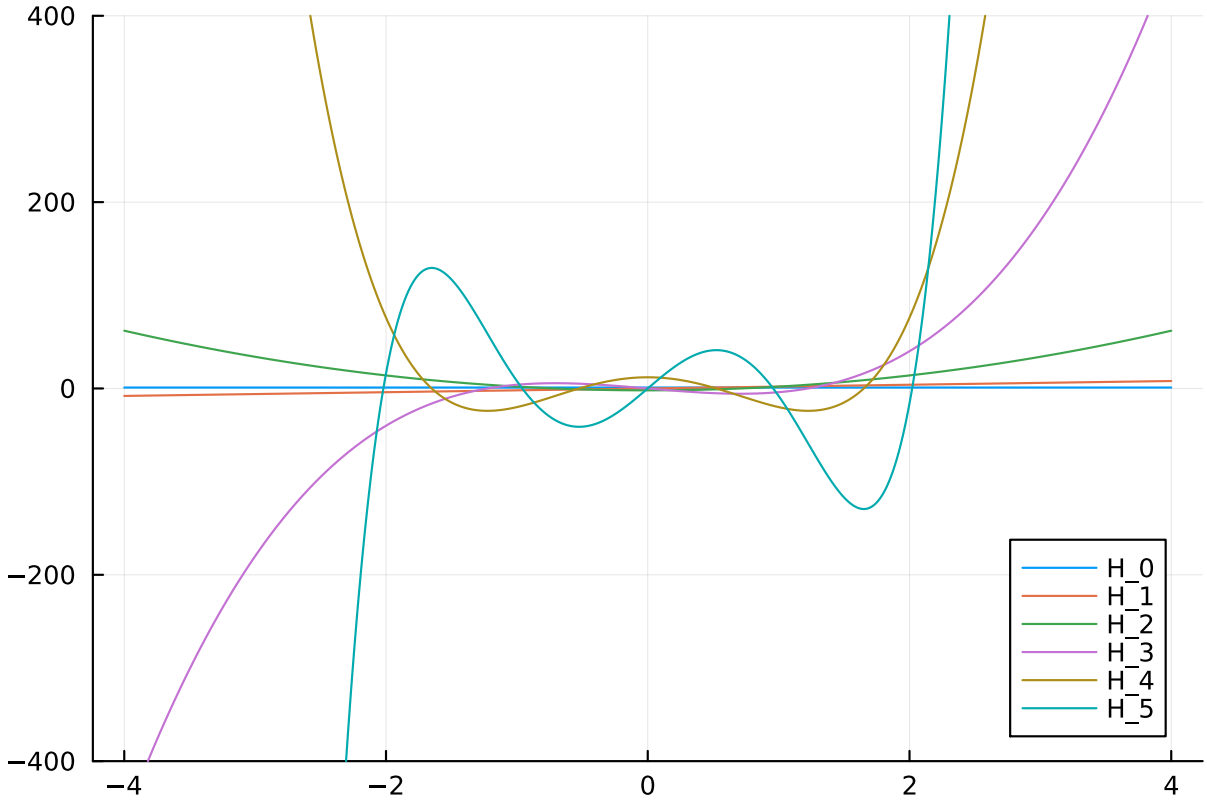
Classical orthogonal polynomials are orthogonal with respect to the following three weights:

Name	(a,b)	$w(x)$	Notation	k_n
Hermite	$(-\infty,\infty)$	e^{-x^2}	$H_n(x)$	2^n
Laguerre	$(0,\infty)$	$x^\alpha \mathrm{e}^{-x}$	$L_n^{(\alpha)}(x)$	Table 18.3.1
Jacobi	$(-1,1)$	$(1-x)^\alpha(1+x)^\beta$	$P_n^{(\alpha,\beta)}(x)$	Table 18.3.1

Note out of convention the parameters for Jacobi polynomials are right-to-left order.

We can actually construct these polynomials in Julia, first consider Hermite:

```
using ApproxFun, Plots, LinearAlgebra, ComplexPhasePortrait
H_0 = Fun(Hermite(), [1])
H_1 = Fun(Hermite(), [0,1])
H_2 = Fun(Hermite(), [0,0,1])
H_3 = Fun(Hermite(), [0,0,0,1])
H_4 = Fun(Hermite(), [0,0,0,0,1])
H_5 = Fun(Hermite(), [0,0,0,0,0,1])

xx = -4:0.01:4
plot(xx, H_0.(xx); label="H_0", ylims=(-400,400))
plot!(xx, H_1.(xx); label="H_1", ylims=(-400,400))
plot!(xx, H_2.(xx); label="H_2", ylims=(-400,400))
plot!(xx, H_3.(xx); label="H_3", ylims=(-400,400))
plot!(xx, H_4.(xx); label="H_4", ylims=(-400,400))
plot!(xx, H_5.(xx); label="H_5")
```



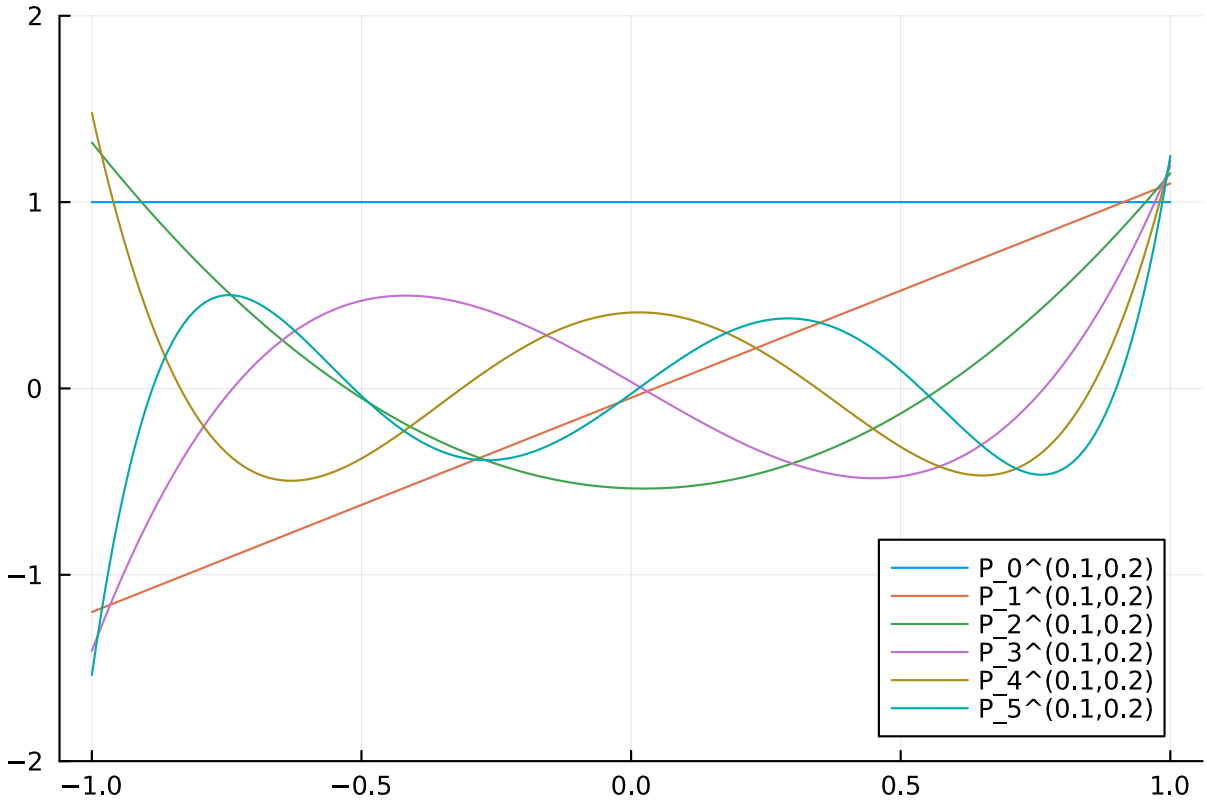
We verify their orthogonality:

```
w = Fun(GaussWeight(), [1.0])

@show sum(H_2*H_5*w)  # means integrate
@show sum(H_5*H_5*w);
```

```
sum(H_2 * H_5 * w) = 0.0
sum(H_5 * H_5 * w) = 6806.222787477181
```

Now Jacobi:

```
α,β = 0.1,0.2
P_0 = Fun(Jacobi(β,α), [1])
P_1 = Fun(Jacobi(β,α), [0,1])
P_2 = Fun(Jacobi(β,α), [0,0,1])
P_3 = Fun(Jacobi(β,α), [0,0,0,1])
P_4 = Fun(Jacobi(β,α), [0,0,0,0,1])
P_5 = Fun(Jacobi(β,α), [0,0,0,0,0,1])

xx = -1:0.01:1
plot( xx, P_0.(xx); label="P_0^($α,$β)", ylims=(-2,2))
plot!(xx, P_1.(xx); label="P_1^($α,$β)")
plot!(xx, P_2.(xx); label="P_2^($α,$β)")
plot!(xx, P_3.(xx); label="P_3^($α,$β)")
plot!(xx, P_4.(xx); label="P_4^($α,$β)")
plot!(xx, P_5.(xx); label="P_5^($α,$β)")
```




```
w = Fun(JacobiWeight(β,α), [1.0])
@show sum(P_2*P_5*w)  # means integrate
@show sum(P_5*P_5*w);

sum(P_2 * P_5 * w) = 1.0328132900334873e-17
sum(P_5 * P_5 * w) = 0.21713358248393153
```

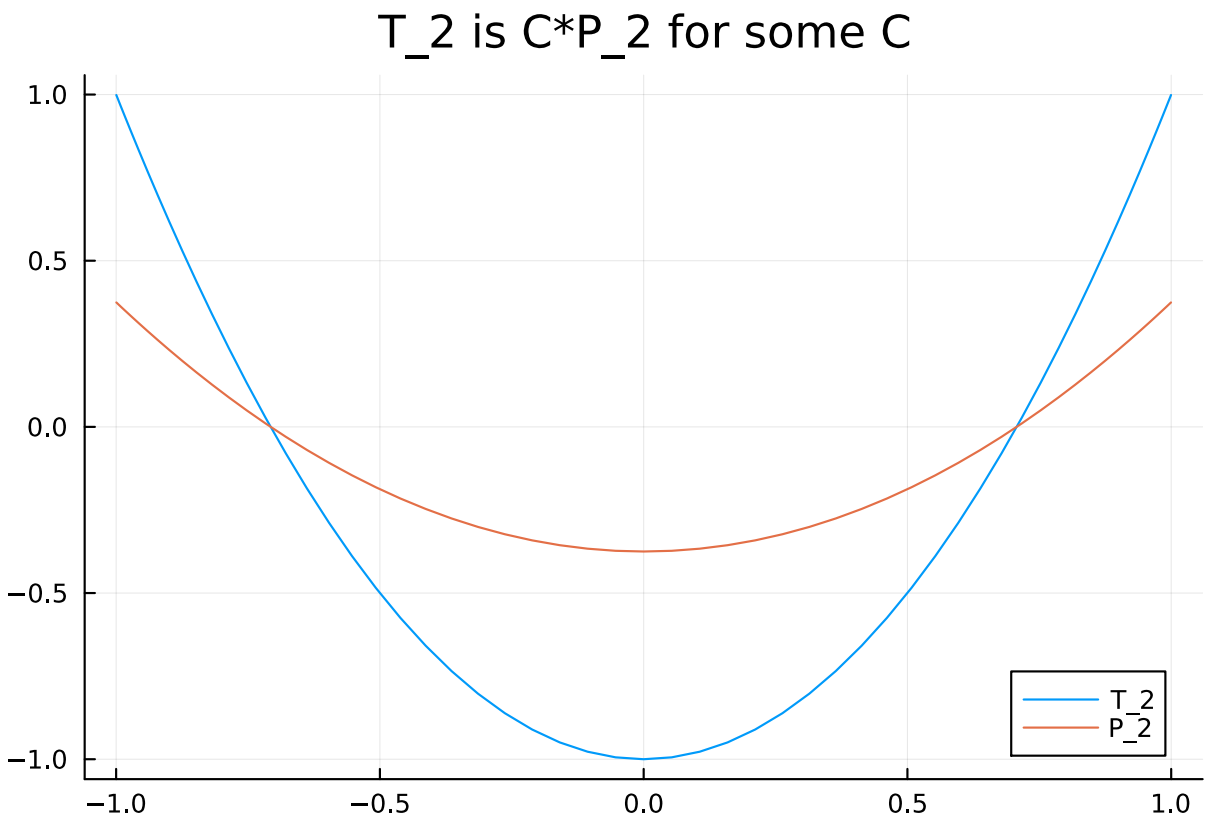
1.10 Legendre, Chebyshev, and ultraspherical polynomials

There are special families of Jacobi weights with their own name.

Name	Jacobi parameters	$w(x)$	Notation	k_n
Jacobi	α, β	$(1-x)^\alpha(1+x)^\beta$	$P_n^{(\alpha,\beta)}(x)$	Table 18.3.1
Legendre	$0, 0$	1	$P_n(x)$	$2^n(1/2)_n/n!$
Chebyshev (1st)	$-\frac{1}{2}, -\frac{1}{2}$	$\frac{1}{\sqrt{1-x^2}}$	$T_n(x)$	$1(n=0), 2^{n-1}(n\neq 0)$
Chebyshev (2nd)	$\frac{1}{2}, \frac{1}{2}$	$\sqrt{1-x^2}$	$U_n(x)$	2^n
Ultraspherical	$\lambda-\frac{1}{2}, \lambda-\frac{1}{2}$	$(1-x^2)^{\lambda-1/2}, \lambda\neq 0$	$C_n^{(\lambda)}(x)$	$2^n(\lambda)_n/n!$

Note that other than Legendre, these polynomials have a different normalization than $P_n^{(\alpha,\beta)}$:

```
T_2 = Fun(Chebyshev(), [0.0,0,1])
P_2 = Fun(Jacobi(-1/2,-1/2), [0.0,0,1])
plot(T_2; label="T_2", title="T_2 is C*P_2 for some C")
plot!(P_2; label="P_2")
```



But because they are orthogonal w.r.t. the same weight, they must be a constant multiple of each-other, as discussed last lecture.

1.10.1 Explicit construction of Chebyshev polynomials (first kind and second kind)

Chebyshev polynomials are pretty much the only OPs with *simple* closed form expressions.

Proposition (Chebyshev first kind formula) $T_n(x) = \cos n \operatorname{acos} x$ or in other words,

$$T_n(\cos\theta) = \cos n\theta$$

Proof We first show that they are orthogonal w.r.t. $1/\sqrt{1-x^2}$. Too easy: do $x = \cos\theta$, $\mathrm{d}x = -\sin\theta \mathrm{d}\theta$ to get (for $n \neq m$)

$$\int_{-1}^{1} \frac{\cos n \operatorname{acos} x \cos m \operatorname{acos} x \mathrm{d}x}{\sqrt{1-x^2}} = -\int_{\pi}^{0} \cos n\theta \cos m\theta \mathrm{d}\theta = \int_0^{\pi} \frac{\mathrm{e}^{\mathrm{i}(-n-m)\theta} + \mathrm{e}^{\mathrm{i}(n-m)\theta} + \mathrm{e}^{\mathrm{i}(m-n)\theta} + \mathrm{e}^{\mathrm{i}(n+m)\theta}}{4} \mathrm{d}\theta =$$

We then need to show it has the right highest order term k_n. Note that $k_0 = k_1 = 1$. Using $z = \mathrm{e}^{\mathrm{i}\theta}$ we see that $\cos n\theta$ has a simple recurrence for $n = 2, 3, \ldots$:

$$\cos n\theta = \frac{z^n + z^{-n}}{2} = 2\frac{z + z^{-1}}{2}\frac{z^{n-1} + z^{1-n}}{2} - \frac{z^{n-2} + z^{2-n}}{2} = 2\cos\theta\cos(n-1)\theta - \cos(n-2)\theta$$

thus

$$\cos n \operatorname{acos} x = 2x \cos(n-1) \operatorname{acos} x - \cos(n-2) \operatorname{acos} x$$

It follows that

$$k_n = 2k_{n-1} = 2^{n-1}k_1 = 2^{n-1}$$

By uniqueness we have $T_n(x) = \cos n \operatorname{acos} x$.

■

Proposition (Chebyshev second kind formula) $U_n(x) = \frac{\sin(n+1)\operatorname{acos} x}{\sin \operatorname{acos} x}$ or in other words,

$$U_n(\cos\theta) = \frac{\sin(n+1)\theta}{\sin\theta}$$

Example For the case of Chebyshev polynomials, we have

$$J = \begin{pmatrix} 0 & 1 & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ & \frac{1}{2} & 0 & \frac{1}{2} & \\ & & \frac{1}{2} & 0 & \ddots \\ & & & \ddots & \ddots \end{pmatrix}$$

Therefore, the Chebyshev coefficients of $xf(x)$ are given by

$$J^\top \mathbf{f} = \begin{pmatrix} 0 & \frac{1}{2} & & & \\ 1 & 0 & \frac{1}{2} & & \\ & \frac{1}{2} & 0 & \frac{1}{2} & \\ & & \frac{1}{2} & 0 & \ddots \\ & & & \ddots & \ddots \end{pmatrix} \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ \vdots \end{pmatrix}$$

1.10.2 Demonstration

In the case where f is a degree $n-1$ polynomial, we can represent $J^\top$ as an $n+1 \times n$ matrix (this makes sense as $xf(x)$ is one more degree than f):

```
f = Fun(exp, Chebyshev())
n = ncoefficients(f) # number of coefficients
@show n
J = zeros(n,n+1)
J[1,2] = 1
for k=2:n
    J[k,k-1] = J[k,k+1] = 1/2
end
J'
```

```
n = 14
15×14 adjoint(::Matrix{Float64}) with eltype Float64:
 0.0  0.5  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 1.0  0.0  0.5  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.5  0.0  0.5  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.5  0.0  0.5  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.5  0.0  0.5  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.5  0.0  0.5  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.5  0.0  0.5  0.0  0.0  0.0  0.0  0.0  0.0
```

```
 0.0  0.0  0.0  0.0  0.0  0.0  0.5  0.0  0.5  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.5  0.0  0.5  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.5  0.0  0.5  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.5  0.0  0.5  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.5  0.0  0.5  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.5  0.0  0.5
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.5  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.5
```

```
cfs = J'*f.coefficients # coefficients of x*f
xf = Fun(Chebyshev(), cfs)

xf(0.1) - 0.1*f(0.1)
```

```
-6.938893903907228e-17
```

We can construct $T_0(x), \ldots, T_{n-1}(x)$ via

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= xT_0(x) \\ T_{k+1}(x) &= 2xT_k(x) - T_{k-1}(x), \qquad 1 \le k \le n-2 \end{aligned}$$

Believe it or not, this is much faster than using $\cos k \operatorname{acos} x$:

```
function recurrence_Chebyshev(n,x)
    T = zeros(n)
    T[1] = 1.0
    T[2] = x*T[1]
    for k = 2:n-1
        T[k+1] = 2x*T[k] - T[k-1]
    end
    T
end

trig_Chebyshev(n,x) = [cos(k*acos(x)) for k=0:n-1]

n = 10
recurrence_Chebyshev(n, 0.1) - trig_Chebyshev(n,0.1) |>norm
```

```
1.1102230246251565e-16
```

```
n = 10000
@time recurrence_Chebyshev(n, 0.1)
@time trig_Chebyshev(n,0.1);
```

```
0.000111 seconds (2 allocations: 78.172 KiB)
  0.000329 seconds (2 allocations: 78.172 KiB)
```

We can also demonstrate Clenshaw's algorithm for evaluating polynomials. To evaluate an expansion in Chebyshev polynomials,

$$\sum_{k=0}^{n-1} f_k T_k(x)$$

we want to solve the system

$$\underbrace{\begin{pmatrix} 1 & -x & \frac{1}{2} & & & \\ & 1 & -x & \frac{1}{2} & & \\ & & \frac{1}{2} & -x & \ddots & \\ & & & \frac{1}{2} & \ddots & \frac{1}{2} \\ & & & & \ddots & -x \\ & & & & & \frac{1}{2} \end{pmatrix}}_{L_x^\top} \begin{pmatrix} \gamma_0 \\ \vdots \\ \gamma_{n-1} \end{pmatrix}$$

via

$$\begin{aligned} \gamma_{n-1} &= 2f_{n-1} \\ \gamma_{n-2} &= 2f_{n-2} + 2x\gamma_{n-1} \\ \gamma_{n-3} &= 2f_{n-3} + 2x\gamma_{n-2} - \gamma_{n-1} \\ &\vdots \\ \gamma_1 &= f_1 + x\gamma_2 - \frac{1}{2}\gamma_3 \\ \gamma_0 &= f_0 + x\gamma_1 - \frac{1}{2}\gamma_2 \end{aligned}$$

then $f(x) = \gamma_0$.

```
function clenshaw_Chebyshev(f,x)
    n = length(f)
    γ = zeros(n)
    γ[n] = 2f[n]
    γ[n-1] = 2f[n-1] +2x*f[n]
    for k = n-2:-1:1
        γ[k] = 2f[k] + 2x*γ[k+1] - γ[k+2]
    end
    γ[2] = f[2] + x*γ[3] - γ[4]/2
    γ[1] = f[1] + x*γ[2] - γ[3]/2
    γ[1]
end

f = Fun(exp, Chebyshev())
clenshaw_Chebyshev(f.coefficients, 0.1) - exp(0.1)
```

```
-1.3322676295501878e-15
```

With some high performance computing tweaks, this can be made more accurate. This is the algorithm used for evaluating functions in ApproxFun:

```
f(0.1) - exp(0.1)
```

```
0.0
```

2 Interpolation and Gaussian quadrature

Polynomial interpolation is the process of finding a polynomial that equals data at a precise set of points. *Quadrature* is the act of approximating an integral by a weighted sum:

$$\int_a^b f(x)w(x)\mathrm{d}x \approx \sum_{j=1}^n w_j f(x_j).$$

In these notes we see that the two concepts are intrinsically linked: interpolation leads naturally to quadrature rules. Moreover, by using a set of points x_j linked to orthogonal polynomials we get significantly more accurate rules, and in fact can use quadrature to compute expansions in orthogonal polynomials that interpolate. That is, we can mirror the link between the Trapezium rule, Fourier series, and interpolation but now for polynomials.

1. Polynomial Interpolation: we describe how to interpolate a function by a polynomial.
2. Truncated Jacobi matrices: we see that truncated Jacobi matrices are diagonalisable

in terms of orthogonal polynomials and their zeros.

2. Interpolatory quadrature rule: polynomial interpolation leads naturally to ways to integrate

functions, but onely realisable in the simplest cases.

3. Gaussian quadrature: Using roots of orthogonal polynomials and truncated Jacobi matrices

leads naturally to an efficiently computable interpolatory quadrature rule. The *miracle* is its exact for twice as many polynomials as expected.

2.1 1. Polynomial Interpolation

We already saw a special case of polynomial interpolation, where we saw that the polynomial

$$f(z) \approx \sum_{k=0}^{n-1} \hat{f}_k^n z^k$$

equaled f at evenly spaced points on the unit circle: $\mathrm{e}^{\mathrm{i}2\pi j/n}$. But here we consider the following:

Definition (interpolatory polynomial) Given n distinct points $x_1, \ldots, x_n \in \mathbb{R}$ and n *samples* $f_1, \ldots, f_n \in \mathbb{R}$, a degree $n-1$ *interpolatory polynomial* $p(x)$ satisfies

$$p(x_j) = f_j$$

The easiest way to solve this problem is to invert the Vandermonde system:

Definition (Vandermonde) The *Vandermonde matrix* associated with n distinct points $x_1, \ldots, x_n \in \mathbb{R}$ is the matrix

$$V := \begin{bmatrix} 1 & x_1 & \cdots & x_1^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & x_n^{n-1} \end{bmatrix}$$

Proposition (interpolatory polynomial uniqueness) The interpolatory polynomial is unique, and the Vandermonde matrix is invertible.

Proof Suppose p and $\tilde{p}$ are both interpolatory polynomials. Then $p(x) - \tilde{p}(x)$ vanishes at n distinct points x_j. By the fundamental theorem of algebra it must be zero, i.e., $p = \tilde{p}$.

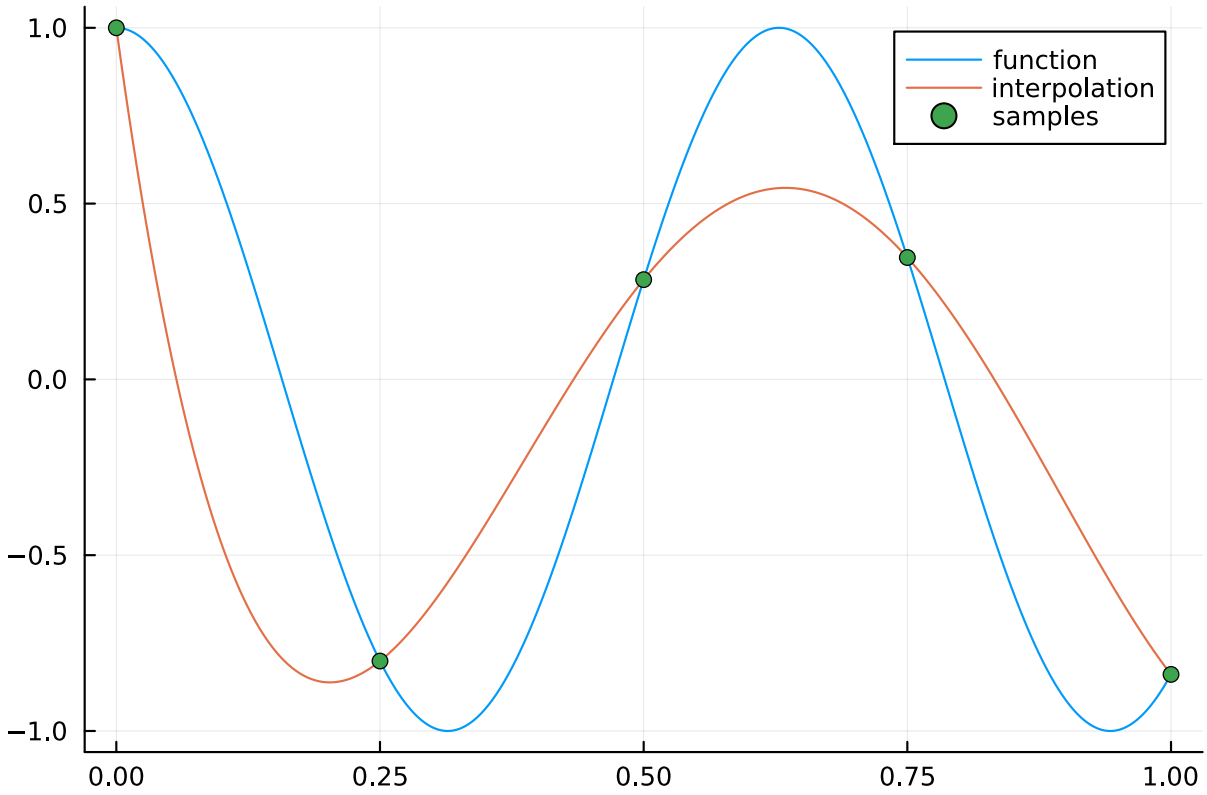
For the second part, if $V\mathbf{c} = 0$ for $\mathbf{c} \in \mathbb{R}$ then for $q(x) = c_1 + \cdots + c_n x^{n-1}$ we have

$$q(x_j) = \mathbf{e}_j^\top V\mathbf{c} = 0$$

hence q vanishes at n distinct points and is therefore 0, i.e., $\mathbf{c} = 0$. ■

Thus a quick-and-dirty way to to do interpolation is to invert the Vandermonde matrix (which we saw in the least squares setting with more samples then coefficients):

```
using Plots, LinearAlgebra
f = x -> cos(10x)
n = 5

x = range(0, 1; length=n)# evenly spaced points (BAD for interpolation)
V = x .^ (0:n-1)' # Vandermonde matrix
c = V \ f.(x) # coefficients of interpolatory polynomial
p = x -> dot(c, x .^ (0:n-1))

g = range(0,1; length=1000) # plotting grid
plot(g, f.(g); label="function")
plot!(g, p.(g); label="interpolation")
scatter!(x, f.(x); label="samples")
```



But it turns out we can also construct the interpolatory polynomial directly. We will use the following with equal 1 at one grid point and are zero at the others:

Definition (Lagrange basis polynomial) The *Lagrange basis polynomial* is defined as

$$\ell_k(x) := \prod_{j \neq k} \frac{x - x_j}{x_k - x_j} = \frac{(x - x_1) \cdots (x - x_{k-1})(x - x_{k+1}) \cdots (x - x_n)}{(x_k - x_1) \cdots (x_k - x_{k-1})(x_k - x_{k+1}) \cdots (x_k - x_n)}$$

Plugging in the grid points verifies the following:

Proposition (delta interpolation)

$$\ell_k(x_j) = \delta_{kj}$$

We can use these to construct the interpolatory polynomial:

Theorem (Lagrange interpolation) The unique polynomial of degree at most $n-1$ that interpolates f at n distinct points x_j is:

$$p(x) = f(x_1)\ell_1(x) + \cdots + f(x_n)\ell_n(x)$$

Proof Note that

$$p(x_j) = \sum_{j=1}^{n} f(x_j)\ell_k(x_j) = f(x_j)$$

so we just need to show it is unique. Suppose $\tilde{p(x)}$ is a polynomial of degree at most $n-1$ that also interpolates f. Then $\tilde{p-p}$ vanishes at n distinct points. Thus by the fundamental theorem of algebra it must be zero.

■

Example We can interpolate $\exp(x)$ at the points $0, 1, 2$:

$$\begin{aligned} p(x) &= \ell_1(x) + \mathrm{e}\ell_2(x) + \mathrm{e}^2\ell_3(x) = \frac{(x-1)(x-2)}{(-1)(-2)} + \mathrm{e}\frac{x(x-2)}{(-1)} + \mathrm{e}^2\frac{x(x-1)}{2} \\ &= (1/2 - \mathrm{e} + \mathrm{e}^2/2)x^2 + (-3/2 + 2\mathrm{e} - \mathrm{e}^2/2)x + 1 \end{aligned}$$

Remark Interpolating at evenly spaced points is a really **bad** idea: interpolation is inheritely ill-conditioned. The problem sheet asks you to explore this experimentally.

2.2 2. Roots of orthogonal polynomials and truncated Jacobi matrices

We now consider roots (zeros) of orthogonal polynomials $p_n(x)$. This is important as we shall see they are useful for interpolation and quadrature. For interpolation to be well-defined we first need to guarantee that the roots are distinct.

Lemma An orthogonal polynomial $p_n(x)$ has exactly n distinct roots.

Proof

Suppose $x_1, \ldots, x_j$ are the roots where $p_n(x)$ changes sign, that is,

$$p_n(x) = c_k(x - x_k)^{2p+1} + O((x - x_k)^{2p+2})$$

for $c_k \neq 0$ and $k = 1, \ldots, j$ and $p \in \mathbb{N}$, as $x \to x_k$. Then

$$p_n(x)(x - x_1) \cdots (x - x_j)$$

does not change signs: it behaves like $c_k(x - x_k)^{2p+2} + O(x - x_k)^{2p+3}$ as $x \to x_k$. In other words:

$$\langle p_n, (x - x_1) \cdots (x - x_j) \rangle = \int_a^b p_n(x)(x - x_1) \cdots (x - x_j) w(x) \mathrm{d}x \neq 0.$$

where $w(x)$ is the weight of orthogonality. This is only possible if $j = n$ as $p_n(x)$ is orthogonal w.r.t. all lower degree polynomials.

■

Definition (truncated Jacobi matrix) Given a symmetric Jacobi matrix X, (or the weight $w(x)$ whose orthonormal polynomials are associated with X) the *truncated Jacobi matrix* is

$$X_n := \begin{bmatrix} a_0 & b_0 & & \\ b_0 & \ddots & \ddots & \\ & \ddots & a_{n-2} & b_{n-2} \\ & & b_{n-2} & a_{n-1} \end{bmatrix} \in \mathbb{R}^{n \times n}$$

Lemma (zeros) The zeros $x_1, \ldots, x_n$ of an orthonormal polynomial $q_n(x)$ are the eigenvalues of the truncated Jacobi matrix X_n. More precisely,

$$X_n Q_n = Q_n \begin{bmatrix} x_1 & & \\ & \ddots & \\ & & x_n \end{bmatrix}$$

for the orthogonal matrix

$$Q_n = \begin{bmatrix} q_0(x_1) & \cdots & q_0(x_n) \\ \vdots & \cdots & \vdots \\ q_{n-1}(x_1) & \cdots & q_{n-1}(x_n) \end{bmatrix} \begin{bmatrix} \alpha_1^{-1} & & \\ & \ddots & \\ & & \alpha_n^{-1} \end{bmatrix}$$

where $\alpha_j = \sqrt{q_0(x_j)^2 + \cdots + q_{n-1}(x_j)^2}$.

Proof

We construct the eigenvector (noting $b_{n-1} q_n(x_j) = 0$):

$$X_n \begin{bmatrix} q_0(x_j) \\ \vdots \\ q_{n-1}(x_j) \end{bmatrix} = \begin{bmatrix} a_0 q_0(x_j) + b_0 q_1(x_j) \\ b_0 q_0(x_j) + a_1 q_1(x_j) + b_1 q_2(x_j) \\ \vdots \\ b_{n-3} q_{n-3}(x_j) + a_{n-2} q_{n-2}(x_j) + b_{n-2} q_{n-1}(x_j) \\ b_{n-2} q_{n-2}(x_j) + a_{n-1} q_{n-1}(x_j) + b_{n-1} q_n(x_j) \end{bmatrix} = x_j \begin{bmatrix} q_0(x_j) \\ q_1(x_j) \\ \vdots \\ q_{n-1}(x_j) \end{bmatrix}$$

The result follows from normalising the eigenvectors. Since X_n is symmetric the eigenvector matrix is orthogonal. ■

Example (Chebyshev roots) Consider $T_n(x) = \cos n \operatorname{acos} x$. The roots are $x_j = \cos\theta_j$ where $\theta_j = (j-1/2)\pi/n$ for $j = 1, \ldots, n$ are the roots of $\cos n\theta$ that are inside $[0, \pi]$.

Consider the $n = 3$ case where we have

$$x_1, x_2, x_3 = \cos(\pi/6), \cos(\pi/2), \cos(5\pi/6) = \sqrt{3}/2, 0, -\sqrt{3/2}$$

We also have from the 3-term recurrence:

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_2(x) = 2xT_1(x) - T_0(x) &= 2x^2 - 1 \\ T_3(x) = 2xT_2(x) - T_1(x) &= 4x^2 - 3x \end{aligned}$$

We orthonormalise by rescaling

$$\begin{aligned} q_0(x) &= 1/\sqrt{\pi} \\ q_k(x) &= T_k(x)\sqrt{2}/\sqrt{\pi} \end{aligned}$$

so that the Jacobi matrix is symmetric:

$$x[q_0(x)|q_1(x)|\cdots] = [q_0(x)|q_1(x)|\cdots] \underbrace{\begin{bmatrix} 0 & 1/\sqrt{2} & & & \\ 1/\sqrt{2} & 0 & 1/2 & & \\ & 1/2 & 0 & 1/2 & \\ & & 1/2 & 0 & \ddots \\ & & & \ddots & \ddots \end{bmatrix}}_{X}$$

We can then confirm that we have constructed an eigenvector/eigenvalue of the 3×3 truncation of the Jacobi matrix, e.g. at $x_2 = 0$:

$$\begin{bmatrix} 0 & 1/\sqrt{2} & \\ 1/\sqrt{2} & 0 & 1/2 \\ & 1/2 & 0 \end{bmatrix} \begin{bmatrix} q_0(0) \\ q_1(0) \\ q_2(0) \end{bmatrix} = \frac{1}{\sqrt{\pi}} \begin{bmatrix} 0 & 1/\sqrt{2} & \\ 1/\sqrt{2} & 0 & 1/2 \\ & 1/2 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ -\frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

2.3 3. Interpolatory quadrature rules

Definition (interpolatory quadrature rule) Given a set of points $\mathbf{x} = [x_1, \ldots, x_n]$ the interpolatory quadrature rule is:

$$\Sigma_n^{w,\mathbf{x}}[f] := \sum_{j=1}^{n} w_j f(x_j)$$

where

$$w_j := \int_a^b \ell_j(x) w(x) \mathrm{d}x$$

Proposition (interpolatory quadrature is exact for polynomials) Interpolatory quadrature is exact for all degree $n-1$ polynomials p:

$$\int_a^b p(x)w(x)\mathrm{d}x = \Sigma_n^{w,\mathbf{x}}[f]$$

Proof The result follows since, by uniqueness of interpolatory polynomial:

$$p(x) = \sum_{j=1}^{n} p(x_j)\ell_j(x)$$

■

Example (arbitrary points) Find the interpolatory quadrature rule for $w(x) = 1$ on $[0, 1]$ with points $[x_1, x_2, x_3] = [0, 1/4, 1]$? We have:

$$\begin{aligned} w_1 &= \int_0^1 w(x)\ell_1(x)\mathrm{d}x = \int_0^1 \frac{(x-1/4)(x-1)}{(-1/4)(-1)}\mathrm{d}x = -1/6 \\ w_2 &= \int_0^1 w(x)\ell_2(x)\mathrm{d}x = \int_0^1 \frac{x(x-1)}{(1/4)(-3/4)}\mathrm{d}x = 8/9 \\ w_3 &= \int_0^1 w(x)\ell_3(x)\mathrm{d}x = \int_0^1 \frac{x(x-1/4)}{3/4}\mathrm{d}x = 5/18 \end{aligned}$$

That is we have

$$\Sigma_n^{w,\mathbf{x}}[f] = -\frac{f(0)}{6} + \frac{8f(1/4)}{9} + \frac{5f(1)}{18}$$

This is indeed exact for polynomials up to degree 2 (and no more):

$$\Sigma_n^{w,\mathbf{x}}[1] = 1, \Sigma_n^{w,\mathbf{x}}[x] = 1/2, \Sigma_n^{w,\mathbf{x}}[x^2] = 1/3, \Sigma_n^{w,\mathbf{x}}[x^3] = 7/24 \neq 1/4.$$

Example (Chebyshev roots) Find the interpolatory quadrature rule for $w(x) = 1/\sqrt{1-x^2}$ on $[-1, 1]$ with points equal to the roots of $T_3(x)$. This is a special case of Gaussian quadrature which we will approach in another way below. We use:

$$\int_{-1}^1 w(x)\mathrm{d}x = \pi, \int_{-1}^1 xw(x)\mathrm{d}x = 0, \int_{-1}^1 x^2 w(x)\mathrm{d}x = \pi/2$$

Recall from before that $x_1, x_2, x_3 = \sqrt{3}/2, 0, -\sqrt{3}/2$. Thus we have:

$$\begin{aligned} w_1 &= \int_{-1}^1 w(x)\ell_1(x)\mathrm{d}x = \int_{-1}^1 \frac{x(x+\sqrt{3}/2)}{(\sqrt{3}/2)\sqrt{3}\sqrt{1-x^2}}\mathrm{d}x = \frac{\pi}{3} \\ w_2 &= \int_{-1}^1 w(x)\ell_2(x)\mathrm{d}x = \int_{-1}^1 \frac{(x-\sqrt{3}/2)(x+\sqrt{3}/2)}{(-3/4)\sqrt{1-x^2}}\mathrm{d}x = \frac{\pi}{3} \\ w_3 &= \int_{-1}^1 w(x)\ell_3(x)\mathrm{d}x = \int_{-1}^1 \frac{(x-\sqrt{3}/2)x}{(-\sqrt{3})(-\sqrt{3}/2)\sqrt{1-x^2}}\mathrm{d}x = \frac{\pi}{3} \end{aligned}$$

(It's not a coincidence that they are all the same but this will differ for roots of other OPs.) That is we have

$$\Sigma_n^{w,\mathbf{x}}[f] = \frac{\pi}{3}(f(\sqrt{3}/2) + f(0) + f(-\sqrt{3}/2)$$

This is indeed exact for polynomials up to degree $n-1=2$, but it goes all the way up to $2n-1=5$:

$$\Sigma_n^{w,\mathbf{x}}[1] = \pi, \Sigma_n^{w,\mathbf{x}}[x] = 0, \Sigma_n^{w,\mathbf{x}}[x^2] = \frac{\pi}{2},$$
$$\Sigma_n^{w,\mathbf{x}}[x^3] = 0, \Sigma_n^{w,\mathbf{x}}[x^4] = \frac{3\pi}{8}, \Sigma_n^{w,\mathbf{x}}[x^5] = 0$$
$$\Sigma_n^{w,\mathbf{x}}[x^6] = \frac{9\pi}{32} \neq \frac{5\pi}{16}$$

We shall explain this miracle next.

2.4 4. Gaussian quadrature

Gaussian quadrature is the interpolatory quadrature rule corresponding to the grid x_j defined as the roots of the orthonormal polynomial $q_n(x)$. We shall see that it is exact for polynomials up to degree $2n-1$, i.e., double the degree of other interpolatory quadrature rules from other grids.

Definition (Gauss quadrature) Given a weight $w(x)$, the Gauss quadrature rule is:

$$\int_a^b f(x)w(x)\mathrm{d}x \approx \underbrace{\sum_{j=1}^n w_j f(x_j)}_{\Sigma_n^w[f]}$$

where $x_1, \ldots, x_n$ are the roots of the orthonormal polynomials $q_n(x)$ and

$$w_j := \frac{1}{\alpha_j^2} = \frac{1}{q_0(x_j)^2 + \cdots + q_{n-1}(x_j)^2}.$$

Equivalentally, $x_1, \ldots, x_n$ are the eigenvalues of X_n and

$$w_j = \int_a^b w(x)\mathrm{d}x Q_n[1,j]^2.$$

(Note we have $\int_a^b w(x)\mathrm{d}x q_0(x)^2 = 1$.)

In analogy to how Fourier series are orthogonal with respect to Trapezium rule, Orthogonal polynomials are orthogonal with respect to Gaussian quadrature:

Lemma (Discrete orthogonality) For $0 \leq \ell, m \leq n-1$, the orthonormal polynomials $q_n(x)$ satisfy

$$\Sigma_n^w[q_\ell q_m] = \delta_{\ell m}$$

Proof

$$\Sigma_n^w[q_\ell q_m] = \sum_{j=1}^{n} \frac{q_\ell(x_j)q_m(x_j)}{\alpha_j^2} = [q_\ell(x_1)/\alpha_1|\cdots|q_\ell(x_n)/\alpha_n]\begin{bmatrix} q_m(x_1)/\alpha_1 \\ \vdots \\ q_m(x_n)/\alpha_n \end{bmatrix} = \mathbf{e}_\ell Q_n Q_n^\top \mathbf{e}_m = \delta_{\ell m}$$

■

Just as approximating Fourier coefficients using Trapezium rule gives a way of interpolating at the grid, so does Gaussian quadrature:

Theorem (interpolation via quadrature) For the orthonormal polynomials $q_n(x)$,

$$f_n(x) = \sum_{k=0}^{n-1} c_k^n q_k(x) \text{ for } c_k^n := \Sigma_n^w[f q_k]$$

interpolates $f(x)$ at the Gaussian quadrature points $x_1, \ldots, x_n$.

Proof

Consider the Vandermonde-like matrix:

$$\tilde{V} := \begin{bmatrix} q_0(x_1) & \cdots & q_{n-1}(x_1) \\ \vdots & \ddots & \vdots \\ q_0(x_n) & \cdots & q_{n-1}(x_n) \end{bmatrix}$$

and define

$$Q_n^w := \tilde{V}^\top \begin{bmatrix} w_1 & & \\ & \ddots & \\ & & w_n \end{bmatrix} = \begin{bmatrix} q_0(x_1)w_1 & \cdots & q_0(x_n)w_n \\ \vdots & \ddots & \vdots \\ w_1 q_{n-1}(x_1) & \cdots & q_{n-1}(x_n)w_n \end{bmatrix}$$

so that

$$\begin{bmatrix} c_0^n \\ \vdots \\ c_{n-1}^n \end{bmatrix} = Q_n^w \begin{bmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix}.$$

Note that if $p(x) = [q_0(x)|\cdots|q_{n-1}(x)]\mathbf{c}$ then

$$\begin{bmatrix} p(x_1) \\ \vdots \\ p(x_n) \end{bmatrix} = \tilde{V}\mathbf{c}$$

But we see that (similar to the Fourier case)

$$Q_n^w \tilde{V} = \begin{bmatrix} \Sigma_n^w[q_0 q_0] & \cdots & \Sigma_n^w[q_0 q_{n-1}] \\ \vdots & \ddots & \vdots \\ \Sigma_n^w[q_{n-1} q_0] & \cdots & \Sigma_n^w[q_{n-1} q_{n-1}] \end{bmatrix} = I_n$$

■

Corollary Gaussian quadrature is an interpolatory quadrature rule with the interpolation points equal to the roots of q_n:

$$\Sigma_n^w[f] = \Sigma_n^{w,\mathbf{x}}[f]$$

Proof We want to show its the same as integrating the interpolatory polynomial:

$$\int_a^b f_n(x)w(x)\mathrm{d}x = \frac{1}{q_0(x)}\sum_{k=0}^{n-1} c_k^n \int_a^b q_k(x)q_0(x)w(x)\mathrm{d}x = \frac{c_0^n}{q_0} = \Sigma_n^w[f].$$

■

A consequence of being an interpolatory quadrature rule is that it is exact for all polynomials of degree $n-1$. The *miracle* of Gaussian quadrature is it is exact for twice as many!

Theorem (Exactness of Gauss quadrature) If $p(x)$ is a degree $2n-1$ polynomial then Gauss quadrature is exact:

$$\int_a^b p(x)w(x)\mathrm{d}x = \Sigma_n^w[p].$$

Proof Using polynomial division algorithm (e.g. by matching terms) we can write

$$p(x) = q_n(x)s(x) + r(x)$$

where s and r are degree $n-1$ and $q_n(x)$ is the degree n orthonormal polynomial. Then we have:

$$\Sigma_n^w[p] = \underbrace{\Sigma_n^w[q_n s]}_{0 \text{ since evaluating } q_n \text{ at zeros}} + \Sigma_n^w[r] = \int_a^b r(x)w(x)\mathrm{d}x = \underbrace{\int_a^b q_n(x)s(x)w(x)\mathrm{d}x}_{0 \text{ since } s \text{ is degree} < n} + \int_a^b r(x)w(x)\mathrm{d}x$$

$$= \int_a^b p(x)w(x)\mathrm{d}x.$$

■

Example (Chebyshev expansions) Consider the construction of Gaussian quadrature for $n = 3$. To determine the weights we need:

$$w_j^{-1} = \alpha_j^2 = q_0(x_j)^2 + q_1(x_j)^2 + q_2(x_j)^2 = \frac{1}{\pi} + \frac{2}{\pi}x_j^2 + \frac{2}{\pi}(2x_j^2 - 1)^2$$

We can check each case and deduce that $w_j = \pi/3$. Thus we recover the interpolatory quadrature rule. Further, we can construct the transform

$$\begin{aligned} Q_3^w &= \begin{bmatrix} w_1 q_0(x_1) & w_2 q_0(x_2) & w_3 q_0(x_3) \\ w_1 q_1(x_1) & w_2 q_1(x_2) & w_3 q_1(x_3) \\ w_1 q_3(x_1) & w_2 q_3(x_2) & w_3 q_3(x_3) \end{bmatrix} \\ &= \frac{\pi}{3}\begin{bmatrix} 1/\sqrt{\pi} & 1/\sqrt{\pi} & 1/\sqrt{\pi} \\ x_1\sqrt{2/\pi} & x_2\sqrt{2/\pi} & x_3\sqrt{2/\pi} \\ (2x_1^2-1)\sqrt{2/\pi} & (2x_2^2-1)\sqrt{2/\pi} & (2x_3^2-1)\sqrt{2/\pi} \end{bmatrix} \\ &= \frac{\sqrt{\pi}}{3}\begin{bmatrix} 1 & 1 & 1 \\ \sqrt{6}/2 & 0 & -\sqrt{6}/2 \\ 1/\sqrt{2} & -\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \end{aligned}$$

We can use this to expand a polynomial, e.g. x^2:

$$Q_3^2 \begin{bmatrix} x_1^2 \\ x_2^2 \\ x_3^2 \end{bmatrix} = \frac{\sqrt{\pi}}{3} \begin{bmatrix} 1 & 1 & 1 \\ \sqrt{6}/2 & 0 & -\sqrt{6}/2 \\ 1/\sqrt{2} & -\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} 3/4 \\ 0 \\ 3/4 \end{bmatrix} = \begin{bmatrix} \sqrt{\pi}/2 \\ 0 \\ \sqrt{\pi}/(2\sqrt{2}) \end{bmatrix}$$

In other words:

$$x^2 = \frac{\sqrt{\pi}}{2} q_0(x) + \frac{\sqrt{\pi}}{2\sqrt{2}} q_2(x) = \frac{1}{2} T_0(x) + \frac{1}{2} T_2(x)$$

which can be easily confirmed.

2.5 Approximation with Chebyshev polynomials

Previously, we used the formula, derived via trigonometric manipulations,

$$T_1(x) = xT_0(x), \qquad T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$

Rearranging, this becomes

$$xT_0(x) = T_1(x), \qquad xT_n(x) = \frac{T_{n-1}(x)}{2} + \frac{T_{n+1}(x)}{2}$$

This tells us that we have the three-term recurrence with $a_n = 0$, $b_0 = 1$, $c_n = b_n = \frac{1}{2}$ for $n > 0$.

This can be extended to function approximation. Provided the sum converges absolutely and uniformly in x, we can write

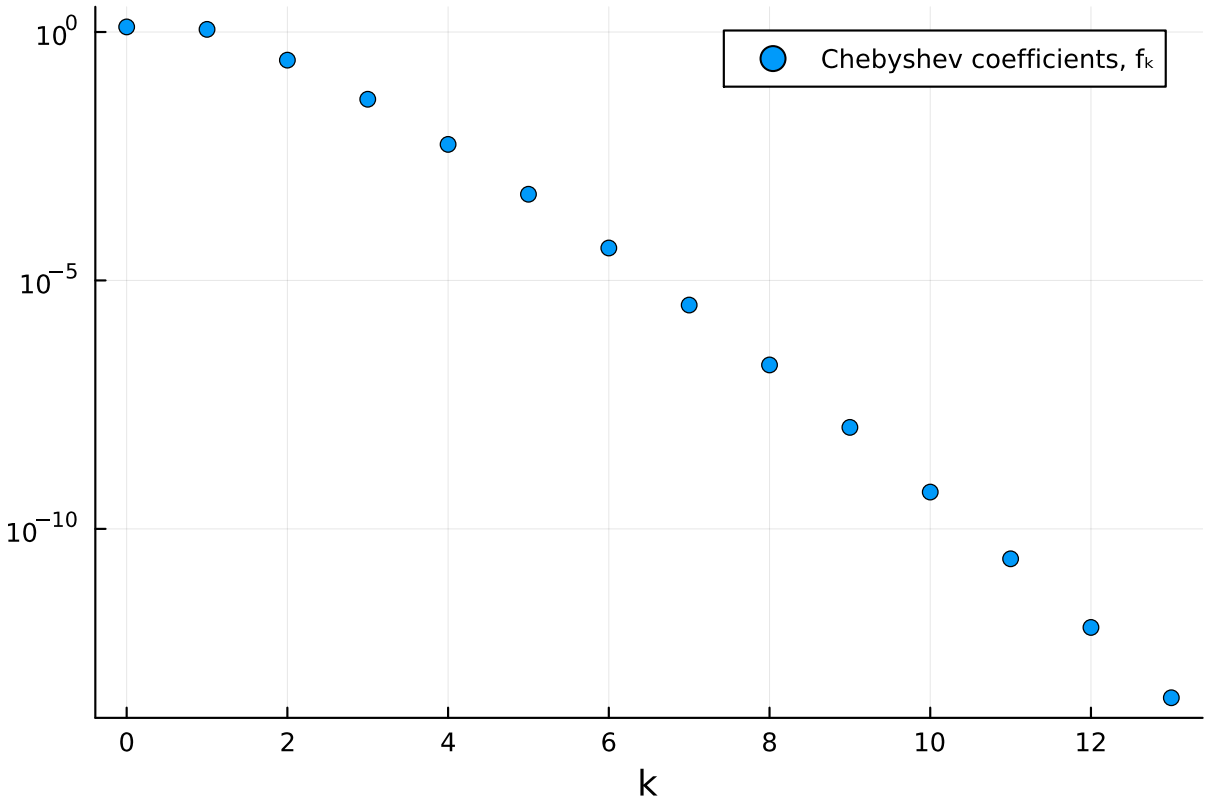
$$f(x) = \sum_{k=0}^{\infty} f_k T_k(x).$$

In practice, we can approximate smooth functions by a finite truncation:

$$f(x) \approx \sum_{k=0}^{n-1} f_k T_k(x)$$

Here we see that e^x can be approximated by a Chebyshev approximation using 14 coefficients and is accurate to 16 digits:

```
f = Fun(x -> exp(x), Chebyshev())
scatter(0:ncoefficients(f)-1,abs.(f.coefficients);yscale=:log10,label="Chebyshev
coefficients, f_k",xlabel="k")
```




```
@show ncoefficients(f)
@show f(0.1) # equivalent to f.coefficients'*[cos(k*acos(x)) for k=0:ncoefficients(f)-1]
@show exp(0.1);
```

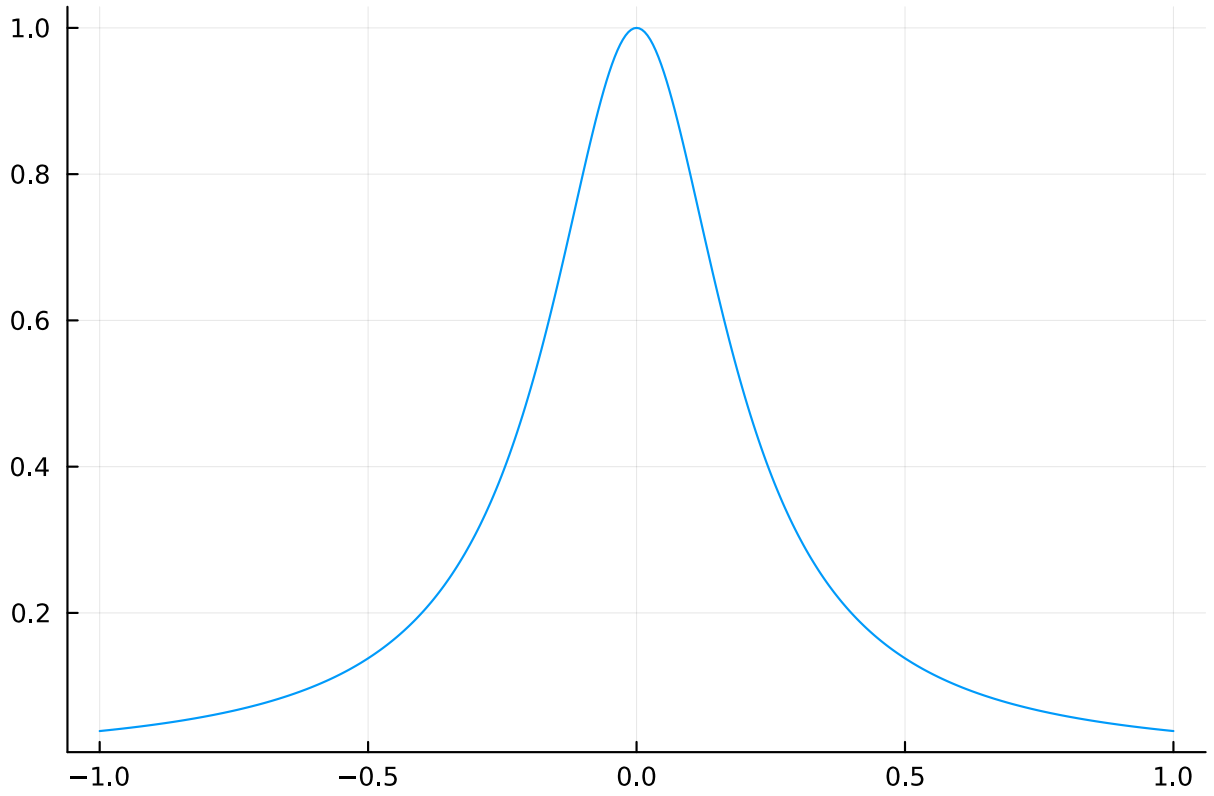
```
ncoefficients(f) = 14
f(0.1) = 1.1051709180756477
exp(0.1) = 1.1051709180756477
```

The accuracy of this approximation is typically dictated by the smoothness of f: the more times we can differentiate, the faster it converges. For analytic functions, it's dictated by the domain of analyticity, just like Laurent/Fourier series. In the case above, e^x is entire hence we get faster than exponential convergence.

Chebyshev expansions work even when Taylor series do not. For example, the following function has poles at $\pm\frac{i}{5}$, which means the radius of convergence for the Taylor series is $|x| < \frac{1}{5}$, but Chebyshev polynomials continue to work on $[-1, 1]$:

```
f = Fun( x -> 1/(25x^2 + 1), Chebyshev())
@show ncoefficients(f)
plot(f;label=false)
```

```
ncoefficients(f) = 189
```



This can be explained for Chebyshev expansion by noting that it is the cosine expansion / Fourier expansion of an even function:

$$f(x) = \sum_{k=0}^{\infty} f_k T_k(x) \Leftrightarrow f(\cos\theta) = \sum_{k=0}^{\infty} f_k \cos k\theta$$

2.5.1 Exponential decay of Fourier coefficients of periodic, analytic functions revisited

Before we get to the decay of Chebyshev coefficients, we revisit the proof of the exponential decay of *Fourier* coefficients in Lecture 6. Suppose $f(\theta)$ is 2π-periodic and analytic on $\theta \in [-\pi, \pi)$, then

$$f(\theta) = \sum_{k=-\infty}^{\infty} \hat{f}_k \mathrm{e}^{\mathrm{i}k\theta}$$

where

$$\hat{f}_k = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(\theta) \mathrm{e}^{-\mathrm{i}k\theta} \mathrm{d}\theta.$$

Recall in Lecture 6 we set $z = \mathrm{e}^{\mathrm{i}\theta}$ in which case the Fourier series of f becomes a Laurent series of a function $g(z)$:

$$f(\theta) = \sum_{k=-\infty}^{\infty} \hat{f}_k \mathrm{e}^{\mathrm{i}k\theta} = \sum_{k=-\infty}^{\infty} g_k z^k =: g(z),$$

with $g_k = \hat{f}_k$. We proved that if $g(z)$ is analytic on the closed annulus $A_{r,R} = \{z : r \le |z| \le R\}$, $0 < r < 1$, $R > 1$ then for all $k \in \mathbb{Z}$, $|g_k| \le M \min\left\{\frac{1}{R^k}, \frac{1}{r^k}\right\}$ where $M = \sup_{z\in A_{r,R}} |g(z)|$. This result implies the exponential decay of the Fourier coefficients of f.

An annulus in the z-plane corresponds to a strip of width 2π in the (complex) θ-plane under the transformation $z = \mathrm{e}^{\mathrm{i}\theta}$, $\Re\theta \in [-\pi, \pi)$:

$$\begin{aligned} &z \in A_{r,R} = \{z : r \le |z| \le R\} \qquad \underbrace{\Longleftrightarrow}_{z=\mathrm{e}^{\mathrm{i}\theta}} \\ &\theta \in S_{r,R} = \{\theta : -\pi \le \Re\theta < \pi, -\log(R) \le \Im\theta \le \log(1/r)\}. \end{aligned}$$

Suppose $f(\theta)$ is real-valued on $[-\pi, \pi)$, then $\overline{f(\theta)} = f(\overline{\theta})$. Hence if the closest singularity to the real θ-axis is at $\theta = \theta_x + \mathrm{i}\theta_y$, with $\theta_x \in [-\pi, \pi)$ and $\theta_y > 0$, then f also has a singularity at $\theta_x - \mathrm{i}\theta_y$. Thus f is analytic in the strip

$$S_{r,R} = \theta \in \{\theta : -\pi \le \Re\theta < \pi, -\log(R) \le \Im\theta \le \log(1/r)\}$$

with

$$\frac{1}{r} = R < \mathrm{e}^{\theta_y}$$

and the Fourier coefficients are bounded by

$$|f_k| = |g_k| \le M \min\left\{\frac{1}{R^k}, \frac{1}{r^k}\right\} = Mr^{|k|} = MR^{-|k|}, \qquad k \in \mathbb{Z},$$

where $M = \sup_{z\in A_{r,R}} |g(z)| = \sup_{\theta\in S_{r,R}} |f(\theta)|$. The larger the strip of analyticity, the larger we can make R and the faster the Fourier coefficients of f decay as $|k| \to \infty$ (hence the faster the truncated Fourier expansion $\sum_{k=-n}^{n} \hat{f}_k \mathrm{e}^{\mathrm{i}k\theta}$ of f converges to f as $n \to \infty$).

Example (see also Lecture 6) The function

$$f(\theta) = \frac{1}{2 - \cos\theta},$$

has poles at $\theta = \pm\mathrm{i}\log(2 + \sqrt{3})$; it is analytic in the strip $S_{r,R}$ with $R = 1/r < 2 + \sqrt{3}$ and the maximum of $|f(\theta)|$ on $S_{r,R}$ is

$$M = \frac{2}{4 - R^{-1} + R},$$

hence

$$|f_k| = |g_k| \le \frac{2}{4 - R - R^{-1}} R^{-|k|}, \qquad k \in \mathbb{Z},$$

for all $R < 2 + \sqrt{3}$.

```
# This works in Julia 1.7.2, just use the FFT
#g =Fun(θ -> 1/(2-cos(θ)), Laurent(-π .. π))
#g_+ = g.coefficients[1:2:end]
#scatter(abs.(g_+); yscale=:log10, label="|g_k|", legend=:bottomleft,xlabel="k")
#R = 1.1
```

```
#scatter!(2/(4-R-inv(R))*R.^(-(0:length(g_+))), label = "R = $R")
#R = 3.5
#scatter!(2/(4-R-inv(R))*R.^(-(0:length(g_+))), label = "R = $R")
#R = 2+sqrt(3)-0.1
#scatter!(2/(4-R-inv(R))*R.^(-(0:length(g_+))), label = "R = $R")
```

2.5.2 Exponential decay of Chebyshev coefficients of analytic functions

Suppose $f(x)$ is analytic on $[-1, 1]$, then

$$\begin{aligned} f(x) &= \sum_{k=0}^{\infty} f_k T_k(x) \\ &= \sum_{k=0}^{\infty} f_k \cos k\theta \qquad (x = \cos\theta) \\ &= \sum_{k=0}^{\infty} \frac{f_k}{2}\left(z^k + z^{-k}\right) \qquad (z = \mathrm{e}^{\mathrm{i}\theta}) \\ &=: \sum_{k=-\infty}^{\infty} g_k z^k =: g(z) \qquad (g_0 = f_0, g_k = g_{-k} = f_k/2, k \geq 0) \end{aligned}$$

Now we can use the bound on the Laurent coefficients of $g(z)$ to bound the Chebyshev coefficients of $f(x)$. First we need to establish what is the image in the (complex) x-plane of an annulus in the z-plane under the transformation $2x = z + z^{-1}$, which is known as the Joukowsky map.

Let $\rho > 1$ and

$$A_{1,\rho} = \{z : 1 \leq |z| \leq \rho\}, \qquad A_{\rho^{-1},1} = \{z : \rho^{-1} \leq |z| \leq 1\}.$$

The Joukowsky transformation maps $A_{1,\rho}$ and $A_{\rho^{-1},1}$ to the following ellipse (known as a Bernstein ellipse) in the x-plane:

$$E_\rho = \left\{x : \frac{(\Re x)^2}{\alpha^2} + \frac{(\mathrm{i}mx)^2}{\beta^2} \leq 1, \alpha = \frac{1}{2}\left(\rho + \rho^{-1}\right), \beta = \frac{1}{2}\left(\rho - \rho^{-1}\right)\right\}.$$

We conclude that if $f(x)$ is analytic on E_ρ (or $g(z)$ is analytic on $A_{\rho^{-1},\rho}$) and $M = \sup_{x\in E_\rho} |f(x)| = \sup_{z\in A_{\rho^{-1},\rho}} |g(z)|$, then

$$|f_k| = 2|g_k| \leq 2M\rho^{-k}, \qquad k \geq 1.$$

The larger the Bernstein ellipse on which $f(x)$ is analytic, the faster the decay of the Chebyshev coefficients as $k \to \infty$ (and hence the faster the convergence of the Chebyshev expansion of f).

A truncated Chebyshev expansion with n terms of a function $f(x)$ that is analytic on E_ρ converges at essentially the same exponential rate as the bound on the Chebyshev coefficients as $n \to \infty$:

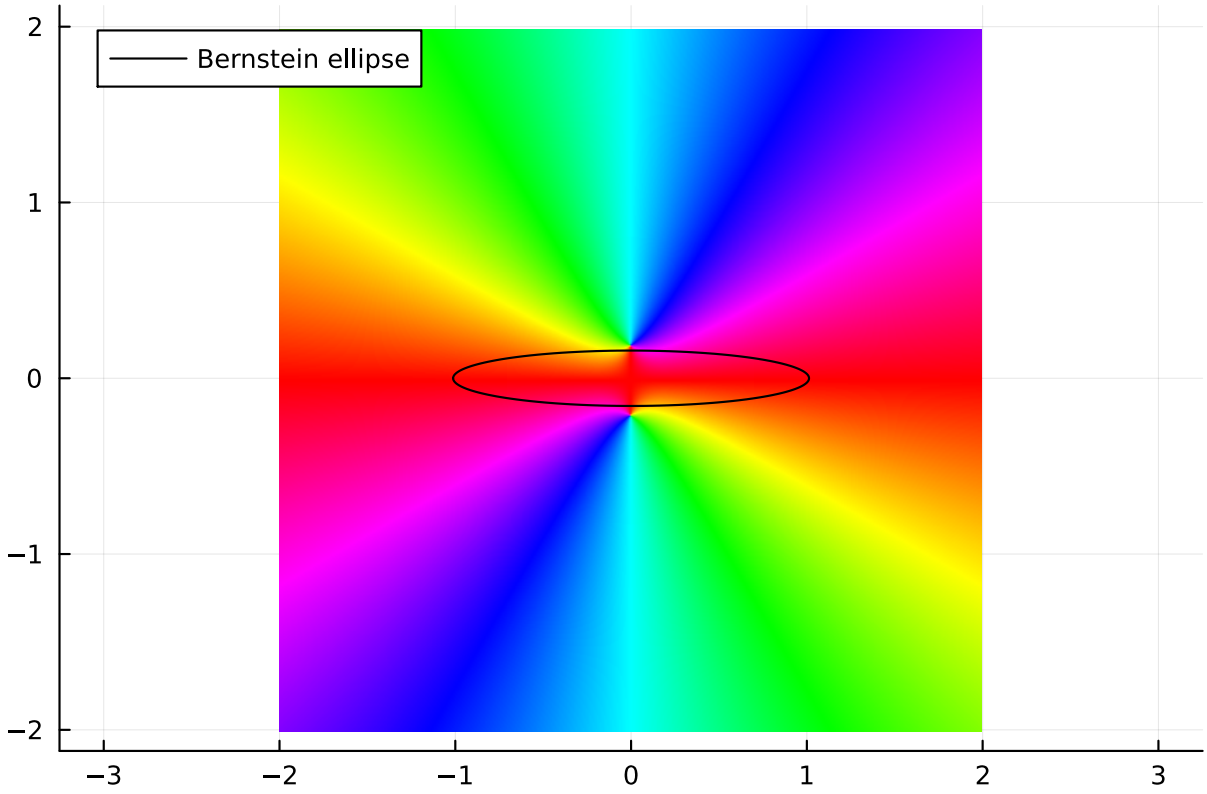
$$\left|f(x) - \sum_{k=0}^{n-1} f_k T_k(x)\right| = \left|\sum_{k=n}^{\infty} f_k T_k(x)\right| \le \sum_{k=n}^{\infty} |f_k| \le 2M \sum_{k=n}^{\infty} \rho^{-k}$$
$$= 2M \frac{\rho^{-n}}{1-\rho^{-1}}$$

Example In the case of $f(x) = \frac{1}{25x^2+1}$, setting $2x = z + z^{-1}$, we find that

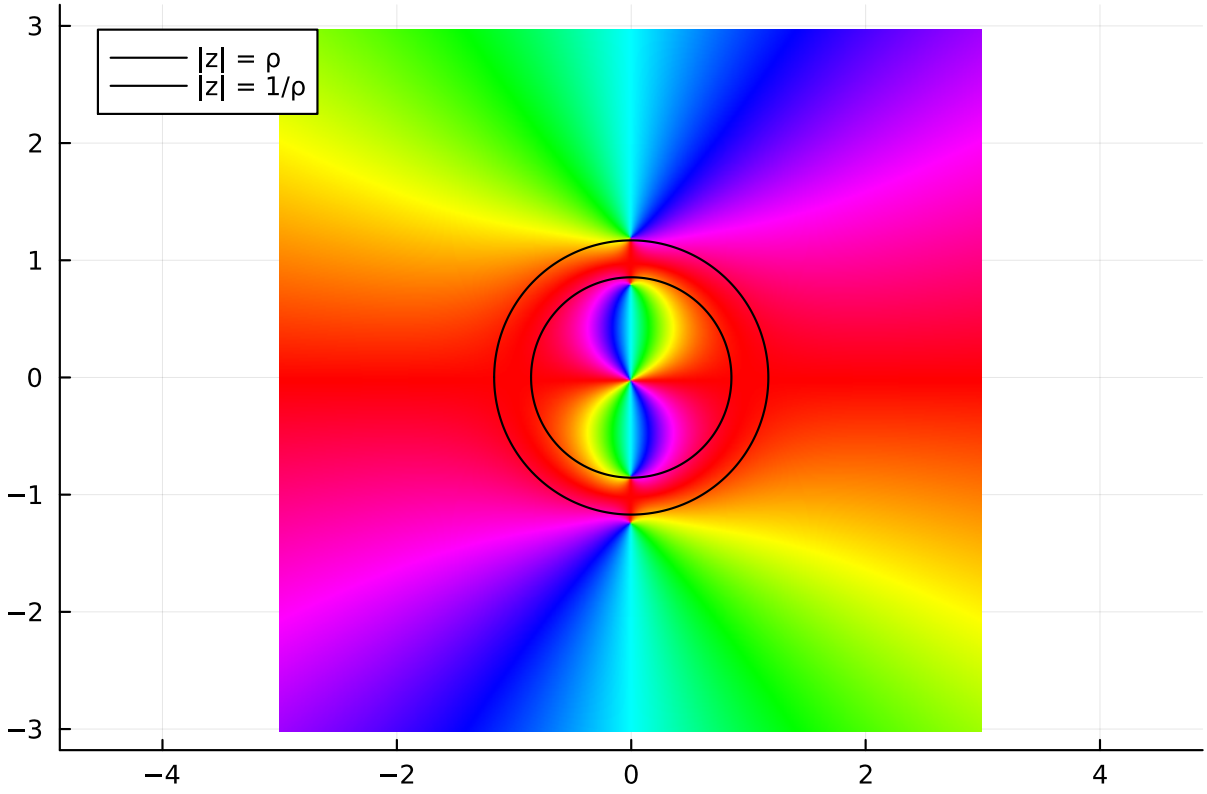
$$f(x) = f\left(\frac{z+z^{-1}}{2}\right) = g(z) = \frac{4z^2}{25 + 54z^2 + 25z^4}.$$

In the complex x-plane, $f(x)$ has poles at $\pm i/5$ and is analytic on E_ρ with $\beta = (\rho - \rho^{-1})/2 < 1/5$, hence $\rho < \frac{1+\sqrt{26}}{5}$. In the z-plane, $g(z)$ has poles at $\pm i\frac{1\pm\sqrt{26}}{5} \approx \pm 0.8198040i, \pm 1.2198i$ and is analytic on the annulus $\rho^{-1} \le |z| \le \rho$.

```
ρ = (1 + sqrt(26))/5-0.05;
α = (ρ + 1/ρ)/2
β = (ρ - 1/ρ)/2
θ = -π:0.01:π
f = x -> 1/(25x^2 + 1)
phaseplot(-2..2, -2..2, z -> f(z))
plot!(α*cos.(θ),β*sin.(θ);linecolor="black",label="Bernstein ellipse")
```




```
phaseplot(-3..3, -3..3, z -> f((z+1/z)/2))
plot!(ρ*cos.(θ),ρ*sin.(θ),linecolor="black",label="|z| = ρ")
plot!(cos.(θ)/ρ,sin.(θ)/ρ,linecolor="black",label="|z| = 1/ρ")
```



For $\beta = (\rho - \rho^{-1})/2 < 1/5$, we have

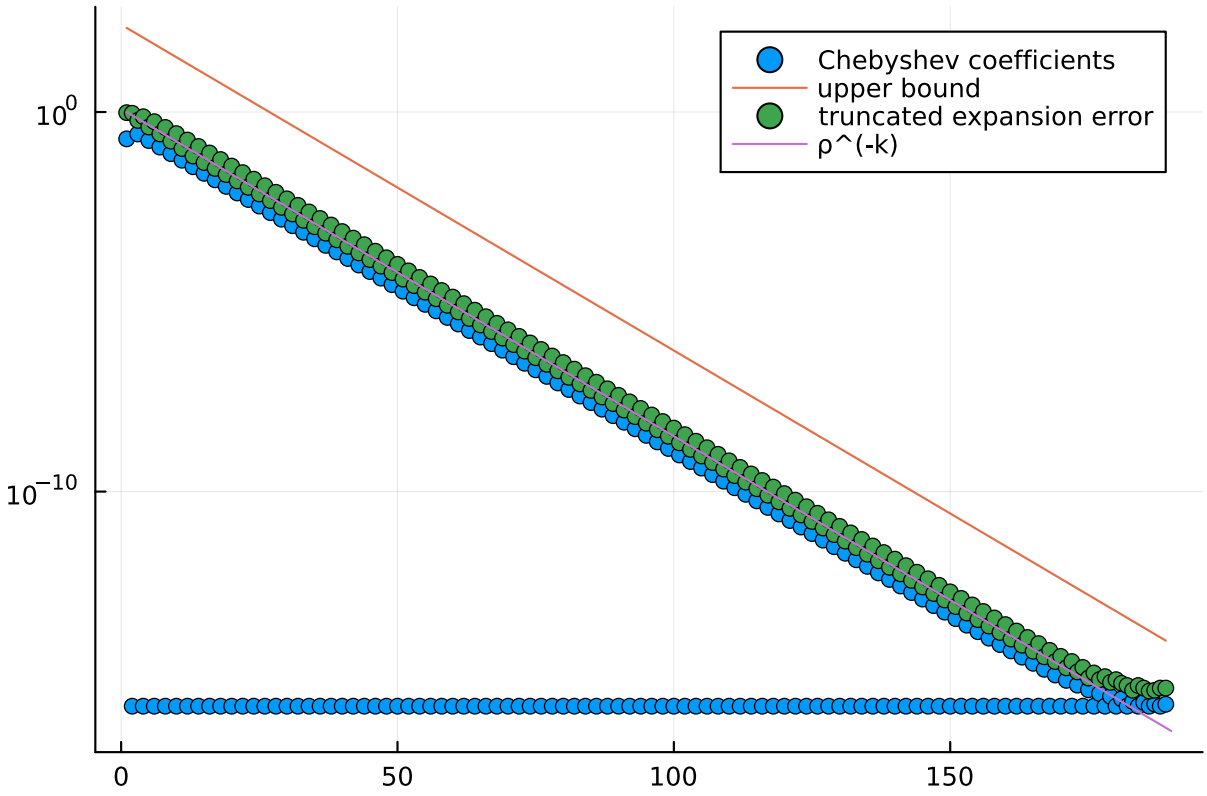
$$M = \sup_{x \in E_\rho} |f(x)| = \frac{1}{1 - 25\beta^2}$$

hence for $k \geq 1$,

$$|f_k| \leq \frac{2}{1 - 25\beta^2} \rho^{-k}, \qquad 1 < \rho < \frac{1 + \sqrt{26}}{5}.$$

Therefore we predict a rate of decay of about 1.2198^{-k}:

```
bound(β,k) = 2/(1-25*β^2)*(β + sqrt(β^2 + 1))^(-k)
f = Fun( x -> 1/(25x^2 + 1), Chebyshev())
scatter(abs.(f.coefficients) .+ eps(); yscale=:log10, label="Chebyshev coefficients")
plot!(1:ncoefficients(f), bound.(1/5-0.001,1:ncoefficients(f)); label="upper bound")
# Also calculate the error of truncated Chebyshev expansions with n terms for n = 1, 2,
...
xx = -1:0.001:1 # a fine grid on which to evaluate
Errv = [(maximum(abs.(f.(xx)-Fun(f,Chebyshev(),n).(xx)))) for n = 1:ncoefficients(f)]
scatter!(1:ncoefficients(f),Errv.+eps();label="truncated expansion error")
plot!( 1.2198.^(-(0:ncoefficients(f))); label="ρ^(-k)")
```



Applied Complex Analysis (2021)

3 Lecture 20: Orthogonal polynomials and differential equations

This lecture we do the following:

1. Recurrence relationships for Chebyshev and ultraspherical polynomials
   - Conversion
   - Three-term recurrence and Jacobi operators
2. Application: solving differential equations
   - First order constant coefficients differential equations
   - Second order constant coefficient differential equations with boundary conditions
   - Non-constant coefficients
3. Differential equations satisfied by orthogonal polynomials

That is, we introduce recurrences related to ultraspherical polynomials. This allows us to represent general linear differential equations as almost-banded systems.

3.1 Recurrence relationships for Chebyshev and ultraspherical polynomials

We have discussed general properties, but now we want to discuss some classical orthogonal polynomials, beginning with Chebyshev (first kind) $T_n(x)$, which is orthogonal w.r.t. $\frac{1}{\sqrt{1-x^2}}$ and ultraspherical $C_n^{(\lambda)}(x)$, which is orthogonal w.r.t. $(1-x^2)^{\lambda-\frac{1}{2}}$ for $\lambda\rangle 0$. Note that Chebyshev (second kind) satisfies $U_n(x) = C_n^{(1)}(x)$.

For Chebyshev, recall we have the normalization constant (here we use a superscript $T_n(x) = k_n^{\mathrm{T}} x^n + O(x^{n-1})$)

$$k_0^{\mathrm{T}} = 1, k_n^{\mathrm{T}} = 2^{n-1}$$

For Ultraspherical $C_n^{(\lambda)}$, this is

$$k_n^{(\lambda)} = \frac{2^n(\lambda)_n}{n!} = \frac{2^n\lambda(\lambda+1)(\lambda+2)\cdots(\lambda+n-1)}{n!}$$

where $(\lambda)_n$ is the Pochhammer symbol. Note for $U_n(x) = C_n^{(1)}(x)$ this simplifies to $k_n^{\mathrm{U}} = k_n^{(1)} = 2^n$.

We have already found the recurrence for Chebyshev:

$$xT_n(x) = \frac{T_{n-1}(x)}{2} + \frac{T_{n+1}(x)}{2}$$

We will show that we can use this to find the recurrence for *all* ultraspherical polynomials. But first we need some special recurrences.

Remark Jacobi, Laguerre, and Hermite all have similar relationships, which will be discussed further in the problem sheet.

3.1.1 Derivatives

It turns out that the derivative of $T_n(x)$ is precisely a multiple of $U_{n-1}(x)$, and similarly the derivative of $C_n^{(\lambda)}$ is a multiple of $C_{n-1}^{(\lambda+1)}$.

Proposition (Chebyshev derivative)

$$T_n'(x) = nU_{n-1}(x)$$

Proof We first show that $T_n'(x)$ is orthogonal w.r.t. $\sqrt{1-x^2}$ to all polynomials of degree $m < n-1$, denoted f_m, using integration by parts:

$$\langle T_n', f_m\rangle_{\mathrm{U}} = \int_{-1}^1 T_n'(x)f_m(x)\sqrt{1-x^2}\mathrm{d}x = -\int_{-1}^1 T_n(x)(f_m'(x)(1-x^2) - xf_m)\frac{1}{\sqrt{1-x^2}}\mathrm{d}x = -\langle T_n, f_m'(1-x$$

since $f_m'(1-x^2) - xf_m$ is degree $m-1+2 = m+1 < n$.

The constant works out since

$$T_n'(x) = \frac{\mathrm{d}}{\mathrm{d}x}(2^{n-1}x^n) + O(x^{n-2}) = n2^{n-1}x^{n-1} + O(x^{n-2})$$

■

The exact same proof shows the following:

Proposition (Ultraspherical derivative) $\frac{\mathrm{d}}{\mathrm{d}x}C_n^{(\lambda)}(x) = 2\lambda C_{n-1}^{(\lambda+1)}(x)$

Like the three-term recurrence and Jacobi operators, it is useful to express this in matrix form. That is, for the derivatives of $T_n(x)$ we get

$$\frac{\mathrm{d}}{\mathrm{d}x}\begin{pmatrix} T_0(x) \\ T_1(x) \\ T_2(x) \\ \vdots \end{pmatrix} = \begin{pmatrix} 0 & & & \\ 1 & 2 & & \\ & & 3 & \\ & & & \ddots \end{pmatrix} \begin{pmatrix} U_0(x) \\ U_1(x) \\ U_2(x) \\ \vdots \end{pmatrix}$$

which lets us know that, for

$$f(x) = (T_0(x), T_1(x), \ldots) \begin{pmatrix} f_0 \\ f_1 \\ \vdots \end{pmatrix}$$

we have a derivative operator in coefficient space as

$$f'(x) = (U_0(x), U_1(x), \ldots) \begin{pmatrix} 0 & 1 & & & \\ & & 2 & & \\ & & & 3 & \\ & & & & \ddots \end{pmatrix} \begin{pmatrix} f_0 \\ f_1 \\ \vdots \end{pmatrix}$$

Demonstration Here we see that applying a matrix to a vector of coefficients successfully calculates the derivative:

```
using ApproxFun, Plots, LinearAlgebra
f = Fun(x -> cos(x^2), Chebyshev())   # f is expanded in Chebyshev coefficients
n = ncoefficients(f)   # This is the number of coefficients
D = zeros(n-1,n)
for k=1:n-1
    D[k,k+1] = k
end
D
```

```
31×32 Matrix{Float64}:
 0.0  1.0  0.0  0.0  0.0  0.0  0.0  …   0.0   0.0   0.0   0.0   0.0   0.0
 0.0  0.0  2.0  0.0  0.0  0.0  0.0      0.0   0.0   0.0   0.0   0.0   0.0
 0.0  0.0  0.0  3.0  0.0  0.0  0.0      0.0   0.0   0.0   0.0   0.0   0.0
 0.0  0.0  0.0  0.0  4.0  0.0  0.0      0.0   0.0   0.0   0.0   0.0   0.0
 0.0  0.0  0.0  0.0  0.0  5.0  0.0      0.0   0.0   0.0   0.0   0.0   0.0
 0.0  0.0  0.0  0.0  0.0  0.0  6.0  …   0.0   0.0   0.0   0.0   0.0   0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0      0.0   0.0   0.0   0.0   0.0   0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0      0.0   0.0   0.0   0.0   0.0   0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0      0.0   0.0   0.0   0.0   0.0   0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0      0.0   0.0   0.0   0.0   0.0   0.0
 ⋮                        ⋮          ⋱                              ⋮
 0.0  0.0  0.0  0.0  0.0  0.0  0.0      0.0   0.0   0.0   0.0   0.0   0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0      0.0   0.0   0.0   0.0   0.0   0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0      0.0   0.0   0.0   0.0   0.0   0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  …  26.0   0.0   0.0   0.0   0.0   0.0
```

```
 0.0  0.0  0.0  0.0  0.0  0.0  0.0      0.0  27.0   0.0   0.0   0.0   0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0      0.0   0.0  28.0   0.0   0.0   0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0      0.0   0.0   0.0  29.0   0.0   0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0      0.0   0.0   0.0   0.0  30.0   0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0   0.0   0.0   0.0   0.0  31.0
```

Here `D*f.coefficients` gives the vector of coefficients corresponding to the derivative, but now the coefficients are in the $U_n(x)$ basis, that is, `Ultraspherical(1)`:

```
fp = Fun(Ultraspherical(1), D*f.coefficients)
fp(0.1)
```

```
-0.001999966666833569
```

Indeed, it matches the "true" derivative:

```
f'(0.1),-2*0.1*sin(0.1^2)
```

```
(-0.0019999666668335634, -0.0019999666668333335)
```

Note that in ApproxFun.jl we can construct these operators rather nicely:

```
D = Derivative()
(D*f)(0.1)
```

```
-0.001999966666833569
```

Here we see that we can produce the ∞-dimensional version as follows:

```
D : Chebyshev() → Ultraspherical(1)
```

```
ConcreteDerivative : Chebyshev() → Ultraspherical(1)
 ·  1.0   ·    ·    ·    ·    ·    ·    ·    ·   ·
 ·   ·   2.0   ·    ·    ·    ·    ·    ·    ·   ·
 ·   ·    ·   3.0   ·    ·    ·    ·    ·    ·   ·
 ·   ·    ·    ·   4.0   ·    ·    ·    ·    ·   ·
 ·   ·    ·    ·    ·   5.0   ·    ·    ·    ·   ·
 ·   ·    ·    ·    ·    ·   6.0   ·    ·    ·   ·
 ·   ·    ·    ·    ·    ·    ·   7.0   ·    ·   ·
 ·   ·    ·    ·    ·    ·    ·    ·   8.0   ·   ·
 ·   ·    ·    ·    ·    ·    ·    ·    ·   9.0  ·
 ·   ·    ·    ·    ·    ·    ·    ·    ·    ·   ⋱
 ·   ·    ·    ·    ·    ·    ·    ·    ·    ·   ⋱
```

3.1.2 Conversion

We can convert between any two polynomial bases using a lower triangular operator, because their spans are equivalent. In the case of Chebyshev and ultraspherical polynomials, they have the added property that they are banded.

Proposition (Chebyshev T-to-U conversion)

$$
\begin{aligned}
T_0(x) &= U_0(x) \\
T_1(x) &= \frac{U_1(x)}{2} \\
T_n(x) &= \frac{U_n(x)}{2} - \frac{U_{n-2}(x)}{2}
\end{aligned}
$$

Proof

Before we do the proof, note that the fact that there are limited non-zero entries follows immediately: if $m < n-2$ we have

$$\langle T_n, U_m\rangle_{\mathrm{U}} = \langle T_n, (1-x^2)U_m\rangle_{\mathrm{T}} = 0$$

To actually determine the entries, we use the trigonometric formulae. Recall for $x = (z + z^{-1})/2$, $z = \mathrm{e}^{\mathrm{i}\theta}$, we have

$$\begin{aligned} T_n(x) &= \cos n\theta = \frac{z^{-n}+z^n}{2} \\ U_n(x) &= \frac{\sin(n+1)\theta}{\sin\theta} = \frac{z^{n+1}-z^{-n-1}}{z-z^{-1}} = z^{-n} + z^{2-n} + \cdots + \cdots + z^{n-2} + z^n \end{aligned}$$

The result follows immediately.

■

Corollary (Ultraspherical λ-to-(λ+1) conversion)

$$C_n^{(\lambda)}(x) = \frac{\lambda}{n+\lambda}(C_n^{(\lambda+1)}(x) - C_{n-2}^{(\lambda+1)}(x))$$

Proof This follows from differentiating the previous result. For example:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}x}T_0(x) &= \frac{\mathrm{d}}{\mathrm{d}x}U_0(x) \\ \frac{\mathrm{d}}{\mathrm{d}x}T_1(x) &= \frac{\mathrm{d}}{\mathrm{d}x}\frac{U_1(x)}{2} \\ \frac{\mathrm{d}}{\mathrm{d}x}T_n(x) &= \frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{U_n(x)}{2} - \frac{U_{n-2}}{2}\right) \end{aligned}$$

becomes

$$\begin{aligned} 0 &= 0 \\ U_0(x) &= C_0^{(2)}(x) \\ nU_{n-1}(x) &= C_{n-1}^{(2)}(x) - C_{n-3}^{(2)}(x) \end{aligned}$$

Differentiating this repeatedly completes the proof.

■

Note we can write this in matrix form, for example, we have

$$\underbrace{\begin{pmatrix} 1 & & & \\ 0 & \frac{1}{2} & & \\ -\frac{1}{2} & 0 & \frac{1}{2} & \\ & \ddots & \ddots & \ddots \end{pmatrix}}_{(R_T^U)^\top} \begin{pmatrix} U_0(x) \\ U_1(x) \\ U_2(x) \\ \vdots \end{pmatrix} = \begin{pmatrix} T_0(x) \\ T_1(x) \\ T_2(x) \\ \vdots \end{pmatrix}$$

therefore,

$$f(x) = (T_0(x), T_1(x), \ldots) \begin{pmatrix} f_0 \\ f_1 \\ \vdots \end{pmatrix} = (U_0(x), U_1(x), \ldots) R_T^U \begin{pmatrix} f_0 \\ f_1 \\ \vdots \end{pmatrix}$$

Again, we can construct this nicely in ApproxFun:

```
R_TU = I : Chebyshev() → Ultraspherical(1)

f = Fun(exp, Chebyshev())
g = R_TU*f

g(0.1) , exp(0.1)

(1.1051709180756477, 1.1051709180756477)
```

3.1.3 Ultraspherical Three-term recurrence

Theorem (three-term recurrence for Chebyshev U)

$$\begin{aligned} xU_0(x) &= \frac{U_1(x)}{2} \\ xU_n(x) &= \frac{U_{n-1}(x)}{2} + \frac{U_{n+1}(x)}{2} \end{aligned}$$

Proof Differentiating

$$\begin{aligned} xT_0(x) &= T_1(x) \\ xT_n(x) &= \frac{T_{n-1}(x)}{2} + \frac{T_{n+1}(x)}{2} \end{aligned}$$

we get

$$\begin{aligned} T_0(x) &= U_0(x) \\ T_n(x) + nxU_{n-1}(x) &= \frac{(n-1)U_{n-2}(x)}{2} + \frac{(n+1)U_n(x)}{2} \end{aligned}$$

substituting in the conversion $T_n(x) = (U_n(x) - U_{n-2}(x))/2$ we get

$$\begin{aligned} T_0(x) &= U_0(x) \\ nxU_{n-1}(x) &= \frac{(n-1)U_{n-2}(x)}{2} + \frac{(n+1)U_n(x)}{2} - (U_n(x) - U_{n-2}(x))/2 = \frac{nU_{n-2}(x)}{2} + \frac{nU_n(x)}{2} \end{aligned}$$

■

Differentiating this theorem again and applying the conversion we get the following

Corollary (three-term recurrence for ultraspherical)

$$xC_0^{(\lambda)}(x) = \frac{1}{2\lambda}C_1^{(\lambda)}(x)$$
$$xC_n^{(\lambda)}(x) = \frac{n+2\lambda-1}{2(n+\lambda)}C_{n-1}^{(\lambda)}(x) + \frac{n+1}{2(n+\lambda)}C_{n+1}^{(\lambda)}(x)$$

Here's an example of the Jacobi operator (which is the transpose of the multiplication by x operator):

```
Multiplication(Fun(), Ultraspherical(2))'
```

```
AdjointOperator : Ultraspherical(2) → Ultraspherical(2)
 0.0                 0.25   ...   ⋅       ⋅                    ⋅    ⋅
 0.6666666666666666  0.0          ⋅       ⋅                    ⋅    ⋅
  ⋅                  0.625        ⋅       ⋅                    ⋅    ⋅
  ⋅                   ⋅           ⋅       ⋅                    ⋅    ⋅
  ⋅                   ⋅           ⋅       ⋅                    ⋅    ⋅
  ⋅                   ⋅      ...  ⋅       ⋅                    ⋅    ⋅
  ⋅                   ⋅          0.4375   ⋅                    ⋅    ⋅
  ⋅                   ⋅          0.0      0.4444444444444444   ⋅    ⋅
  ⋅                   ⋅          0.55     0.0                  0.45 ⋅
  ⋅                   ⋅           ⋅       0.5454545454545454   0.0   ⋱
  ⋅                   ⋅      ...  ⋅       ⋅                        ⋱    ⋱
```

3.2 Application: solving differential equations

The preceding results allowed us to represent

1. Differentiation
2. Conversion
3. Multiplication

as banded operators. We will see that we can combine these, along with

4. Evaluation

to solve ordinary differential equations.

3.2.1 First order, constant coefficient differential equations

Consider the simplest ODE:

$$u(0) = 1$$
$$u'(x) - u(x) = 0, \qquad x \in [-1, 1]$$

and suppose we represent $u(x)$ in its Chebyshev expansion, with coefficients to be determined. In other words, we want to calculate coefficients u_k such that

$$u(x) = \sum_{k=0}^{\infty} u_k T_k(x) = (T_0(x), T_1(x), \ldots) \begin{pmatrix} u_0 \\ u_1 \\ \vdots \end{pmatrix}$$

In this case we know that $u(x) = \mathrm{e}^x$, but we would still need other means to calculate u_k (They are definitely not as simple as Taylor series coefficients).

We can express the constraints as acting on the coefficients. For example, we have

$$u(0) = (T_0(0), T_1(0), \ldots) \begin{pmatrix} u_0 \\ u_1 \\ \vdots \end{pmatrix} = (1, 0, -1, 0, 1, \ldots) \begin{pmatrix} u_0 \\ u_1 \\ \vdots \end{pmatrix}$$

We also have

$$u'(x) = (U_0(x), U_1(x), \ldots) \begin{pmatrix} 0 & 1 & & & \\ & 2 & & & \\ & & 3 & & \\ & & & \ddots & \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ \vdots \end{pmatrix}$$

To represent $u'(x) - u(x)$, we need to make sure the bases are compatible. In other words, we want to express $u(x)$ in its $U_k(x)$ basis using the conversion operator R_T^U:

$$u(x) = (U_0(x), U_1(x), \ldots) \begin{pmatrix} 1 & 0 & -\frac{1}{2} & & \\ & \frac{1}{2} & 0 & -\frac{1}{2} & \\ & & \ddots & \ddots & \ddots \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ \vdots \end{pmatrix}$$

Which gives us,

$$u'(x) - u(x) = (U_0(x), U_1(x), \ldots) \begin{pmatrix} -1 & 1 & \frac{1}{2} & & & \\ & -\frac{1}{2} & 2 & \frac{1}{2} & & \\ & & -\frac{1}{2} & 3 & \frac{1}{2} & \\ & & & \ddots & \ddots & \ddots \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ \vdots \end{pmatrix}$$

Combing the differential part and the evaluation part, we arrive at an (infinite) system of equations for the coefficients $u_0, u_1, \ldots$:

$$\begin{pmatrix} 1 & 0 & -1 & 0 & 1 & \cdots \\ -1 & 1 & \frac{1}{2} & & & \\ & -\frac{1}{2} & 2 & \frac{1}{2} & & \\ & & -\frac{1}{2} & 3 & \frac{1}{2} & \\ & & & \ddots & \ddots & \ddots \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ \vdots \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \end{pmatrix}$$

How to solve this system is outside the scope of this course (though a simple approach is to truncate the infinite system to finite systems). We can however do this in ApproxFun:

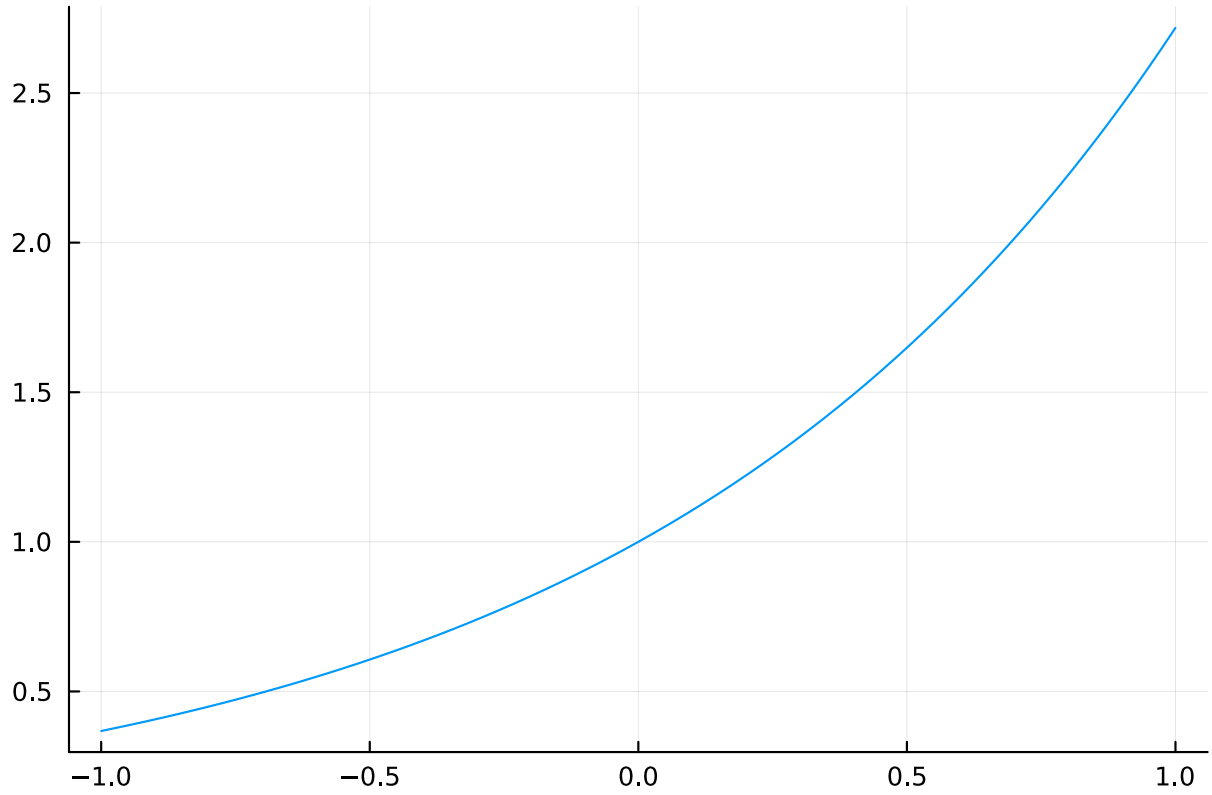
```
B  = Evaluation(0.0) : Chebyshev()
D  = Derivative() : Chebyshev() → Ultraspherical(1)
```

```
R_TU = I : Chebyshev() → Ultraspherical(1)
L = [B;
     D - R_TU]
```

```
InterlaceOperator : Chebyshev() → 2-element ArraySpace:
Space{D, Float64} where D[ConstantSpace(Point(0.0)), Ultraspherical(1)]
  1.0   0.0  -1.0   0.0   1.0   0.0  -1.0   0.0   1.0  0.0  ⋯
 -1.0   1.0   0.5   0.0   0.0   0.0   0.0   0.0   0.0  0.0  ⋱
  0.0  -0.5   2.0   0.5   0.0   0.0   0.0   0.0   0.0  0.0  ⋱
  0.0   0.0  -0.5   3.0   0.5   0.0   0.0   0.0   0.0  0.0  ⋱
  0.0   0.0   0.0  -0.5   4.0   0.5   0.0   0.0   0.0  0.0  ⋱
  0.0   0.0   0.0   0.0  -0.5   5.0   0.5   0.0   0.0  0.0  ⋱
  0.0   0.0   0.0   0.0   0.0  -0.5   6.0   0.5   0.0  0.0  ⋱
  0.0   0.0   0.0   0.0   0.0   0.0  -0.5   7.0   0.5  0.0  ⋱
  0.0   0.0   0.0   0.0   0.0   0.0   0.0  -0.5   8.0  0.5  ⋱
  0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0  -0.5  9.0  ⋱
  ⋮     ⋱     ⋱     ⋱     ⋱     ⋱     ⋱     ⋱     ⋱    ⋱    ⋱
```

We can solve this system as follows:

```
u = L \ [1; 0]
plot(u;legend=false)
```



It matches the "true" result:

Note we can incorporate right-hand sides as well, for example, to solve $u'(x) - u(x) = f(x)$, by expanding f in its Chebyshev U series.

3.2.2 Second-order constant coefficient equations

This approach extends to second-order constant-coefficient equations by using ultraspherical polynomials. Consider

$$\begin{aligned} u(-1) &= 1 \\ u(1) &= 0 \\ u''(x) + u'(x) + u(x) &= 0 \end{aligned}$$

Evaluation works as in the first-order case. To handle second-derivatives, we need $C^{(2)}$ polynomials:

```
D_0 = Derivative() : Chebyshev() → Ultraspherical(1)
D_1 = Derivative() : Ultraspherical(1) → Ultraspherical(2)
D_1*D_0  # 2 zeros not printed in (1,1) and (1,2) entry
```

```
ConcreteDerivative : Chebyshev() → Ultraspherical(2)
 ⋅  ⋅  4.0   ⋅    ⋅    ⋅    ⋅    ⋅    ⋅    ⋅   ⋅
 ⋅  ⋅   ⋅   6.0   ⋅    ⋅    ⋅    ⋅    ⋅    ⋅   ⋅
 ⋅  ⋅   ⋅    ⋅   8.0   ⋅    ⋅    ⋅    ⋅    ⋅   ⋅
 ⋅  ⋅   ⋅    ⋅    ⋅   10.0   ⋅    ⋅    ⋅    ⋅   ⋅
 ⋅  ⋅   ⋅    ⋅    ⋅    ⋅   12.0   ⋅    ⋅    ⋅   ⋅
 ⋅  ⋅   ⋅    ⋅    ⋅    ⋅    ⋅   14.0   ⋅    ⋅   ⋅
 ⋅  ⋅   ⋅    ⋅    ⋅    ⋅    ⋅    ⋅   16.0   ⋅   ⋅
 ⋅  ⋅   ⋅    ⋅    ⋅    ⋅    ⋅    ⋅    ⋅   18.0  ⋅
 ⋅  ⋅   ⋅    ⋅    ⋅    ⋅    ⋅    ⋅    ⋅    ⋅   ⋱
 ⋅  ⋅   ⋅    ⋅    ⋅    ⋅    ⋅    ⋅    ⋅    ⋅   ⋱
 ⋅  ⋅   ⋅    ⋅    ⋅    ⋅    ⋅    ⋅    ⋅    ⋅   ⋱
```

For the identity operator, we use two conversion operators:

```
R_TU = I : Chebyshev() → Ultraspherical(1)
R_U2 = I : Ultraspherical(1) → Ultraspherical(2)
R_T2 = R_U2*R_TU
```

```
TimesOperator : Chebyshev() → Ultraspherical(2)
 1.0  0.0   -0.6666666666666666    0.0    …    ⋅                    ⋅
 ⋅
  ⋅   0.25   0.0                  -0.375       ⋅                    ⋅
 ⋅
  ⋅    ⋅     0.16666666666666666   0.0         ⋅                    ⋅
 ⋅
  ⋅    ⋅      ⋅                    0.125       ⋅                    ⋅
 ⋅
  ⋅    ⋅      ⋅                     ⋅          0.07142857142857142  ⋅
 ⋅
  ⋅    ⋅      ⋅                     ⋅     …    0.0                  0.0625
 ⋅
  ⋅    ⋅      ⋅                     ⋅         -0.12698412698412698  0.0
 ⋱
  ⋅    ⋅      ⋅                     ⋅          0.0                 -0.1125
 ⋱
  ⋅    ⋅      ⋅                     ⋅          0.05555555555555555  0.0
```

```
⋱
 ⋮    ⋮      ⋮                      ⋮          ⋮                    0.05
⋱
 ⋮    ⋮      ⋮                      ⋮     ⋯    ⋮                     ⋮
⋱
```

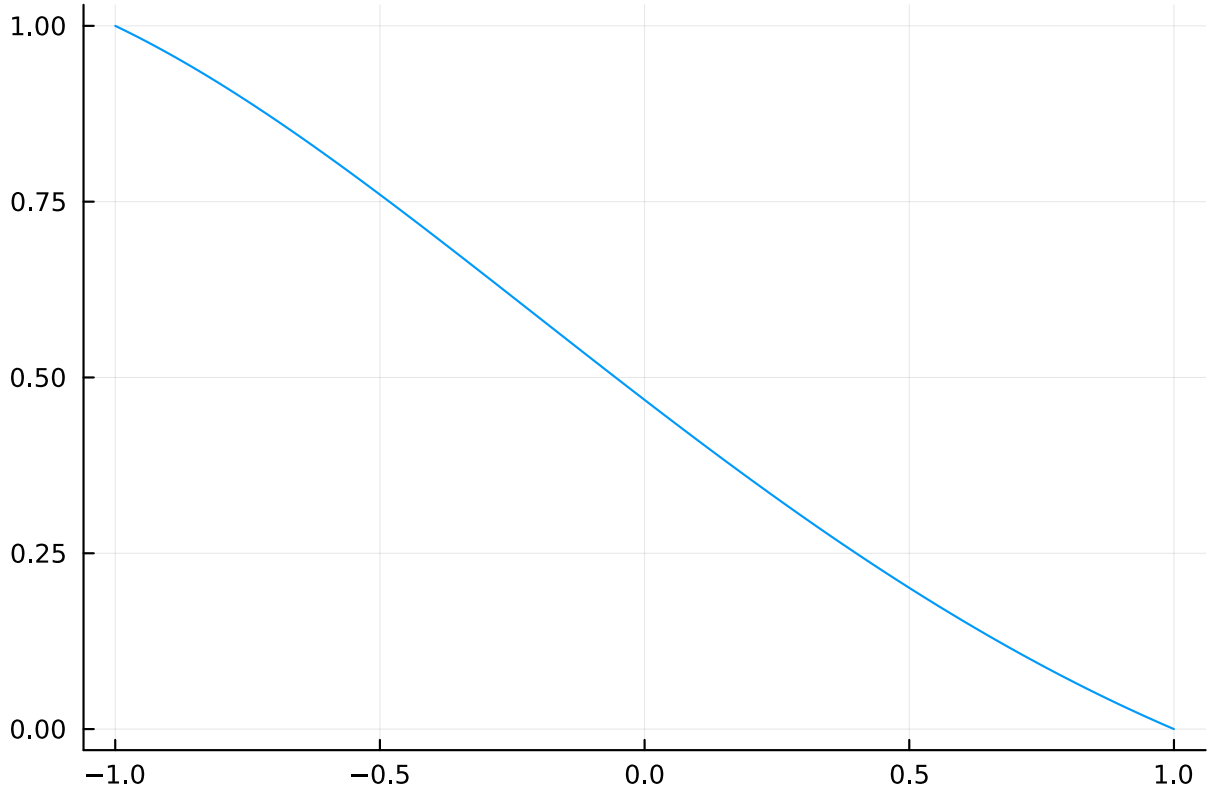
And for the first derivative, we use a derivative and then a conversion:

```
R_U2*D_0  # or could have been D_1*R_TU
```

```
TimesOperator : Chebyshev() → Ultrasphertical(2)
 ⋅  1.0  0.0  -1.0    ⋅     ⋅     ⋅     ⋅     ⋅     ⋅   ⋅
 ⋅   ⋅   1.0   0.0  -1.0    ⋅     ⋅     ⋅     ⋅     ⋅   ⋅
 ⋅   ⋅    ⋅    1.0   0.0  -1.0    ⋅     ⋅     ⋅     ⋅   ⋅
 ⋅   ⋅    ⋅     ⋅    1.0   0.0  -1.0    ⋅     ⋅     ⋅   ⋅
 ⋅   ⋅    ⋅     ⋅     ⋅    1.0   0.0  -1.0    ⋅     ⋅   ⋅
 ⋅   ⋅    ⋅     ⋅     ⋅     ⋅    1.0   0.0  -1.0    ⋅   ⋅
 ⋅   ⋅    ⋅     ⋅     ⋅     ⋅     ⋅    1.0   0.0  -1.0  ⋅
 ⋅   ⋅    ⋅     ⋅     ⋅     ⋅     ⋅     ⋅    1.0   0.0  ⋱
 ⋅   ⋅    ⋅     ⋅     ⋅     ⋅     ⋅     ⋅     ⋅    1.0  ⋱
 ⋅   ⋅    ⋅     ⋅     ⋅     ⋅     ⋅     ⋅     ⋅     ⋅   ⋱
 ⋅   ⋅    ⋅     ⋅     ⋅     ⋅     ⋅     ⋅     ⋅     ⋅   ⋱
```

Putting everything together we get:

```
B_−_1 = Evaluation(-1) : Chebyshev()
B_1 = Evaluation(1) : Chebyshev()
# u(-1)
# u(1)
# u'' + u' +u

L = [B_−_1;
     B_1;
     D_1*D_0 + R_U2*D_0 + R_U2*R_TU]

u = L \ [1.0,0.0,0.0]
plot(u;legend=false)
```



3.2.3 Variable coefficients

Consider the Airy ODE

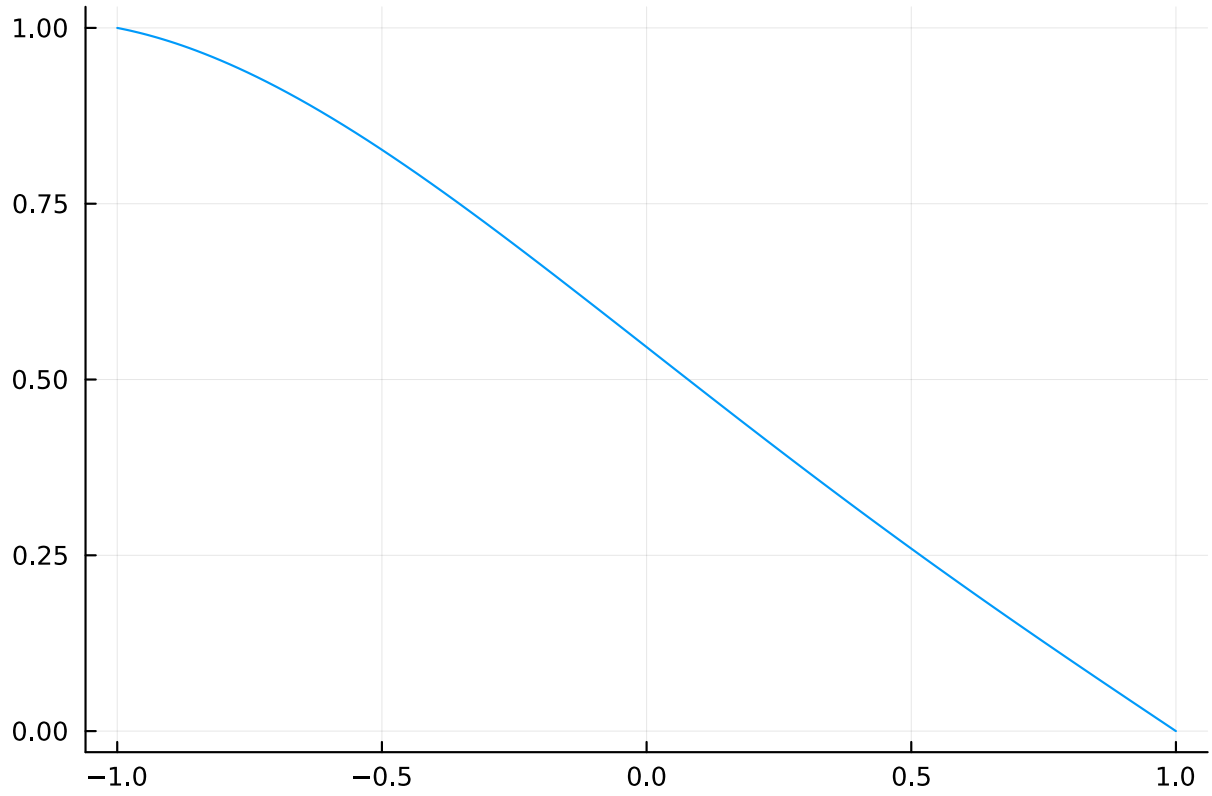
$$\begin{aligned} u(-1) &= 1 \\ u(1) &= 0 \\ u''(x) - xu(x) &= 0 \end{aligned}$$

to handle this, we need only use the Jacobi operator to represent multiplication by x:

```
x = Fun()
X = Multiplication(x) : Chebyshev() → Chebyshev()  # transpose of the Jacobi operator

ConcreteMultiplication : Chebyshev() → Chebyshev()
 0.0  0.5   ·    ·    ·    ·    ·    ·    ·    ·   ·
 1.0  0.0  0.5   ·    ·    ·    ·    ·    ·    ·   ·
  ·   0.5  0.0  0.5   ·    ·    ·    ·    ·    ·   ·
  ·    ·   0.5  0.0  0.5   ·    ·    ·    ·    ·   ·
  ·    ·    ·   0.5  0.0  0.5   ·    ·    ·    ·   ·
  ·    ·    ·    ·   0.5  0.0  0.5   ·    ·    ·   ·
  ·    ·    ·    ·    ·   0.5  0.0  0.5   ·    ·   ·
  ·    ·    ·    ·    ·    ·   0.5  0.0  0.5   ·   ·
  ·    ·    ·    ·    ·    ·    ·   0.5  0.0  0.5  ·
  ·    ·    ·    ·    ·    ·    ·    ·   0.5  0.0  ⋱
  ·    ·    ·    ·    ·    ·    ·    ·    ·   ⋱    ⋱
```

We set up the system as follows:

```
L = [B_−_1;    # u(-1)
     B_1 ;    # u(1)
     D_1*D_0 - R_U2*R_TU*X]    # u'' - x*u

u = L \ [1.0;0.0;0.0]
plot(u; legend=false)
```

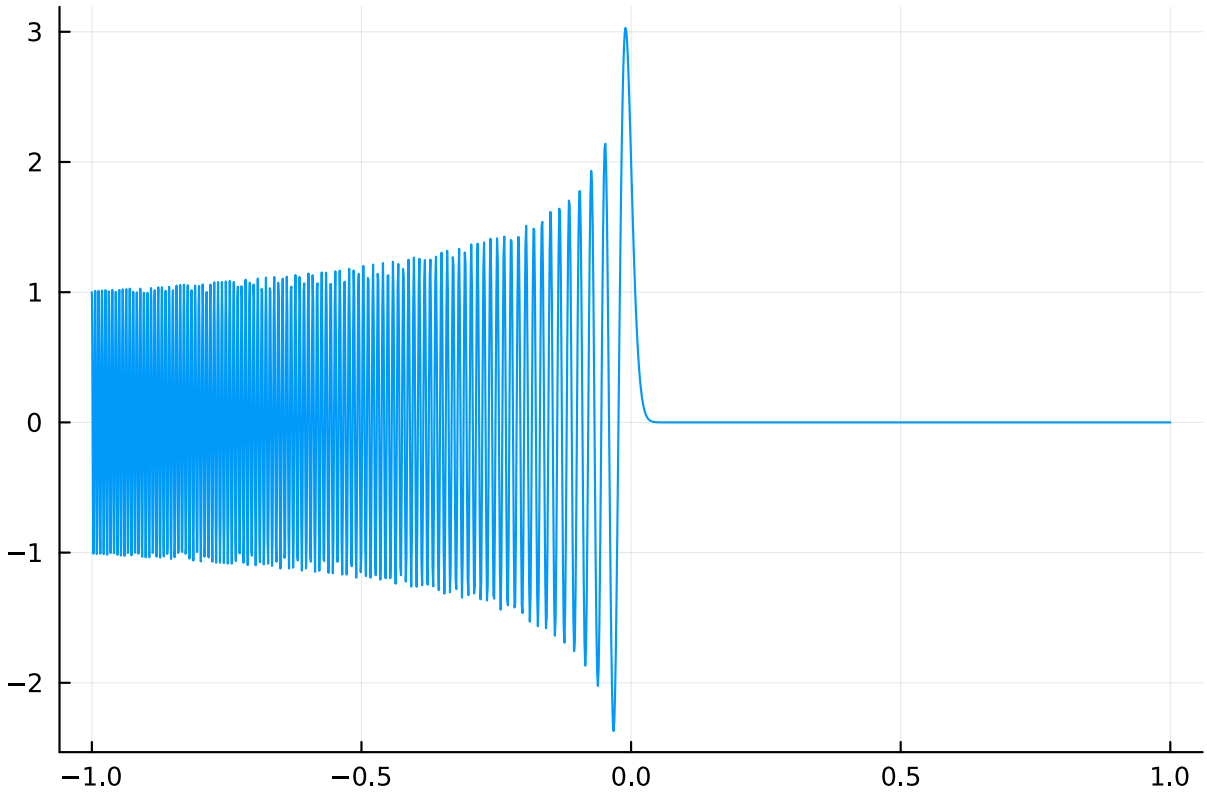


If we introduce a small parameter, that is, solve

$$\begin{aligned} u(-1) &= 1 \\ u(1) &= 0 \\ \epsilon u''(x) - xu(x) &= 0 \end{aligned}$$

we can see it's pretty hard to compute solutions:

```
ε = 1E-6
L = [B_−_1;
     B_1 ;
     ε*D_1*D_0 - R_U2*R_TU*X]

u = L \ [1.0;0.0;0.0]
plot(u; legend=false)
```



Because of the banded structure, this can be solved fast:

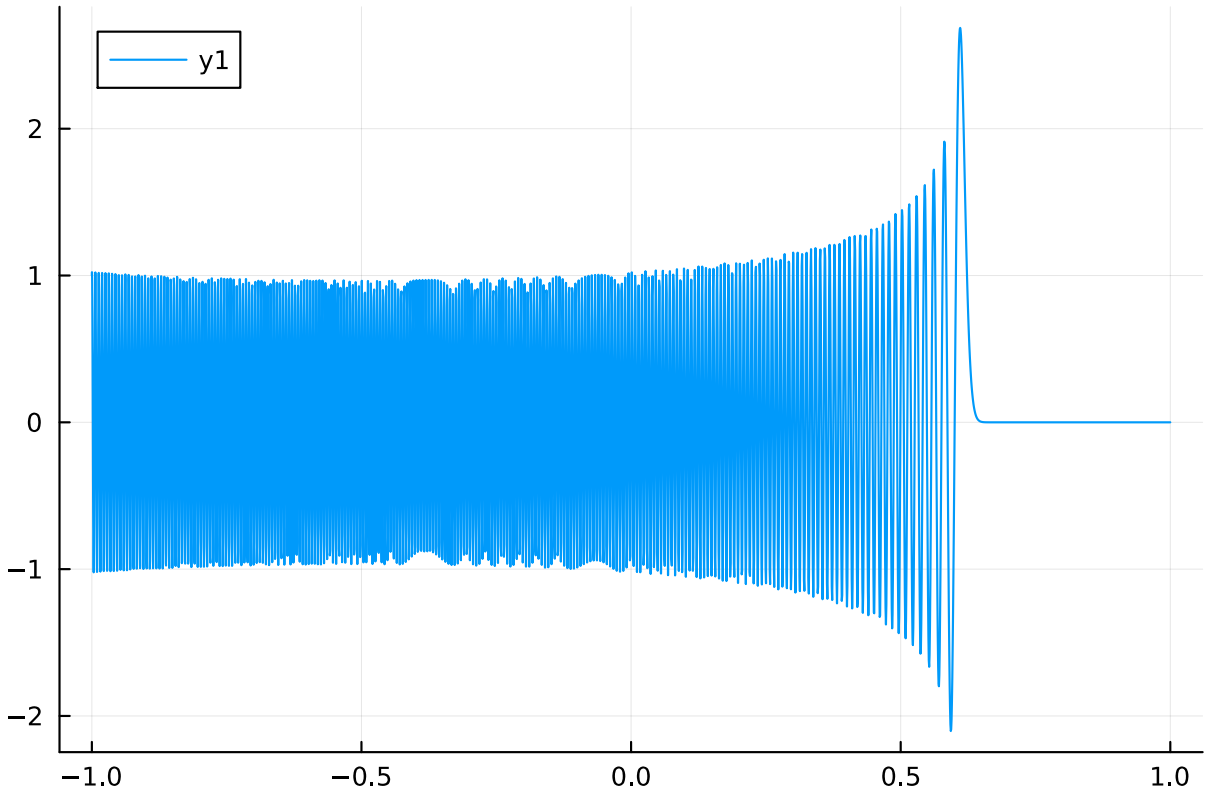
```
ε = 1E-10
L = [B_−_1;
     B_1 ;
     ε*D_1*D_0 - R_U2*R_TU*X]

@time u = L \ [1.0;0.0;0.0]
@show ncoefficients(u);
```

```
1.771782 seconds (11.40 M allocations: 272.910 MiB, 4.27% gc time)
ncoefficients(u) = 62496
```

To handle other variable coefficients, first consider a polynomial $p(x)$. If Multiplication by x is represented by multiplying the coefficients by $J^\top$, then multiplication by p is represented by multiplying the coefficients by $p(J^\top)$:

```
M = -I + X + (X)^2  # represents -1+x+x^2

ε = 1E-6
L = [B_−_1;
     B_1 ;
     ε*D_1*D_0 - R_U2*R_TU*M]

@time u = L \ [1.0;0.0;0.0]

@show ε*u''(0.1) - (-1+0.1+0.1^2)*u(0.1)
plot(u)
```

```
0.053477 seconds (245.78 k allocations: 6.347 MiB)
ε * ((u')')(0.1) - (-1 + 0.1 + 0.1 ^ 2) * u(0.1) = -1.469657728847551e-14
```



For other smooth functions, we first approximate in a polynomial basis, and without loss of generality we use Chebyshev T basis. For example, consider

$$\begin{aligned} u(-1) &= 1 \\ u(1) &= 0 \\ \epsilon u''(x) - \mathrm{e}^x u(x) &= 0 \end{aligned}$$

where

$$\mathrm{e}^x \approx p(x) = \sum_{k=0}^{m-1} p_k T_k(x)$$

Evaluating at a point x, recall Clenshaw's algorithm:

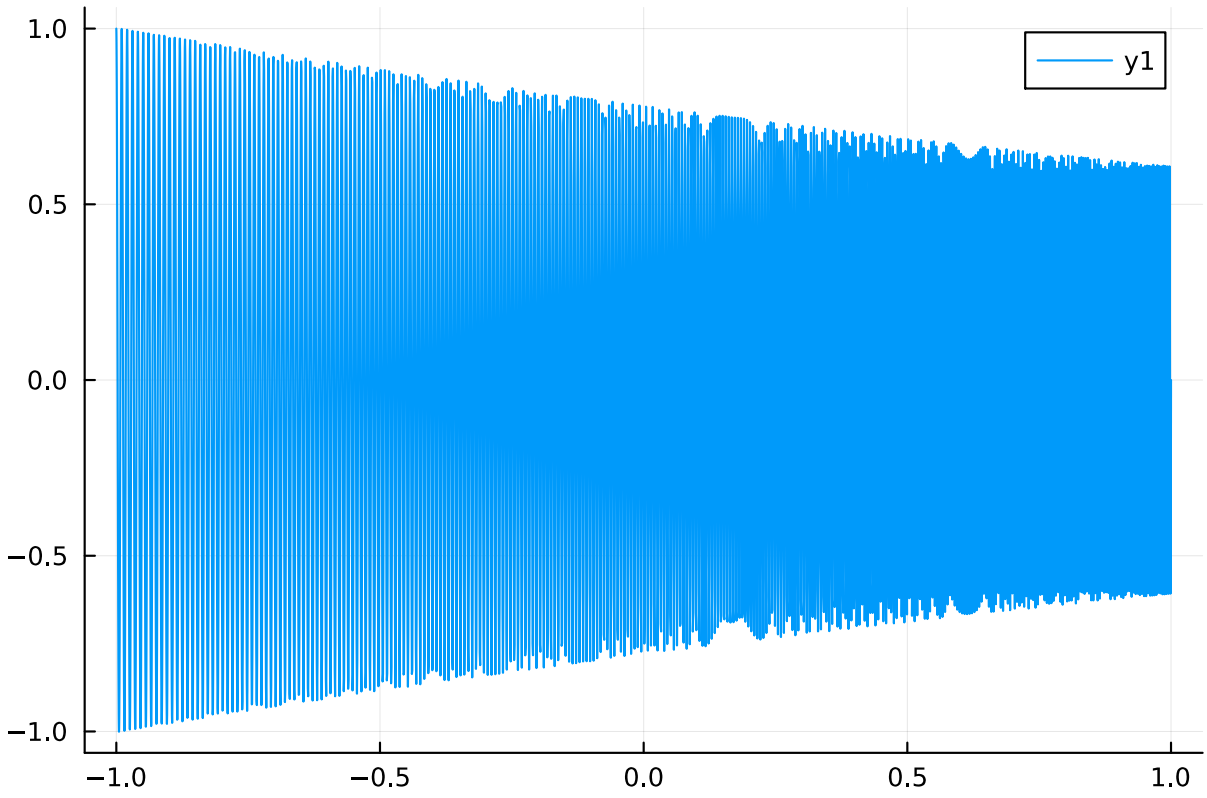
$$\begin{aligned} \gamma_{n-1} &= 2p_{n-1} \\ \gamma_{n-2} &= 2p_{n-2} + 2x\gamma_{n-1} \\ \gamma_{n-3} &= 2p_{n-3} + 2x\gamma_{n-2} - \gamma_{n-1} \\ &\vdots \\ \gamma_1 &= p_1 + x\gamma_2 - \frac{1}{2}\gamma_3 \\ p(x) = \gamma_0 &= p_0 + x\gamma_1 - \frac{1}{2}\gamma_2 \end{aligned}$$

If multiplication by x becomes $J^\top$, then multiplication by $p(x)$ becomes $p(J^\top)$, and hence we calculate:

$$
\begin{aligned}
\Gamma_{n-1} &= 2p_{n-1}I \\
\Gamma_{n-2} &= 2p_{n-2}I + 2J^\top\Gamma_{n-1} \\
\Gamma_{n-3} &= 2p_{n-3}I + 2J^\top\Gamma_{n-2} - \Gamma_{n-1} \\
&\vdots \\
\Gamma_1 &= p_1 I + J^\top\Gamma_2 - \frac{1}{2}\Gamma_3 \\
p(J^\top) = \Gamma_0 &= p_0 + J^\top\Gamma_1 - \frac{1}{2}\Gamma_2
\end{aligned}
$$

Here is an example:

```
p = Fun(exp, Chebyshev()) # polynomial approximation to exp(x)
M = Multiplication(p) : Chebyshev() # constructed using Clenshaw:

ε = 1E-6
L = [B_-_1;
     B_1 ;
     ε*D_1*D_0 + R_U2*R_TU*M]

@time u = L \ [1.0;0.0;0.0]

@show ε*u''(0.1) + exp(0.1)*u(0.1)
plot(u)

0.158782 seconds (908.09 k allocations: 20.127 MiB)
ε * ((u')')(0.1) + exp(0.1) * u(0.1) = 5.10702591327572e-15
```



4 Differential equations satisfied by orthogonal polynomials

This lecture we do the following:

1. Differential equations for orthogonal polynomials
   - SturmLiouville equations
   - Weighted differentiation for ultraspherical polynomials
   - Differential equation for ultraspherical polynomials
2. Application: Eigenstates of Schrödinger operators with quadratic potentials

The three classical weights are (Hermite) $w(x) = \mathrm{e}^{-x^2}$, (Laguerre) $w_\alpha(x) = x^\alpha \mathrm{e}^{-x}$ and (Jacobi) $w_{\alpha,\beta}(x) = (1-x)^\alpha(1+x)^\beta$. Note all weights form a simple hierarchy: when differentiated, they give a linear polynomial times the previous weight in the hierarchy. For Hermite,

$$\frac{\mathrm{d}}{\mathrm{d}x} w(x) = -2xw(x)$$

for Laguerre,

$$\frac{\mathrm{d}}{\mathrm{d}x} w^{(\alpha)}(x) = (\alpha - x) w^{(\alpha-1)}(x)$$

and for Jacobi

$$\frac{\mathrm{d}}{\mathrm{d}x} w^{(\alpha,\beta)}(x) = (\beta(1-x) - \alpha(1+x)) w^{(\alpha-1,\beta-1)}(x)$$

These relationships lead to simple differential equations that have the classical orthogonal polynomials as eigenfunctions.

4.0.1 SturmLiouville operator

We first consider a simple class of operators that are self-adjoint:

Proposition (SturmLiouville self-adjointness) Consider the weighted inner product

$$\langle f, g \rangle_w = \int_a^b f(x)g(x)w(x)\mathrm{d}x$$

then for any continuously differentiable function $q(x)$ satisfying $q(a) = q(b) = 0$, the operator

$$Lu = \frac{1}{w(x)} \frac{\mathrm{d}}{\mathrm{d}x} \left[q(x) \frac{\mathrm{d}u}{\mathrm{d}x} \right]$$

is self-adjoint in the sense

$$\langle Lf, g \rangle_w = \langle f, Lg \rangle_w$$

Proof Simple integration by parts argument:

$$\langle Lf, g\rangle_w = \int_a^b \frac{\mathrm{d}}{\mathrm{d}x}\left[q(x)\frac{\mathrm{d}u}{\mathrm{d}x}\right] g(x)\mathrm{d}x = -\int_a^b q(x)\frac{\mathrm{d}u}{\mathrm{d}x}\frac{\mathrm{d}g}{\mathrm{d}x}\mathrm{d}x = \int_a^b u(x)\frac{\mathrm{d}}{\mathrm{d}x}\left[q(x)\frac{\mathrm{d}g}{\mathrm{d}x}\right]\mathrm{d}x = \int_a^b u(x)\frac{1}{w(x)}\frac{\mathrm{d}}{\mathrm{d}x}\left[q$$

■

We claim that the classical orthogonal polynomials are eigenfunctions of a SturmLiouville problem, that is, in each case there exists a $q(x)$ so that

$$Lp_n(x) = \lambda_n p_n(x)$$

where λ_n is the (real) eigenvalue. We will derive this for the ultraspherical polynomials.

4.0.2 Weighted differentiation for ultraspherical polynomials

We have already seen that Chebyshev and ultraspherical polynomials have simple expressions for derivatives where we decrement the degree and increment the parameter:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}x}T_n(x) &= nU_{n-1}(x) = nC_{n-1}^{(1)}(x)\\ \frac{\mathrm{d}}{\mathrm{d}x}C_n^{(\lambda)}(x) &= 2\lambda C_{n-1}^{(\lambda+1)}(x)\end{aligned}$$

In this section, we see that differentiating the weighted polynomials actually decrements the parameter and increments the degree:

Proposition (weighted differentiation)

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}x}[\sqrt{1-x^2}U_n(x)] &= -\frac{n+1}{\sqrt{1-x^2}}T_{n+1}(x)\\ \frac{\mathrm{d}}{\mathrm{d}x}[(1-x^2)^{\lambda-\frac{1}{2}}C_n^{(\lambda)}(x)] &= -\frac{(n+1)(n+2\lambda-1)}{2(\lambda-1)}(1-x^2)^{\lambda-\frac{3}{2}}C_{n+1}^{(\lambda-1)}(x)\end{aligned}$$

Proof We show the first result by showing that the left-hand side is orthogonal to all polynomials of degree less than $n+1$ by integration by parts:

$$\langle\sqrt{1-x^2}\frac{\mathrm{d}}{\mathrm{d}x}[\sqrt{1-x^2}U_n(x)], p_m(x)\rangle_{\mathrm{T}} = -\int_{-1}^1 \sqrt{1-x^2}U_n(x)p_m'\mathrm{d}x = 0$$

Note that

$$\sqrt{1-x^2}\frac{\mathrm{d}}{\mathrm{d}x}\sqrt{1-x^2}f(x) = (1-x^2)f'(x) - xf(x)$$

Thus we just have to verify the constant in front:

$$\sqrt{1-x^2}\frac{\mathrm{d}}{\mathrm{d}x}[\sqrt{1-x^2}U_n(x) = (-n-1)2^n x^{n+1}$$

The other ultraspherical polynomial follow similarly. ■

4.0.3 Eigenvalue equation for Ultraspherical polynomials

Note that differentiating increments the parameter and decrements the degree while weighted differentiation decrements the parameter and increments the degree. Therefore combining them brings us back to where we started.

In the case of Chebyshev polynomials, this gives us a SturmLiouville equation:

$$\sqrt{1-x^2}\frac{\mathrm{d}}{\mathrm{d}x}\sqrt{1-x^2}\frac{\mathrm{d}T_n}{\mathrm{d}x} = n\sqrt{1-x^2}\frac{\mathrm{d}}{\mathrm{d}x}\sqrt{1-x^2}U_{n-1}(x) = -n^2T_n(x)$$

Note that the chain rule gives us a simple expression as

$$(1-x^2)\frac{\mathrm{d}^2T_n}{\mathrm{d}x^2} - x\frac{\mathrm{d}T_n}{\mathrm{d}x} = -n^2T_n(x)$$

Similarly,

$$(1-x^2)^{\frac{1}{2}-\lambda}\frac{\mathrm{d}}{\mathrm{d}x}(1-x^2)^{\lambda+\frac{1}{2}}\frac{\mathrm{d}C_n^{(\lambda)}}{\mathrm{d}x} = -n(n+2\lambda)C_n^{(\lambda)}(x)$$

or in other words,

$$(1-x^2)\frac{\mathrm{d}^2C_n^{(\lambda)}}{\mathrm{d}x^2} - (2\lambda+1)x\frac{\mathrm{d}C_n^{(\lambda)}}{\mathrm{d}x} = -\frac{n(n+2\lambda)}{2\lambda}C_n^{(\lambda)}(x)$$

definition three term recurrences Clenshaw examples Jacobi matrix vector space roots of OPs are eigenvalues of Jacobi matrix roots of OPs, quadrature, coefficients, interpolation, transforms Gram Schmidt approximation by expansion in OPs derivatives conversion Multiplication function value space and coefficient space The monomial basis Chebyshev polynomials The Chebyshev Fourier connection Differential equations PDE example

1. Runge phenomenon
2. Chebyshev-Fourier connection
3. Orthogonal polynomials
4. Coefficient space and function value space
5. Boundary-value problems
6. Ultraspherical method, collocation method, finite difference method, Galerkin and finite element method
7. Vector spaces
8. Fourier-orthogonality connection
9. Instability of monomial basis