M3M6: Applied Complex Analysis

Dr. Sheehan Olver

s.olver@imperial.ac.uk

1 Lecture 21: Classical orthogonal polynomials

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

1.1 Definition of classical orthogonal polynomials

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

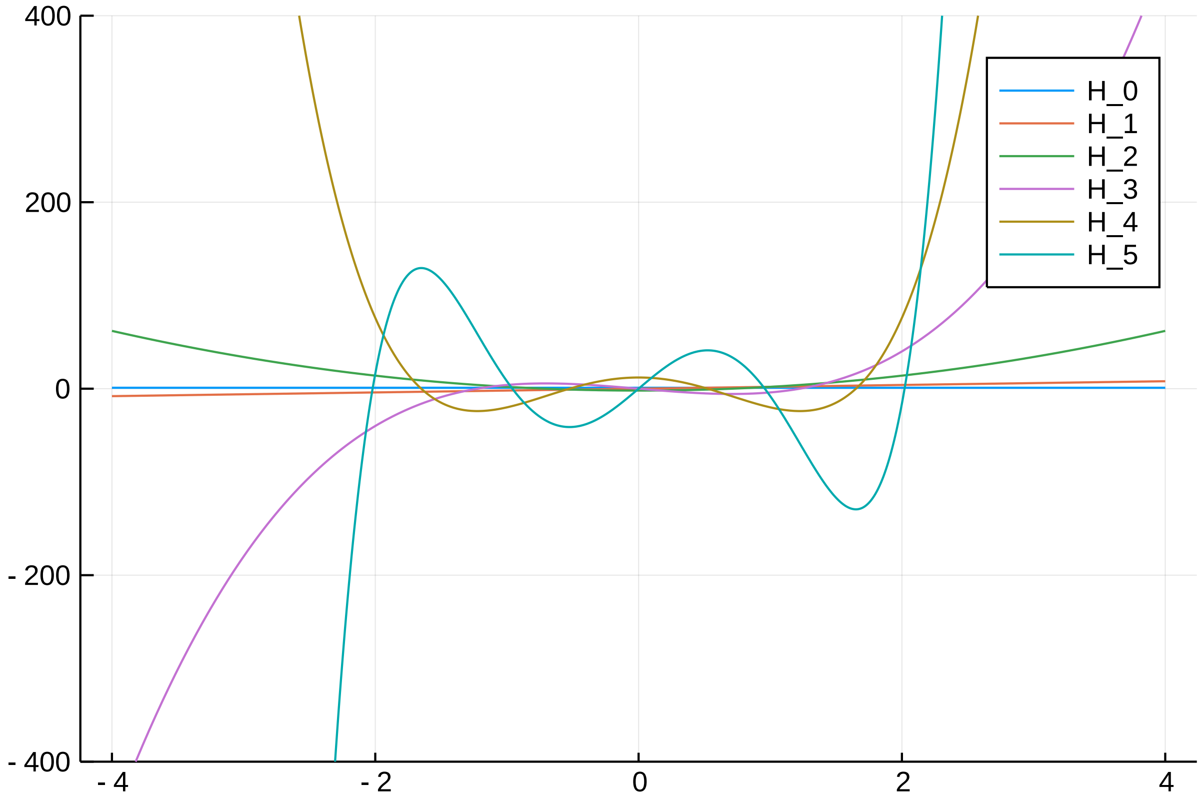
Name	(a, b)	$w(x)$	Notation	k_n
Hermite	$(-\infty, \infty)$	e^{-x^2}	$H_n(x)$	2^n
Laguerre	$(0, \infty)$	$x^\alpha e^{-x}$	$L_n^{(\alpha)}(x)$	Table 18.3.1
Jacobi	$(-1, 1)$	$(1-x)^\alpha (1+x)^\beta$	$P_n^{(\alpha,\beta)}(x)$	Table 18.3.1

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

sum(H_2 * H_5 * w) = 0.0

@show sum(H_5*H_5*w);

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

sum(P_2 * P_5 * w) = -1.235990476633475e-17
```

```julia
@show sum(P_5*P_5*w);
```

```
sum(P_5 * P_5 * w) = 0.21713358248393155
```

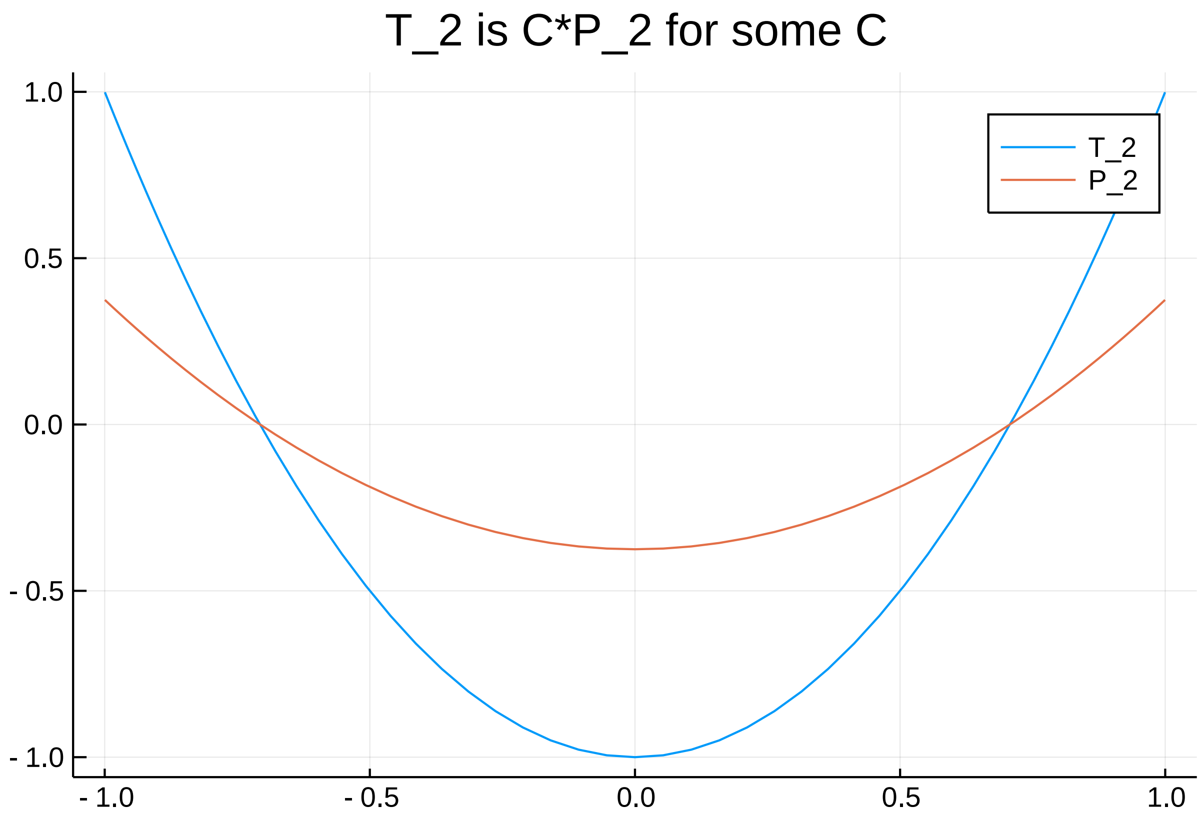
1.2 Legendre, Chebyshev, and ultraspherical polynomials

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 $P_n^{(\alpha,\beta)}$:

```julia
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.

1.2.1 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 $x = \cos\theta$, $\mathrm{d}x = -\sin\theta$ to get (for $n \neq m$)

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

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

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

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

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

1.2.2 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
```

```
n = 14
```

```
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'
```

```
15×14 Adjoint{Float64,Array{Float64,2}}:
 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)
```

```
4.163336342344337e-17
```

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

$$
\begin{aligned}
T_0(x) &= 1 \\
T_1(x) &= xT_0(x) \\
T_2(x) &= 2xT_0(x) - T_0(x) \\
T_3(x) &= 2xT_1(x) - T_1(x) \qquad \vdots
\end{aligned}
$$

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)
```

```
0.000037 seconds (6 allocations: 78.359 KiB)
```

```
@time trig_Chebyshev(n,0.1);
```

```
0.000257 seconds (6 allocations: 78.359 KiB)
```

We can also demonstrate Clenshaw's algorithm for evaluating polynomials. For Chebyshev, 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 tweeks, this can be made more accurate. This is the algorithm used for evaluating functions in ApproxFun:

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

```
0.0
```

1.3 Approximation with Chebyshev polynomials

Last lecture, we used the formula, derived via trigonometric manipulations,

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

Rearranging, this becomes

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

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:

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

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())
@show f.coefficients
```

```
f.coefficients = [1.2660658777520084, 1.1303182079849703, 0.271495339534076
56, 0.044336849848663804, 0.0054742404420936785, 0.0005429263119139036, 4.4
97732295427654e-5, 3.198436462511398e-6, 1.992124804817033e-7, 1.1036771902
153892e-8, 5.505896578301994e-10, 2.4979644681942872e-11, 1.039110420972266
8e-12, 3.9896905223990586e-14]
```

```julia
@show ncoefficients(f)
```

```
ncoefficients(f) = 14
```

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

```
f(0.1) = 1.1051709180756477
```

```julia
@show exp(0.1);
```

```
exp(0.1) = 1.1051709180756477
```

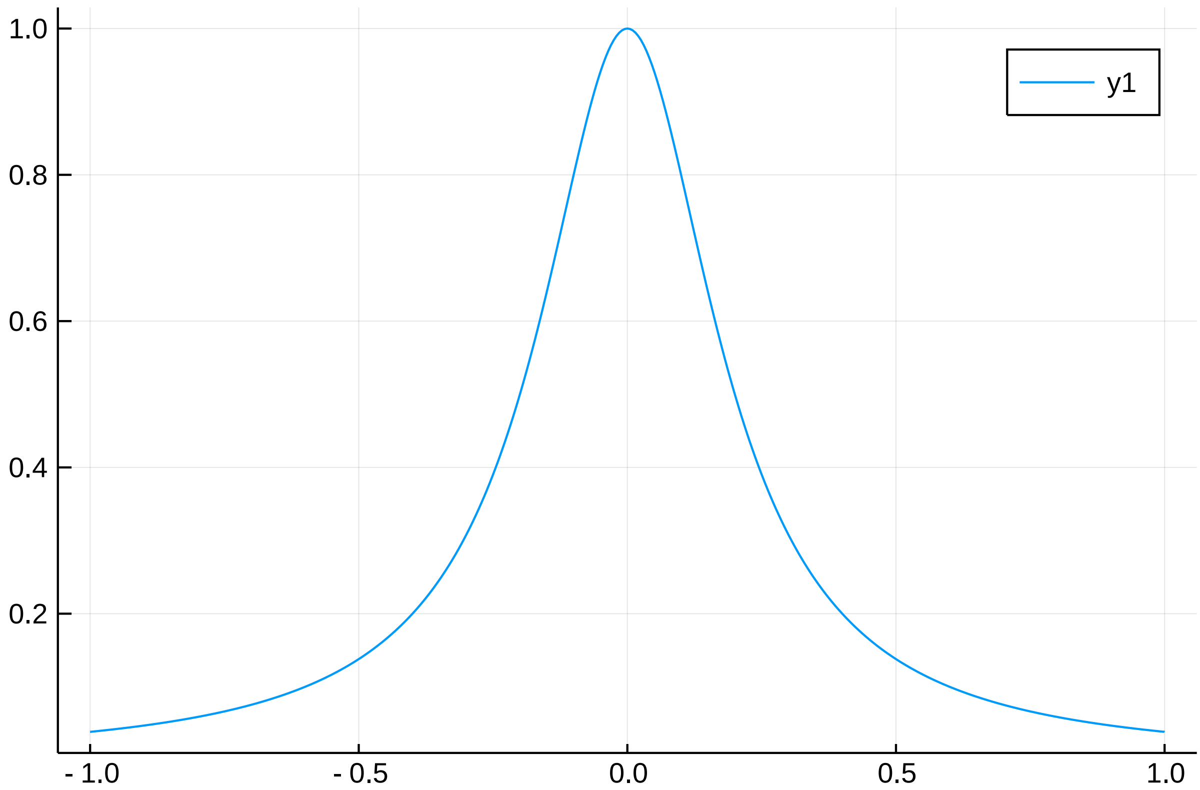
The accuracy of this approximation is typically dictated by the smoothness of f: the more times we can differentiate, the faster it converges. For analytic functions, it's dictated by the domain of analyticity, just like Laurent/Fourier series. In the case above, e^x is entire hence we get faster than 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 $[-1, 1]$:

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

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

```julia
plot(f)
```



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

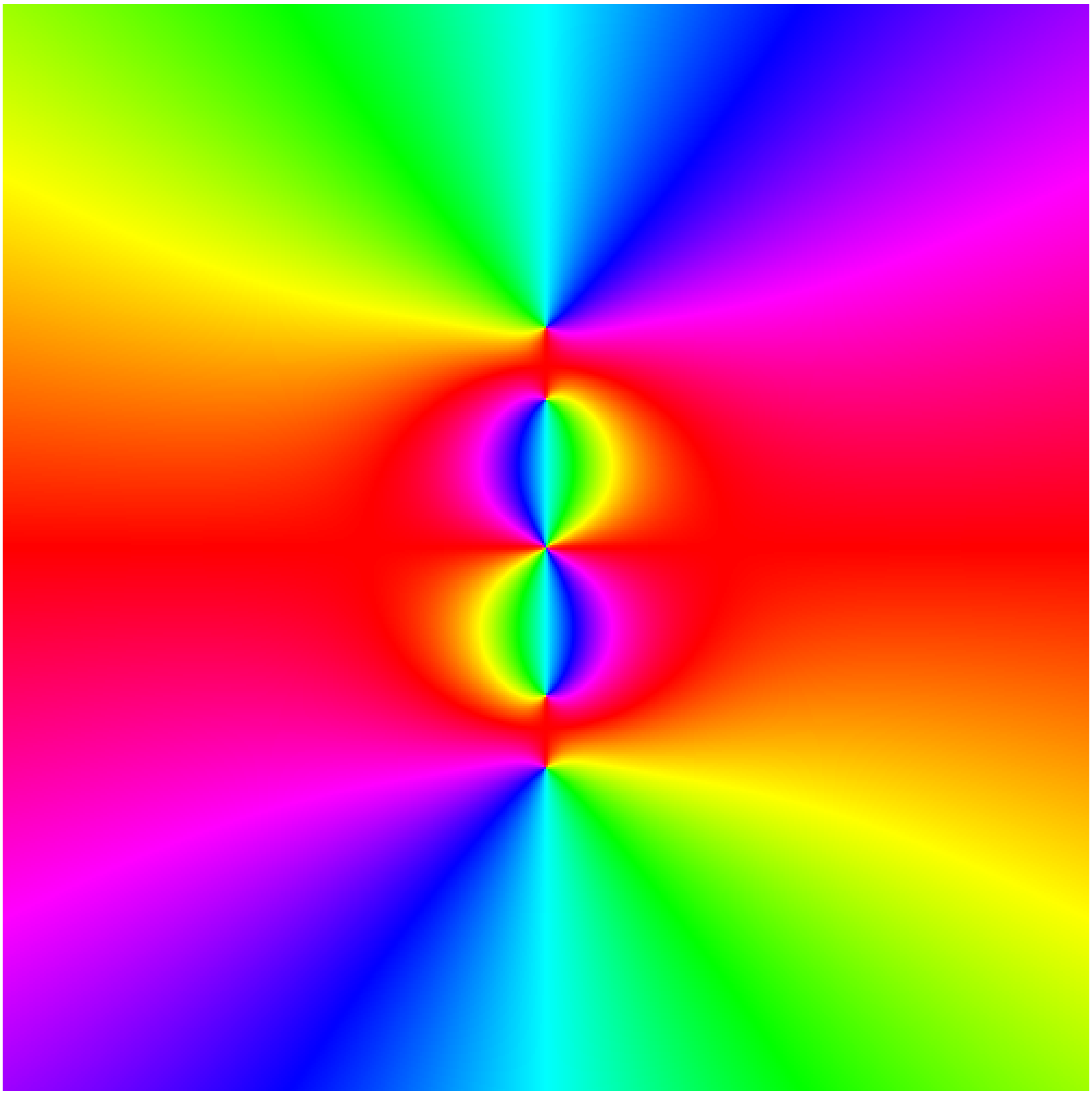
$$f(x) = \sum_{k=0}^{\infty} f_k T_k(x) \Leftrightarrow f(\cos\theta) = \sum_{k=0}^{\infty} f_k \cos k\theta$$

From Lecture 4 we saw that Fourier coefficients decay exponentially (thence the approximation converges exponentially fast) of $f\left(\frac{z+z^{-1}}{2}\right)$ is analytic in an ellipse. In the case of $f(x) = \frac{1}{25x^2+1}$, we find that

$$f(z) = \frac{4z^2}{25 + 54z^2 + 25z^4}$$

which has poles at $ \pm 0.8198040\I,\pm1.2198\I: $

```
f = x -> 1/(25x^2 + 1)
portrait(-3..3, -3..3, z -> f((z+1/z)/2))
```



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

```
f = Fun( x -> 1/(25x^2 + 1), Chebyshev())
```

```
plot(abs.(f.coefficients) .+ 1E-40; yscale=:log10, label="Chebyshev coefficients")
plot!( 1.2198.^(-(0:ncoefficients(f))); label="R^(-k)")
```