1 Chapter 3: Chebyshev methods for non-periodic functions

Here we discuss:

1. Why high-degree polynomial interpolation in equally spaced points typically give poor approximations of functions
2. Why high-degree polynomial interpolants in unequally spaced points are typically excellent approximants of functions
3. Chebyshev polynomials
4. The connection between Chebyshev and Fourier series
5. Computing Chebyshev interpolants and their derivatives with the FFT
6. The solution of the wave equation using the method of separation of variables
7. The numerical solution of the wave equation using a combination of finite differences and Chebyshev interpolants
8. The accuracy of Chebyshev interpolants and their derivatives

In the previous chapter we learned that smooth periodic functions and their derivatives can be well approximated by trigonometric interpolants:

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

We also learnt that 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 approximated by expansions in Chebyshev polynomials,

$$f(x) \approx \sum_{k=0}^{n} \tilde{b}_k^n T_k(x).$$

We'll see that Chebyshev expansions are Fourier expansions in disguise and therefore we can devise efficient approximation methods for PDEs with Chebyshev polynomials that use the FFT.

1.1 The Runge phenomenon

Suppose we want to approximate the function $f(x) = \frac{1}{1+25x^2}$ (known as the Runge function) on the interval $x \in [-1, 1]$. We know that a trigonometric interpolant will not converge very fast to f. Instead, we approximate f by a polynomial interpolant that interpolates f at $n+1$ *equally spaced* points on $[-1, 1]$. One method to compute the interpolating polynomial

is to use the formula for the Lagrange interpolating polynomial that we learned about in Chapter 1. Instead, here we use the following "quick and dirty" method, based on inverting the Vandermonde matrix.

Let the unique polynomial of degree $\leq n$ that interpolates f at the distinct points $x_0, \ldots, x_n$ be denoted by $p_{n+1}(x)$, then we can express $p_{n+1}(x)$ in the monomial basis (as opposed to the Lagrange basis) as

$$p_{n+1}(x) = c_0 + c_1 x + \cdots + c_n x^n,$$

and since $p_{n+1}(x_j) = f(x_j)$ for $j = 0, \ldots, n$, we have that

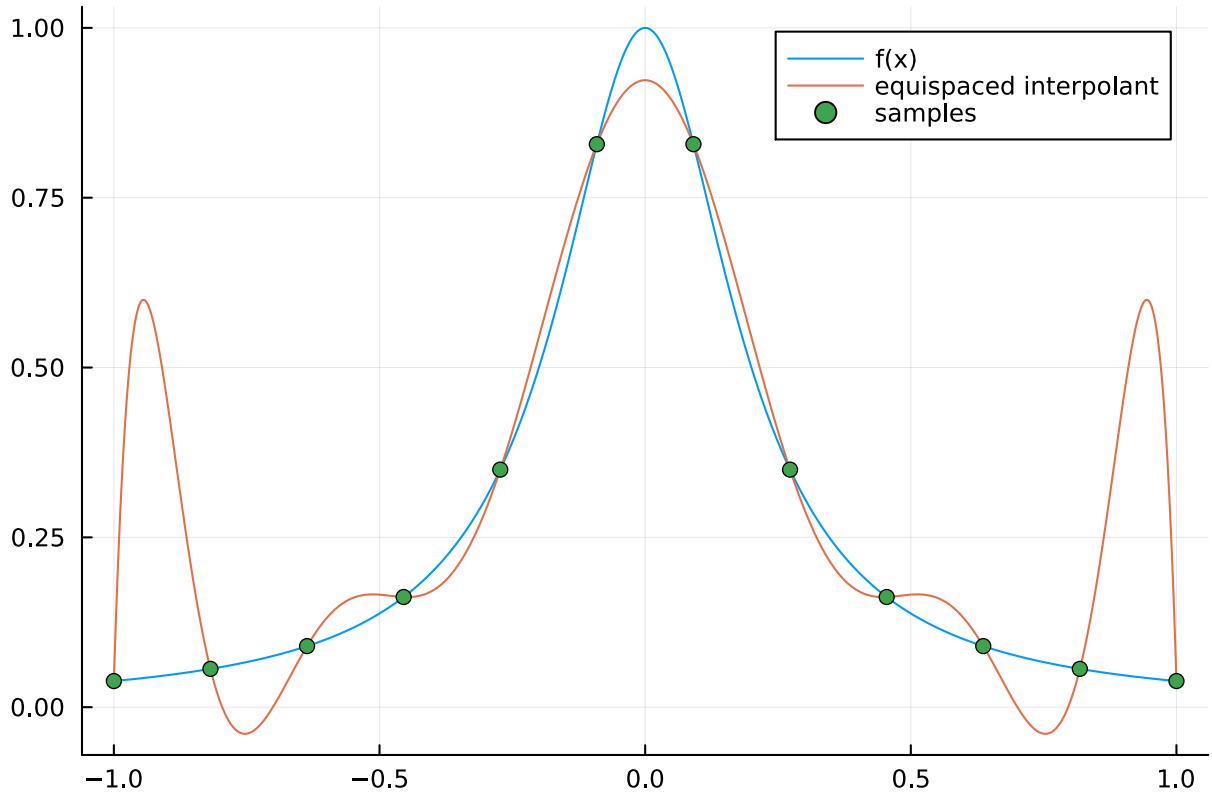
$$\underbrace{\begin{bmatrix} f(x_0) \\ \vdots \\ f(x_n) \end{bmatrix}}_{\mathbf{f}} = \underbrace{\begin{bmatrix} p_{n+1}(x_0) \\ \vdots \\ p_{n+1}(x_n) \end{bmatrix}}_{\mathbf{p}} = \begin{bmatrix} 1 & x_0 & \cdots & x_0^n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & x_n^n \end{bmatrix} \underbrace{\begin{bmatrix} c_0 \\ \vdots \\ c_n \end{bmatrix}}_{\mathbf{c}}$$

Definition (Vandermonde) The *Vandermonde matrix* associated with $n+1$ distinct points $x_0, \ldots, x_n \in \mathbb{R}$ is the matrix

$$V := \begin{bmatrix} 1 & x_0 & \cdots & x_0^n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & x_n^n \end{bmatrix}.$$

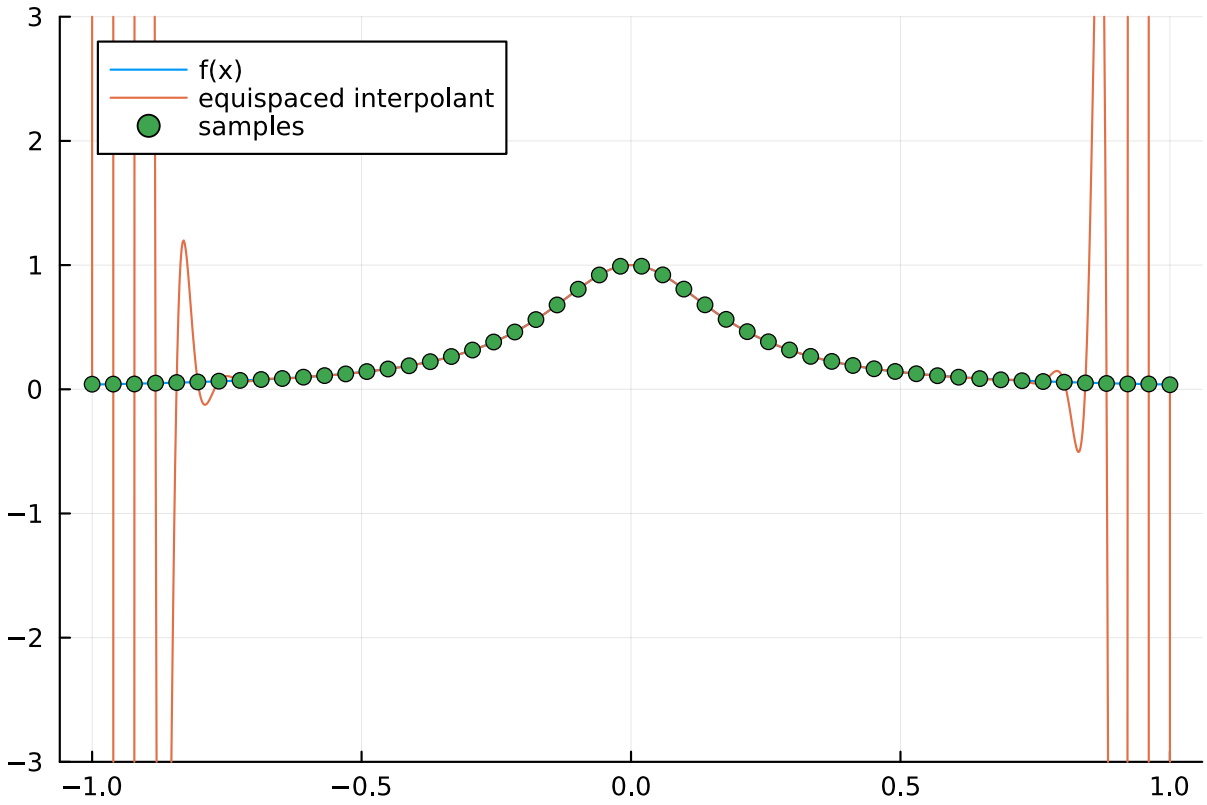
It can be shown that V is non-singular, hence we can compute the coefficients $c_0, \ldots, c_n$ by solving the linear system $V\mathbf{c} = \mathbf{f}$

```
using Plots, LinearAlgebra, ApproxFun, FFTW

f = x -> 1/(25x^2 + 1)

# Construct an interpolant through n + 1 equally spaced points on [-1, 1]
n = 11
x = range(-1,1;length=n+1)  # n+1 equispaced nodes
xx = range(-1,1;length=1001)  # plotting grid
V = x .^ (0:n)' # Vandermonde matrix
c = V \ f.(x) # coefficients of interpolatory polynomial
p = x -> dot(c, x .^ (0:n))  # interpolating polynomial
plot(xx,f.(xx);label="f(x)")
plot!(xx,p.(xx);label="equispaced interpolant")
scatter!(x,p.(x);label="samples")
```



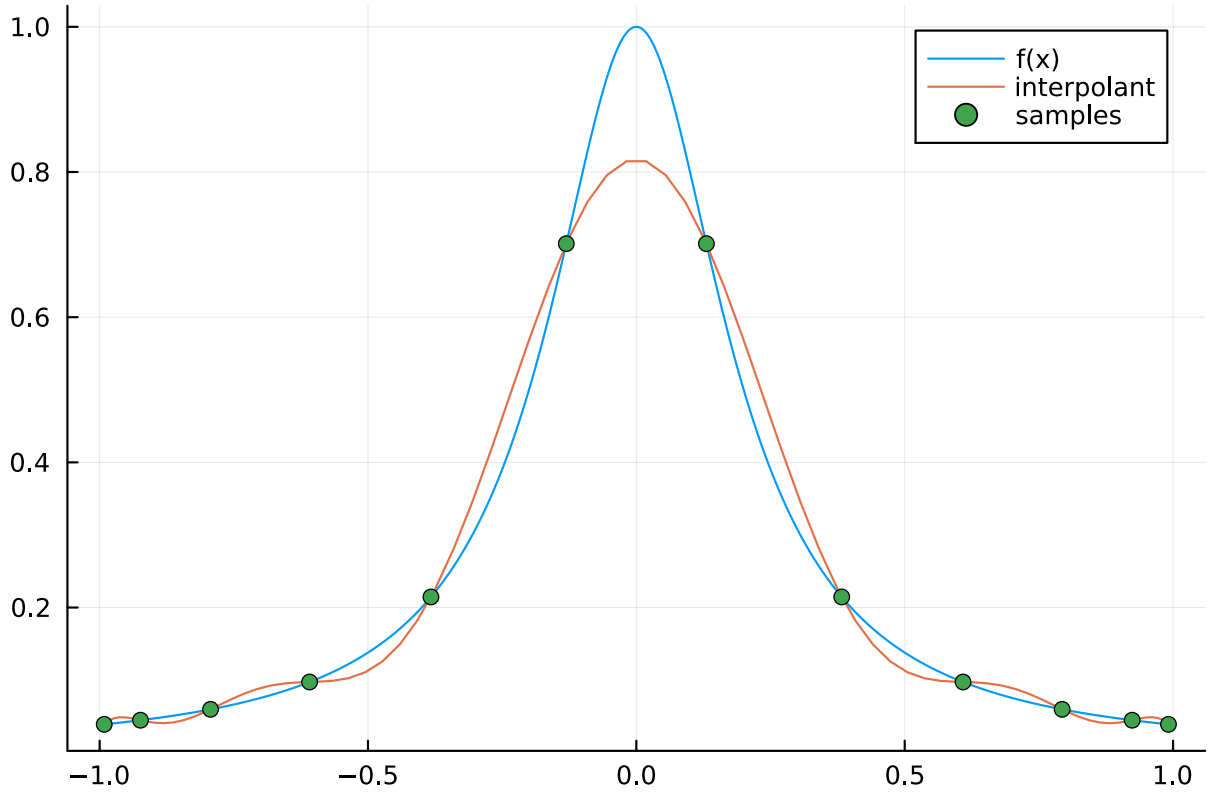
The oscillations at the ends of the interval become larger as we increase n:

```
n = 51
x = range(-1,1;length=n+1)  # n+1 equispaced nodes
V = x .^ (0:n)' # Vandermonde matrix
c = V \ f.(x) # coefficients of interpolatory polynomial
plot(xx,f.(xx);label="f(x)",ylims=(-3,3))
plot!(xx,p.(xx);label="equispaced interpolant")
scatter!(x,p.(x);label="samples")
```



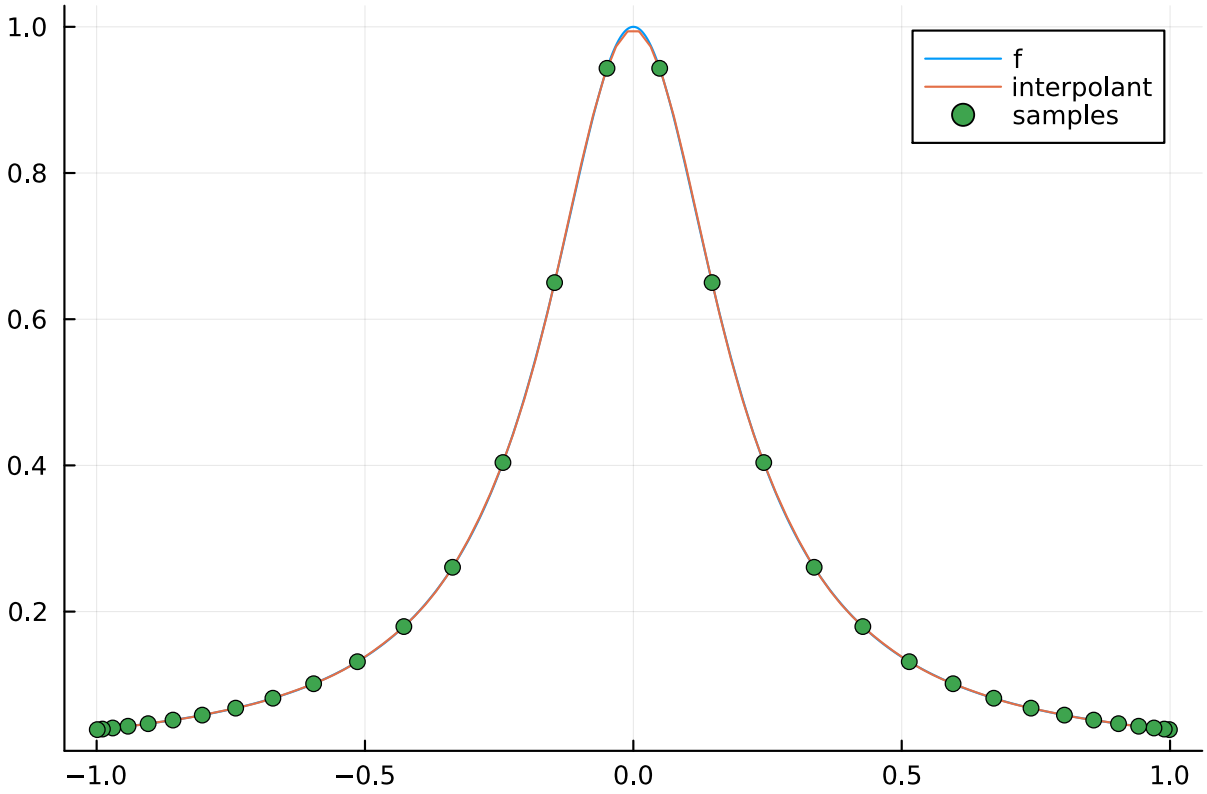
These oscillations of the interpolant through equally spaced nodes is known as the *Runge phenomenon* (not to be confused with the Gibbs phenomenon in Chapter 2). Equally spaced interpolation nodes is clearly not a good choice for high-degree polynomial interpolation. Maybe the oscillations at the endpoints can be suppressed if we choose nodes that are clustered at the ends of the interval?

```
# Construct an interpolant using ApproxFun
n = 11
S = Chebyshev()
p = Fun(f,S,n+1) # an interpolant through clustered nodes
x = points(S,n+1)  # n+1 unevenly spaced points
plot(xx,f.(xx);label="f(x)")
plot!(p;label="interpolant")
scatter!(x,p.(x);label="samples")
```



In the above figure, we interpolated $f(x)$ at the roots (zeros) of a Chebyshev polynomial, which we'll learn about later. The interpolants at the roots of the Chebyshev polynomials converge exponentially fast to f on $[-1, 1]$ as n (the number of interpolation nodes) increases.

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



Later we'll learn that interpolants at the roots of Chebyshev polynomials can be computed using the FFT ($\mathcal{O}(n \log n)$ complexity), which is much more efficient than inverting a Vandermonde matrix ($\mathcal{O}(n^3)$ complexity).

Remark An understanding of the reasons why equispaced interpolation failed for the above function and why interpolants through clustered points converged fast require tools from complex analysis and potential theory (See *Approximation Theory and Approximation Practice* by L.N. Trefethen).

1.2 An introduction to Chebyshev polynomials

Definition (Chebyshev polynomials of the first kind)

$$T_n(x) = \cos n \operatorname{acos} x, \qquad x \in [-1, 1],$$

or in other words,

$$T_n(\cos\theta) = \cos n\theta, \qquad \theta \in [0, \pi].$$

Proposition (Chebyshev three-term recurrence) The Chebyshev polynomials satisfy the following three-term recurrence:

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

Proof

It follows immediately from the definition that $T_0(x) = 1$, $T_1(x) = x$ and for $n \geq 1$

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

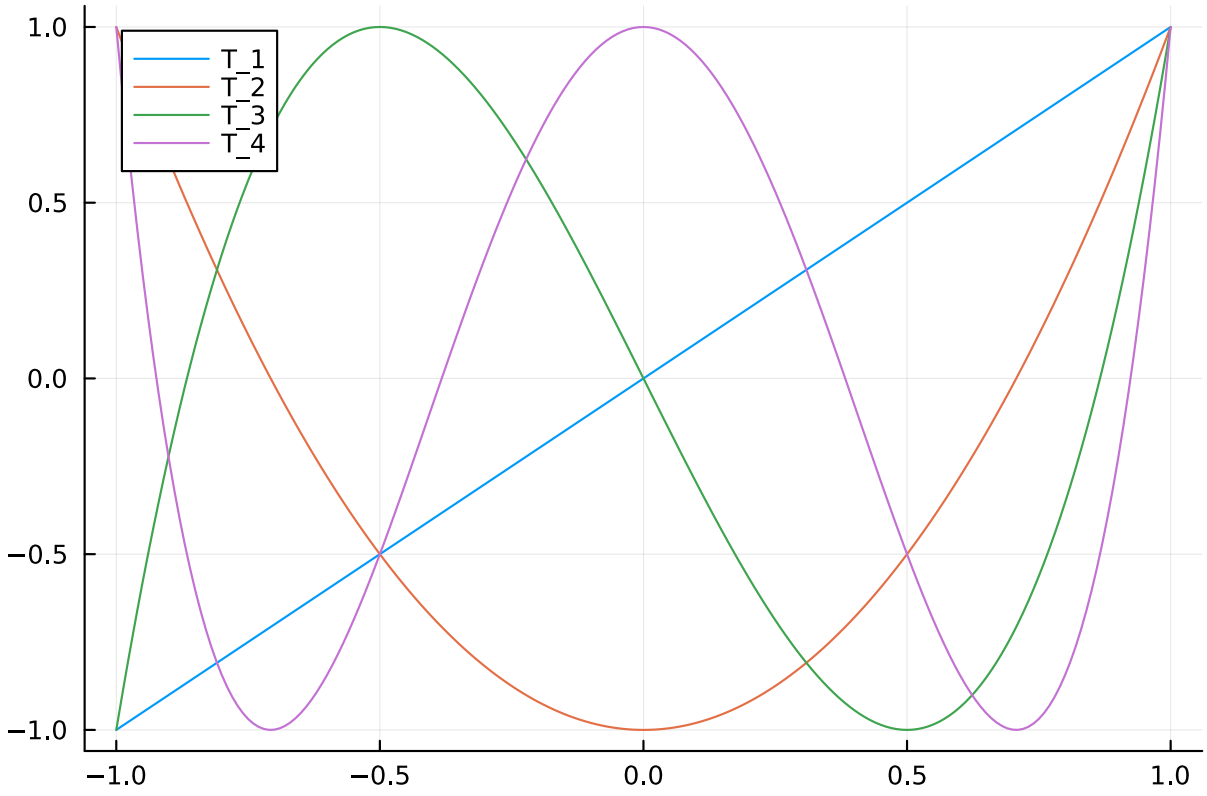
In other words $T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$ for $n \geq 1$.

■

Corollary $T_n(x)$ is a polynomial of degree precisely n.

Here are a few Chebyshev polynomials

```
xx = -1:0.01:1
p = plot()
for k = 1:4
    plot!(xx,Fun(Chebyshev(),[fill(0,k);1]).(xx);label="T_$k")
end
p
```



In the example above, we resolved the Runge phenomenon by interpolating the function at the roots of $T_n(x)$, which are at

$$x_j = \cos\left(\frac{(2j+1)\pi}{2n}\right), \qquad j = 0, \ldots, n-1,$$

since $T_n(x_j) = \cos n\,\mathrm{acos} x_j = \cos\left(\frac{(2j+1)\pi}{2}\right) = 0$. Later on, we shall be interested in the points

$$x_j = \cos\left(\frac{j\pi}{n}\right), \qquad j = 0, \ldots, n,$$

which are the extrema (maxima and minima) of $T_{n+2}(x)$. Notice that the roots of T_n are in $(-1, 1)$, whereas the extrema of T_{n+2} are in $[-1, 1]$.

1.3 The relation between Chebyshev and Fourier series

Suppose a function $f(x)$ has a convergent Chebyshev series for $x \in [-1, 1]$, i.e.,

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

Let's call the function that results from the change of variable $x = \cos\theta$, $g(\theta)$, i.e.,

$$g(\theta) = f(\cos\theta).$$

Notice that g is an even, 2π-periodic function of θ. We have that

$$\begin{aligned}
f(x) &= \sum_{k=0}^{\infty} b_k T_k(x) \\
&= \sum_{k=0}^{\infty} b_k \cos k\theta \qquad (x = \cos\theta) \\
&= g(\theta) \\
&= \sum_{k=0}^{\infty} b_k \left(\frac{\mathrm{e}^{\mathrm{i}k\theta} + \mathrm{e}^{-\mathrm{i}k\theta}}{2} \right) \\
&= \sum_{k=-\infty}^{\infty} c_k \mathrm{e}^{\mathrm{i}k\theta} \qquad (c_0 = b_0, c_k = c_{-k} = b_k/2, k \geq 1).
\end{aligned}$$

That is, a Chebyshev series is a Fourier series in disguise! We have just proved the following result.

Proposition (Chebyshev coefficients defined as Fourier coefficients) If $g(\theta) = f(\cos\theta)$ can be expressed as a Fourier series, then $f(x)$ has a Chebyshev series and the Chebyshev coefficients b_k, $k \geq 0$ of $f(x)$ are related to the Fourier coefficients c_k, $k \in \mathbb{Z}$ of $g(\theta)$ as follows,

$$b_0 = c_0, \qquad b_k = 2c_k = 2c_{-k}.$$

where

$$c_k = \frac{1}{2\pi} \int_0^{2\pi} g(\theta) \mathrm{e}^{-\mathrm{i}k\theta} \, \mathrm{d}\theta.$$

Remark We can prove that $c_{-k} = c_k$ for $g(\theta)$ by observing that

$$\begin{aligned}
c_k &= \frac{1}{2\pi} \int_0^{2\pi} g(\theta) \mathrm{e}^{-\mathrm{i}k\theta} \, \mathrm{d}\theta \\
&= \frac{(-1)^k}{2\pi} \int_{-\pi}^{\pi} g(\theta + \pi) \mathrm{e}^{-\mathrm{i}k\theta} \, \mathrm{d}\theta \\
&= \frac{(-1)^k}{2\pi} \int_{-\pi}^{\pi} g(\theta + \pi) \left(\cos k\theta - \mathrm{i} \sin k\theta \right) \, \mathrm{d}\theta \\
&= \frac{(-1)^k}{2\pi} \int_{-\pi}^{\pi} g(\theta + \pi) \cos k\theta \, \mathrm{d}\theta
\end{aligned}$$

where we used the fact that $\sin k\theta$ is an odd function, $\cos k\theta$ is an even function and $h(\theta) := g(\theta + \pi)$ is an even function of θ because

$$h(\theta) = g(\theta + \pi) = f(\cos(\theta + \pi)) = f(\cos(-\theta + \pi)) = g(-\theta + \pi) = h(-\theta).$$

Therefore if we replace k with $-k$, we get the same integral and hence $c_k = c_{-k}$.

1.4 The relation between Chebyshev and trigonometric interpolants

From Chapter 2, we know that the truncated Fourier series

$$q_{2n+1}(\theta) = \sum_{k=-n}^{n} \tilde{c}_k^{2n+1} \mathrm{e}^{\mathrm{i}k\theta},$$

where

$$\tilde{c}_k^{2n+1} = \frac{1}{2n+1} \sum_{j=0}^{2n} g(\theta_j) \mathrm{e}^{-\mathrm{i}k\theta_j}, \qquad k = -n, \ldots, n, \qquad \theta_j = \frac{2\pi j}{2n+1}, \qquad j = 0, \ldots, 2n,$$

interpolates $g(\theta)$ at $\theta = \theta_j$, $j = 0, \ldots, 2n$. Using parity arguments (as we did when we proved that $c_k = c_{-k}$ for $g(\theta)$), we can show that $\tilde{c}_k^{2n+1} = \tilde{c}_{-k}^{2n+1}$. Therefore,

$$\begin{aligned} q_{2n+1}(\theta) &= \tilde{c}_0^{2n+1} + \sum_{k=1}^{n} \tilde{c}_k^{2n+1} \left(\mathrm{e}^{\mathrm{i}k\theta} + \mathrm{e}^{-\mathrm{i}k\theta} \right) \\ &= \tilde{c}_0^{2n+1} + 2 \sum_{k=1}^{n} \tilde{c}_k^{2n+1} \cos k\theta \\ &= \tilde{c}_0^{2n+1} + 2 \sum_{k=1}^{n} \tilde{c}_k^{2n+1} T_k(x) \qquad (x = \cos\theta) \\ &:= p_{n+1}(x) \end{aligned}$$

from which we get the following result.

Proposition (Chebyshev polynomial interpolant constructed from a trigonometric interpolant) The polynomial of degree $\leq n$ defined by

$$p_{n+1}(x) = \sum_{k=0}^{n} \tilde{b}_k^n T_k(x)$$

where

$$\tilde{b}_0^n = \tilde{c}_0^{2n+1}, \qquad \tilde{b}_k^n = 2\tilde{c}_k^{2n+1} = 2\tilde{c}_{-k}^{2n+1}$$

interpolates $f(x)$ at x_j, where

$$x_j = \cos\left(\frac{2\pi j}{2n+1}\right), \qquad j = 0, \ldots, n$$

Proof Since

$$q_{2n+1}(\theta_j) = g(\theta_j), \qquad \theta_j = \frac{2\pi j}{2n+1}, \qquad j = 0, \ldots, 2n$$

and $q_{2n}(\theta) = p_{2n+1}(\cos\theta) = p_{2n+1}(x)$ where $x = \cos\theta$, it follows that

$$p_{2n+1}(x_j) = p_{2n+1}(\cos\theta_j) = q_{2n+1}(\theta_j) = g(\theta_j) = f(\cos\theta_j) = f(x_j), \qquad j = 0, \ldots, 2n.$$

The result follows by noting that only $n+1$ of the $2n+1$ points x_j, $j = 0, \ldots, 2n$ are distinct because

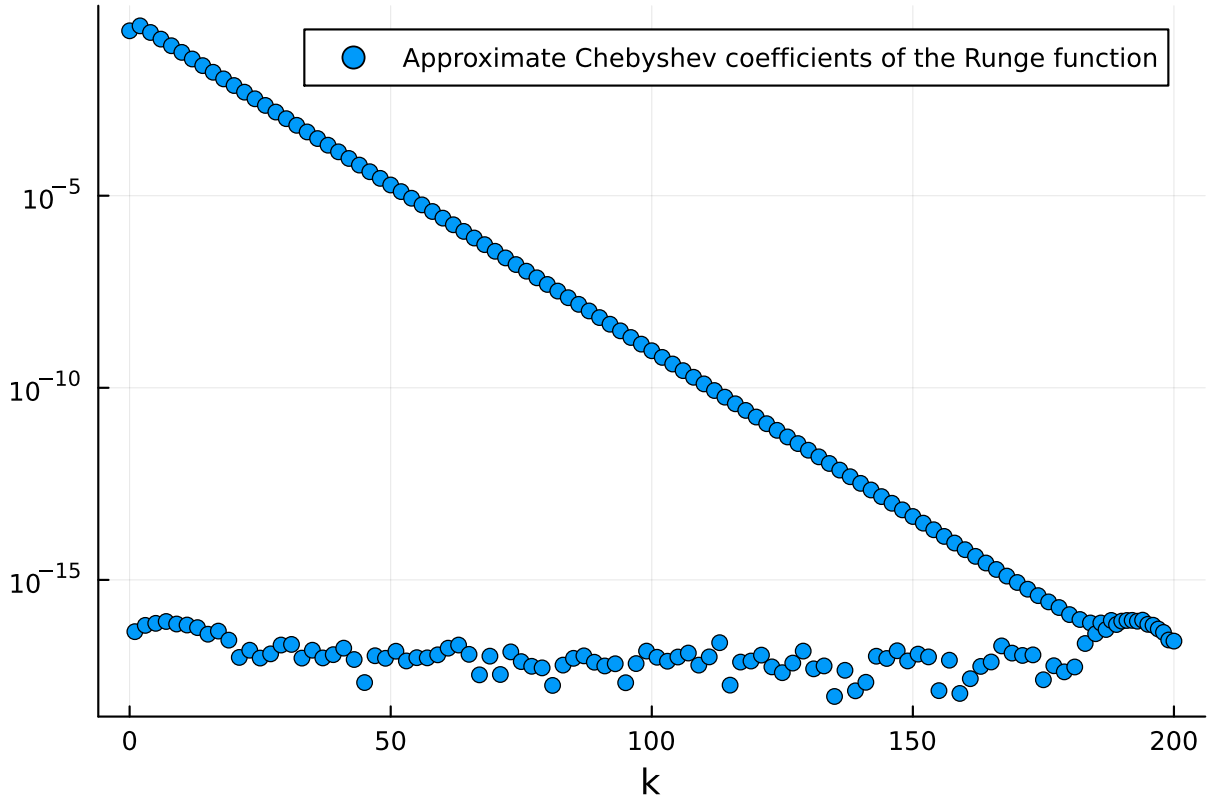
$$x_n = \cos\left(\frac{2\pi n}{2n+1}\right) = \cos\left(\pi - \frac{\pi}{2n+1}\right) = \cos\left(\pi + \frac{\pi}{2n+1}\right) = x_{n+1}$$

and similarly $x_{n-1} = x_{n+2}$, …, $x_1 = x_{2n}$. ■

We refer to $p_{n+1}(x) = \sum_{k=0}^{n} \tilde{b}_k^n T_k(x)$ as a Chebyshev interpolant of $f(x)$ and the $\tilde{b}_k^n$ are the approximate Chebyshev coefficients of the the function f we are approximating.

Recall that the approximate Fourier coefficients $\tilde{c}_k^{2n+1}$ can be computed with the FFT, hence we can compute the coefficients $\tilde{b}_k^n$ of a Chebyshev interpolant with the FFT. We can use the (inverse) FFT to map the coefficients $\tilde{b}_k^n$ to the function values $f(x_j)$. Here's an example:

```
# Use the FFT to compute the approximate Chebyshev coefficients of the Runge function
g = θ -> f(cos(θ))
n = 200
θ = range(0,2π;length= 2n+2)[1:end-1] # the points θ_j
ct = fft(g.(θ))/(2n+1) # compute approximate Fourier coefficients of g
# approximate Chebyshev coefficients
bt = [ct[1];2ct[2:n+1]]
scatter(0:n,abs.(bt);
xlabel="k", yscale=:log10,label="Approximate Chebyshev coefficients of the Runge
function")
```



The approximate Chebyshev coefficients decay exponentially fast with k because $f(x)$, the Runge function, is analytic on $[-1, 1]$.

```
# Map approximate Chebyshev coefficients to function values
norm((2n+1)*ifft(ct) - f.(cos.(θ)),Inf)
```

```
3.544325313357017e-16
```

In ApproxFun.jl, the command `Fun(f)` approximates a function using a Chebyshev interpolant (in Chebfun, the command `chebfun` does the same). The number of Chebyshev coefficients required the approximate the function to roughly 10^{-16} relative accuracy is determined automatically. Here's an example:

```
fc = Fun(f)
```

```
Fun(Chebyshev(),[0.19611613513818404, 0.0, -0.26361085189847744, 0.0, 0.177
16716982434333, 0.0, -0.11907023492210417, 0.0, 0.0800245376074016, 0.0  ...
 0.0, 1.2368591486255537e-16, 0.0, -7.257531816796535e-17, 0.0, 5.855258184
7052e-17, 0.0, -3.05491432152328e-17, 0.0, 2.7525712049567778e-17])
```

```
length(fc.coefficients)
```

```
189
```

Let's check that the approximate Chebyshev coefficients we computed with the FFT are approximate equal to the Chebyshev coefficients computed in with ApproxFun:

```
@show bt[1:189] ≈ fc.coefficients
norm(bt[1:189] - fc.coefficients,Inf)
```

```
bt[1:189] ≈ fc.coefficients = true
1.3121635288093246e-16
```

1.5 Differentiating Chebyshev interpolants with the FFT

We can approximate derivatives of $f(x)$ by differentiating $q_n(\theta)$ in Fourier coefficient space via the FFT, which is very efficient, as we learnt in Chapter 2. The first derivatives of $p_{n+1}(x)$ and $q_{2n+1}(\theta)$ are related as follows:

$$f'(x) \approx \frac{\mathrm{d}}{\mathrm{d}x} p_{n+1}(x) = \frac{\mathrm{d}}{\mathrm{d}x} q_{2n+1}(\theta) = q'_{2n+1}(\theta)\frac{\mathrm{d}\theta}{\mathrm{d}x} = -\frac{q'_{2n+1}(\theta)}{\sin\theta}$$

```
df = x -> -50x/(1 + 25x^2)^2 # f', the derivative of the Runge function
# differentiate q in coeffcient space and map back to function values space
dqn = (2n + 1)*ifft( im*ifftshift(-n:n).* ct)
dpn = -dqn ./sin.(θ)  # p_n', derivative of p_n
x = cos.(θ)
# the error of the approximation f' ≈ p'
norm(df.(x[2:end]) - dpn[2:end],Inf)
```

```
1.340491672910366e-12
```

Note that $p'_{n+1}(x) = -q'_{2n+1}(\theta)/\sin\theta$ is not well-defined for $\theta = 0, \pi$ (i.e., at $x = \cos\theta = \pm 1$). For these cases we use l'Hopital's rule.

We typically prefer to approximate a function (or solution to a differential equation) at the *Chebyshev points*,

$$x_j = \cos\left(\frac{j\pi}{n}\right), \qquad j = 0, \ldots, n,$$

because these include the end points of the interval $[-1, 1]$ where we typically impose boundary conditions.

Here is a function that takes as input the function values $f(x_j)$, $j = 0, \ldots, n$ at the Chebyshev points and it returns $p'_{n+1}(x_j)$, $j = 0, \ldots, n$ via the FFT (i.e., the derivate of the Chebyshev interpolant at the Chebyshev points, which approximate $f'(x_j)$). The formulae for performing differentiation at the Chebyshev points via the FFT are slightly different from the formulae discussed above, where we considered a different set of points, namely $x_j = \cos\theta_j = \cos\left(\frac{2\pi j}{2n+1}\right), j = 0, \ldots, 2n$

```
function chebfft(f)
    n = length(f)-1
    x = cos.((0:n)*π/n) # Chebyshev points
    ii = 0:n-1
    q = [f; f[n:-1:2]]       # transform x -> θ
    # differentiate the interpolant q_n in coefficient space and map back to function
values
    cq = real.(fft(q))
    dq = real.(ifft(im*[ii; 0; 1-n:-1] .*cq))
    df = zeros(n+1,1)
    # Compute approximations to f' at the interior points
    df[2:n] = -dq[2:n] ./sqrt.(1 .- x[2:n].^2)     # transform θ -> x
    # At the boundary points
    df[1] = sum(ii.^2 .*cq[1:n])/n + .5*n*cq[n+1]
    df[n+1] = sum((-1) .^(ii .+1) .* ii.^2 .*cq[1:n])/n +
              .5*(-1)^(n+1)*n*cq[n+1]
    df
end
```

```
chebfft (generic function with 1 method)
```

Let's test `chebfft`:

```
x = cos.((0:n)*π/n)
norm(chebfft(f.(x)) - df.(x),Inf)
```

```
1.562694418311139e-12
```

1.6 Differentiation matrices for Chebyshev interpolants

In Chapter 2 we learnt about two equivalent representations of the trigonometric interpolant ("coefficient space" and "value space"), each expressed in terms of different basis functions. Here we do the same for polynomial interpolants in the Chebyshev points. We have already seen one representation as an expansion in Chebyshev polynomials and approximate Chebyshev coefficients,

$$p_{n+1}(x) = \sum_{k=0}^{n} \tilde{b}_k^n T_k(x),$$

which allowed us to leverage Fourier series to compute $\tilde{b}_k^n$ and derivatives of $p_{n+1}(x)$ via the FFT. We now build differentiation matrices that act on function values to compute derivatives of $p_{n+1}(x)$ by using its representation in terms of Lagrange basis polynomials and function values.

Recall from the formula for the Lagrange interpolating polynomial that

$$p_{n+1}(x) = \sum_{k=0}^{n} \ell_k(x) f(x_k), \qquad \ell_k(x) = \prod_{\substack{j=0 \\ j\neq k}}^{n} \frac{x - x_j}{x_k - x_j}$$

here we let x_k be the Chebyshev points

$$x_k = \cos\left(\frac{\pi k}{n}\right), \qquad k = 0, \ldots, n.$$

We can represent differentiation of the Chebyshev interpolant via an $(n+1) \times (n+1)$ differentiation matrix D_n: since

$$p'_{n+1}(x) = \sum_{k=0}^{n} \ell'_k(x) f(x_k),$$

we have

$$\begin{bmatrix} p'_{n+1}(x_0) \\ \vdots \\ p'_{n+1}(x_n) \end{bmatrix} = \begin{bmatrix} \ell'_0(x_0) & \cdots & \ell'_n(x_0) \\ \vdots & \vdots & \vdots \\ \ell'_0(x_n) & \cdots & \ell'_n(x_n) \end{bmatrix} \begin{bmatrix} f(x_0) \\ \vdots \\ f(x_n) \end{bmatrix}$$

or in vector form,

$$\mathbf{p}' = D_n \mathbf{f}.$$

Here is a function that builds the differentiation matrix D_n:

```
function cheb(n)

    x = cos.(π*(0:n)/n)
    c = [2; ones(n-1,1); 2].*(-1).^(0:n)
    X = repeat(x,1,n+1)
    dX = X-X'
    D = (c*(1 ./c)')./( dX + I ) # off-diagonal entries
    D = D - diagm(vec(sum(D';dims=1)))# diagonal entries

    D, x

end
```

```
cheb (generic function with 1 method)
```

To summarise, we have two equivalent methods for computing the derivate of the Chebyshev interpolant at the $n+1$ Chebyshev points. The FFT method has $\mathcal{O}(n \log n)$ complexity while applying a differentiation matrix to a vector requires $\mathcal{O}(n^2)$ operations because D_n is a dense matrix.

Let's check that the two methods give the same output (up to rounding errors):

```
n = 50
D,x = cheb(n)
@show D*f.(x) ≈ chebfft(f.(x))
norm(D*f.(x) - chebfft(f.(x)),Inf)
```

```
D * f.(x) ≈ chebfft(f.(x)) = true
5.88418203051333e-14
```

1.7 A PDE example: the wave equation

Consider the following $(1+1)$-dimensional wave equation PDE initial / boundary value problem

$$u_{tt} = u_{xx}, \qquad t > 0, \qquad -1 < x < 1, \qquad u(\pm 1, t) = 0,$$

subject to the initial data

$$u(x, 0) = f(x), \qquad u_t(x, 0) = g(x), \qquad x \in [-1, 1],$$

where we assume that f and g are not both identically zero (because then the trivial solution $u(x,t) = 0$ solves the problem). We are going to approximate the solution to this problem by approximating u_{tt} with finite differences and u_{xx} with differentiation of a Chebyshev interpolant in x. First, we derive the exact solution to this problem using the method of separation of variables.

1.7.1 The method of separation of variables

We attempt to find a solution of the form

$$u(x, t) = X(x)T(t).$$

From the zero boundary conditions, we conclude that

$$u(-1,t) = X(-1)T(t) = 0 = T(t)X(1) = u(1,t) \quad \Rightarrow \quad X(-1) = 0 = X(1).$$

Substituting $u = XT$ into the PDE, we find that

$$\frac{T''(t)}{T(t)} = \frac{X''(x)}{X(x)}.$$

This can hold only if

$$\frac{T''(t)}{T(t)} = \frac{X''(x)}{X(x)} = \lambda \in \mathbb{R}.$$

We need to consider three cases

1. $$\lambda > 0$$
2. $$\lambda = 0$$
3. $$\lambda < 0$$

Cases (1) and (2) are left as an exercise.

For case (3), we set $\lambda = -\kappa^2$ with $\kappa \in \mathbb{R} \setminus \{0\}$ then for X we have the ODE boundary-value (or eigenvalue) problem

$$X'' = -\kappa^2 X, \qquad X(-1) = 0 = X(1)$$

The general solution to the ODE $X'' + \kappa^2 X = 0$ is $X = c_1 \sin \kappa x + c_2 \cos \kappa x$, for arbitrary constants c_1, c_2 ($\sin \kappa x$ and $\cos \kappa x$ are eigenfunctions of the differential operator $\frac{\mathrm{d}^2}{\mathrm{d}x^2}$). To satisfy the boundary conditions $X(-1) = 0 = X(1)$, we require

$$-c_1 \sin \kappa + c_2 \cos \kappa = 0 = c_1 \sin \kappa + c_2 \cos \kappa.$$

Since $\kappa \neq 0$ and $\sin \kappa$ and $\cos \kappa$ are linearly independent functions, these equations hold only if $c_1 = c_2 = 0$, which implies that $X = 0$ which would give the inadmissable solution $u = 0$. Instead, we attempt to find solutions of the form (i) $X = c_1 \sin \kappa x$ and (ii) $X = c_1 \cos \kappa x$. For case (i), we require

$$\kappa = \kappa_n = \pi n, \qquad n \neq 0,$$

in order to satisfy $X(-1) = 0 = X(1)$ and for case (ii),

$$\kappa = \kappa_n = (2n+1)\frac{\pi}{2}$$

where $n \in \mathbb{Z}$. For case (i), T has the solution

$$T(t) = c_1 \sin \kappa_n t + c_2 \cos \kappa_n t \qquad (1)$$

with $\kappa_n = \pi n$ and in case (ii), T has the solution (1) with $\kappa_n = (2n+1)\frac{\pi}{2}$. As before, c_1 and c_2 are arbitrary constants. We have therefore constructed two families of solutions to the wave equation:

$$u = (c_1 \sin \kappa_n t + c_2 \cos \kappa_n t) \sin \kappa_n x, \qquad \kappa_n = \pi n$$

and

$$u = (b_1 \sin \kappa_n t + b_2 \cos \kappa_n t) \cos \kappa_n x, \qquad \kappa_n = (2n+1)\frac{\pi}{2}.$$

Since the wave equation is linear, any scalar multiples and linear combinations of the solutions above are also solutions to the PDE initial / boundary-value problem.

Therfore the following series is a formal solution to the PDE,

$$u(x,t) = \sum_{n=1}^{\infty} (\alpha_n \sin \pi n t + \beta_n \cos \pi n t) \sin \pi n x + \sum_{n=0}^{\infty} (\gamma_n \sin \kappa_n t + \delta_n \cos \kappa_n t) \cos \kappa_n x,$$

where $\kappa_n = (2n+1)\frac{\pi}{2}$ and $\alpha_n, \beta_n, \gamma_n, \delta_n$ are arbitrary constants whose values we shall determine by using the initial data:

$$u(x,0) = \sum_{n=1}^{\infty} \beta_n \sin \pi n x + \sum_{n=0}^{\infty} \delta_n \cos \kappa_n x = f(x)$$

$$u_t(x,0) = \sum_{n=1}^{\infty} \pi n \alpha_n \sin \pi n x + \sum_{n=0}^{\infty} \kappa_n \gamma_n \cos \kappa_n x = g(x)$$

Using the fact that

$$\int_{-1}^{1} \sin \pi n x \sin \pi m x \, \mathrm{d}x = \delta_{n,m} = \begin{cases} 1 & \text{if } n = m \\ 0 & \text{otherwise} \end{cases}$$

$$\int_{-1}^{1} \sin \pi n x \cos \kappa_m x \, \mathrm{d}x = 0, \qquad n, m \geq 0$$

and

$$\int_{-1}^{1} \cos \kappa_n x \cos \kappa_m x \, \mathrm{d}x = \delta_{n,m}$$

it follows that

$$\beta_m = \int_{-1}^{1} f(x) \sin m\pi x \, \mathrm{d}x, \quad m \geq 1, \qquad \delta_m = \int_{-1}^{1} f(x) \cos \kappa_m x \, \mathrm{d}x, \quad m \geq 0$$

and

$$\alpha_m = \frac{1}{\pi n} \int_{-1}^{1} g(x) \sin m\pi x \, \mathrm{d}x, \quad m \geq 1, \qquad \gamma_m = \frac{1}{\kappa_n} \int_{-1}^{1} g(x) \cos \kappa_m x \, \mathrm{d}x, \quad m \geq 0.$$

We have derived the exact solution to the PDE intial/boundary value problem as the sum of two series: a sine series (in x) and a cosine series (in x), each with time-dependent coefficients that are themselves expressed in terms of trigonometric functions. We can express the sine and cosine series as Fourier series with time-dependent coefficients. To compute an approximation to the solution, we could use the FFT to compute approximate coefficients $\tilde{\alpha}_n$, $\tilde{\beta}_n$, $\tilde{\gamma}_n$, $\tilde{\delta}_n$ and then set

$$u(x,t) \approx \sum_{n=1}^{N} \left(\tilde{\alpha}_n \sin \pi n t + \tilde{\beta}_n \cos \pi n t\right) \sin \pi n x + \sum_{n=0}^{N} \left(\tilde{\gamma}_n \sin \kappa_n t + \tilde{\delta}_n \cos \kappa_n t\right) \cos \kappa_n x.$$

This method relies on knowing the exact solution to the problem, which may not be possible for a more complicated PDE. Instead, we shall compute an approximate solution using a combination of finite differences and Chebyshev interpolants, which is a method which doesn't require knowing the exact solution.

Remark We derived an exact solution to the initial / boundary-value PDE problem above. If we consider the "pure" initial-value problem (the Cauchy problem) for the wave equation on the whole real line, i.e.,

$$u_{tt} = u_{xx}, \qquad t > 0, \qquad x \in \mathbb{R},$$

note there are no boundary conditions, then it is easy to verify that

$$u(x,t) = F(x-t) + G(x+t)$$

is a solution to the Cauchy problem, where F and G are unknown functions that can be determined from the initial data

$$u(x,0) = f(x), \qquad u_t(x,0) = g(x).$$

This solution can be derived using the method of characteristics.

1.7.2 Numerical solution

We consider the initial / boundary-value problem for the wave equation

$$u_{tt} = u_{xx}, \qquad t > 0, \qquad -1 < x < 1, \qquad u(\pm 1, t) = 0.$$

We discretise x using the Chebyshev points,

$$x_j = \cos\left(\frac{j\pi}{n_x}\right), \qquad j = 0, \ldots, n_x,$$

and we let $t \in [0, T]$ and set

$$t_i = i\tau, \qquad \tau = \frac{T}{n_t}, \qquad i = 0, \ldots, n_t.$$

We let u_j^i denote the approximation to the exact solution u at $x = x_j$ and $t = t_i$, i.e.,

$$u_j^i \approx u(x_j, t_i),$$

and we let $\mathbf{u}^i$ denote the vector of approximations at "time-level" i:

$$\mathbf{u}^i = \begin{bmatrix} u_0^i \\ u_1^i \\ \vdots \\ u_n^i \end{bmatrix}.$$

The second time-derivative is approximated with a central difference approximation (see Chapter 1, Exercise 1):

$$u_{tt}(x_j, t_i) \approx \frac{u(x_j, t_{i+1}) - 2u(x_j, t_i) + u(x_j, t_{i-1})}{\tau^2} \approx \frac{u_j^{i+1} - 2u_j^i + u_j^{i-1}}{\tau^2}, \qquad j = 0, \ldots, n_x$$

or, in vector notation,

$$u_{tt}(\mathbf{x}, t_i) \approx \frac{\mathbf{u}^{i+1} - 2\mathbf{u}^i + \mathbf{u}^{i-1}}{\tau^2}.$$

The spatial derivative is approximated by differentiation of a Chebyshev interpolant in x:

$$\begin{bmatrix} u_x(x_0, t_i) \\ \vdots \\ u_x(x_n, t_i) \end{bmatrix} \approx \begin{bmatrix} \ell_0'(x_0) & \cdots & \ell_n'(x_0) \\ \vdots & \vdots & \vdots \\ \ell_0'(x_n) & \cdots & \ell_n'(x_n) \end{bmatrix} \begin{bmatrix} u(x_0, t_i) \\ \vdots \\ u(x_n, t_i) \end{bmatrix} \approx \begin{bmatrix} \ell_0'(x_0) & \cdots & \ell_n'(x_0) \\ \vdots & \vdots & \vdots \\ \ell_0'(x_n) & \cdots & \ell_n'(x_n) \end{bmatrix} \begin{bmatrix} u_0^i \\ \vdots \\ u_n^i \end{bmatrix}$$

i.e.,

$$u_x(\mathbf{x}, t_i) \approx D_n \mathbf{u}^i$$

and

$$u_{xx}(\mathbf{x}, t_i) \approx D_n \left(D_n \mathbf{u}^i \right) = D_n^2 \mathbf{u}^i.$$

Therefore we approximate the solution to the PDE as follows:

$$\mathbf{u}^{i+1} = 2\mathbf{u}^i - \mathbf{u}^{i-1} + \tau^2 D_n^2 \mathbf{u}^i, \qquad i = 0, \ldots, n_t - 1.$$

Remark Note that we don't need to include the approximations u_0^i and u_n^i since we know the solution values at $x = \pm 1$ from the zero boundary conditions $u(\pm 1, t) = 0$. For now we ignore this fact since we shall discuss the imposition of boundary conditions later in this module.

Remark In practice, we don't compute $D_n^2 \mathbf{u}^i$ by building the differentiation matrix D_n and squaring it, since it costs $\mathcal{O}(n_x^3)$ operations, however computing $D_n(D_n \mathbf{u}^i)$ has $\mathcal{O}(n_x^2)$ complexity. As we showed above, $D_n \mathbf{u}^i$ can be computed with the FFT with $\mathcal{O}(n_x \log n_x)$ complexity, so we compute $D_n^2 \mathbf{u}^i$ by differentiating twice via the FFT, which thus also has $\mathcal{O}(n_x \log n_x)$ complexity.

We require two starting values to implement the numerical method. We let

$$u(x,0) = f(x) = \mathrm{e}^{-200x^2}.$$

Rather than specifying $u_t(x,0)$, we impose the condition

$$u(x,-\tau) = f(x-\tau) = \mathrm{e}^{-200(x-\tau)^2}.$$

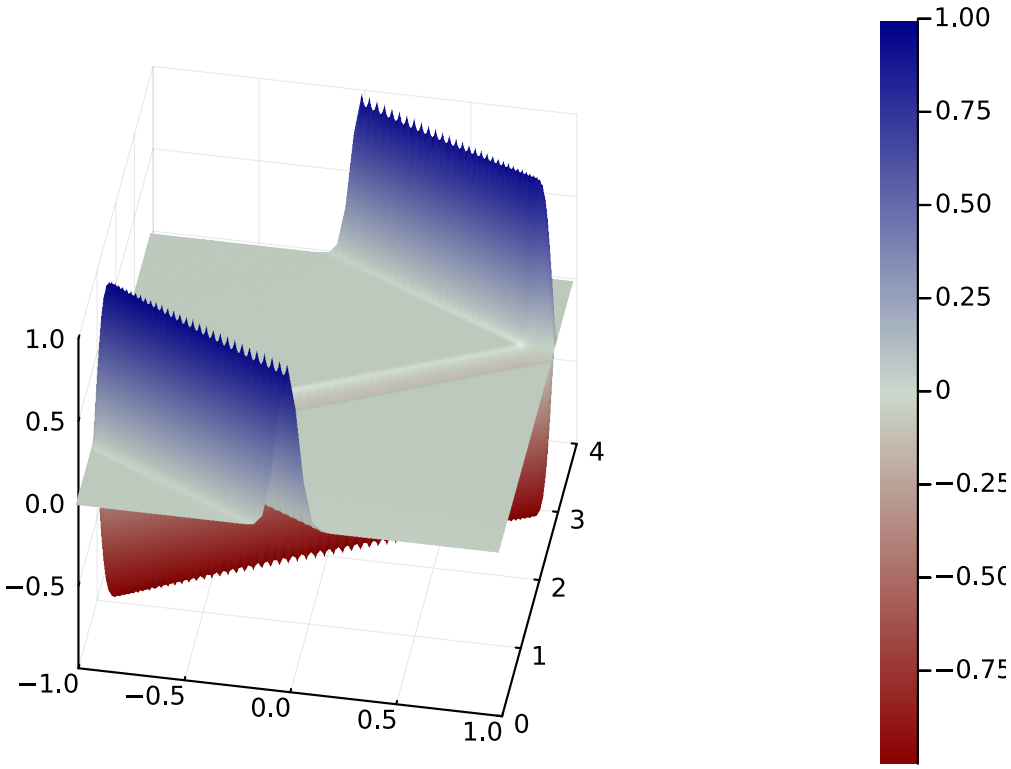
(Recall that $u(x,t) = F(x+t)$ is a solution to the Cauchy problem for the wave equation). Hence, the numerical method is

$$\mathbf{u}^0 = f(\mathbf{x}), \qquad \mathbf{u}^{-1} = f(\mathbf{x}-\tau), \qquad \mathbf{u}^{i+1} = 2\mathbf{u}^i - \mathbf{u}^{i-1} + \tau^2 D_n^2 \mathbf{u}^i, \qquad i = 0, \ldots, n_t - 1.$$

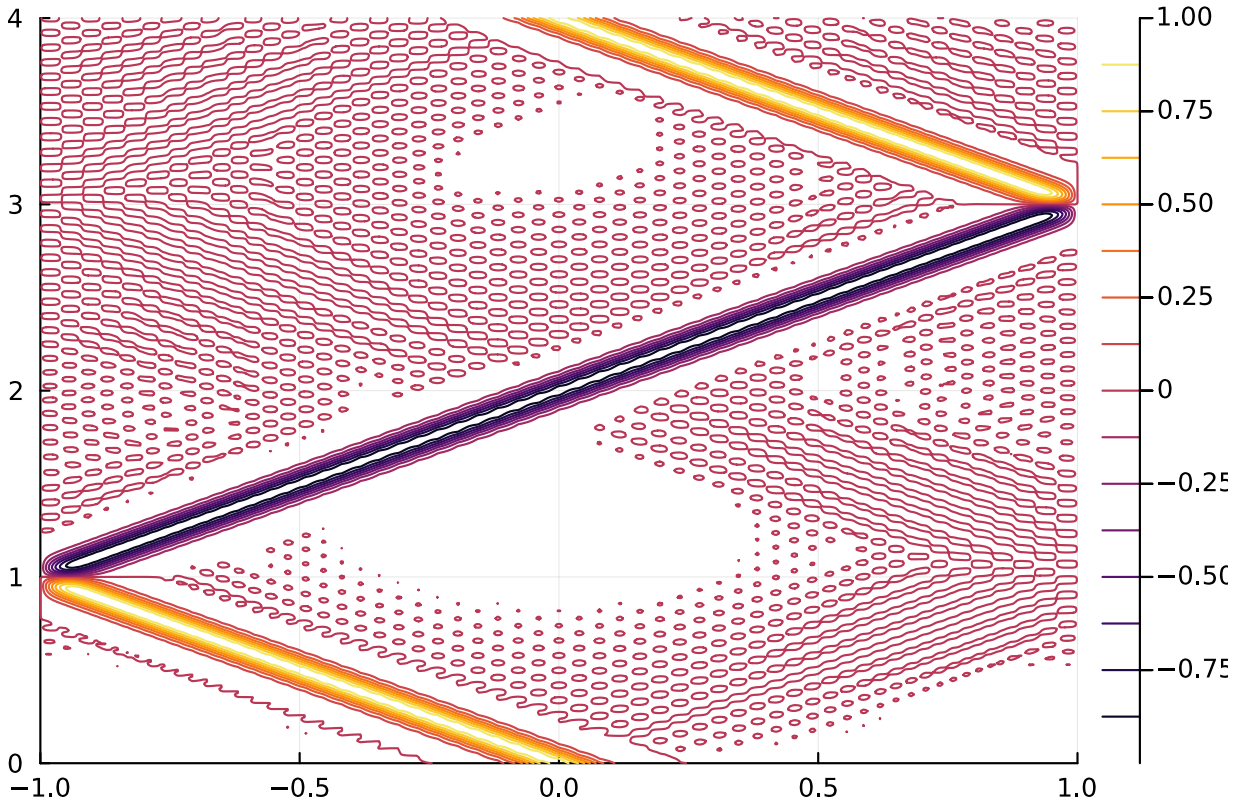
```
N = 80; x = cos.(π*(0:N)/N); dt = 8/N^2;
v = exp.(-200*x.^2);vold = exp.(-200*(x .- dt).^2)
tmax = 4; tplot = 0.01;
plotgap = Int64(round(tplot/dt)); dt = tplot/plotgap;
nplots = @show Int64(round(tmax/tplot))
plotdata = [transpose(v); zeros(nplots,N+1)]; tdata = [0.0];
@time begin
for i = 1:nplots
    for n = 1:plotgap
        global v
        global vold
        w = chebfft(chebfft(v)); w[1]=0; w[N+1] = 0;
        vnew = 2*v - vold + dt^2*w; vold = v; v = vnew;
    end
        global tdata
    plotdata[i+1,:] = v; tdata = [tdata;dt*i*plotgap]
end
end
```

```
Int64(round(tmax / tplot)) = 400
  3.106897 seconds (2.98 M allocations: 315.532 MiB, 3.81% gc time, 55.60%
compilation time)
```

```
surface(x,tdata,plotdata,seriescolor=:redsblues, camera=(10,40))
```




```
contour(x[end:-1:1],tdata,plotdata[:,end:-1:1])
```



1.8 Rate of convergence of Chebyshev interpolants and their derivatives

Since Chebyshev interpolants of a generally non-periodic function $f(x)$ on $[-1,1]$ are trigonometric interpolants of the periodic function $g(\theta) = f(\cos\theta)$, we can deduce the rate of convergence of Chebyshev interpolating polynomials from the rate of convergence of trigonometric interpolants of $g(\theta)$.

Examples

1. For $f(x) = |x|$, $x \in [-1,1]$, it follows that $g(\theta) = |\cos\theta|$, $\theta \in [0,\pi]$. To apply the results we stated for periodic functions in Chapter 2, we need to consider g on the interval $[0,2\pi]$. Since $g(\theta) \in C^0[0,2\pi]$, $\|g^{(k)}\|_1 < \infty$ for $k = 0,1,2$ (verify) and $g^{(1)}(0), g^{(1)}(2\pi) < \infty$, it follows that

$$\|q_{2n+1}(\theta) - g(\theta)\|_\infty = \|p_{n+1}(x) - f(x)\|_\infty = \mathcal{O}\left(n^{-1}\right), \qquad n \to \infty.$$

where $q_{2n+1}(\theta)$ is the trigonometric interpolant of $g(\theta)$ and $p_{n+1}(x)$ is the Chebyshev interpolant of $f(x)$.

2. For $f(x) = |x|^3$ we want to know how fast does the derivative of the Chebyshev interpolant converge to f'. Following the same reasoning as in the previous example, since $g(\theta) \in C^2[0,2\pi]$, $\|g^{(k)}\|_1 < \infty$ for $k = 0,\ldots,4$ and $g^{(3)}(0), g^{(3)}(2\pi) < \infty$, it follows that

$$\|q'_{2n+1}(\theta) - g'(\theta)\|_\infty = \left\|\sqrt{1-x^2}(p'_{n+1}(x) - f'(x))\right\|_\infty = \mathcal{O}\left(n^{-2}\right), \qquad n \to \infty.$$

3. For $f(x) = \mathrm{e}^{-1/x^2}$, we have that $g(\theta) = f(\cos\theta) \in C^\infty[0,2\pi]$ and thus

$$\|q_{2n+1}(\theta) - g(\theta)\|_\infty = \|p_{n+1}(x) - f(x)\|_\infty = \mathcal{O}\left(n^{-\mu}\right), \qquad n \to \infty,$$

for $\mu = 1,2,3,\ldots$ and this rate of convergence also holds for $\|q^{(\nu)}_{2n+1}(\theta) - g^{(\nu)}(\theta)\|_\infty$, $\nu = 0,1,2,\ldots$.

4. For the Runge function $f(x) = 1/(1+25x^2)$, $g(\theta) = f(\cos\theta)$ is analytic on $[0,2\pi]$ and therefore $\|q^{(\nu)}_{2n+1}(\theta) - g^{(\nu)}(\theta)\|_\infty$ and $\|p^{(\nu)}_{n+1}(x) - f^{(\nu)}(x)\|_\infty$ decays exponentially fast with n.

5. For $f(x) = x^m$, we have that

$$\|q_{2n+1}(\theta) - g(\theta)\|_\infty = \|p_{n+1}(x) - f(x)\|_\infty = 0$$

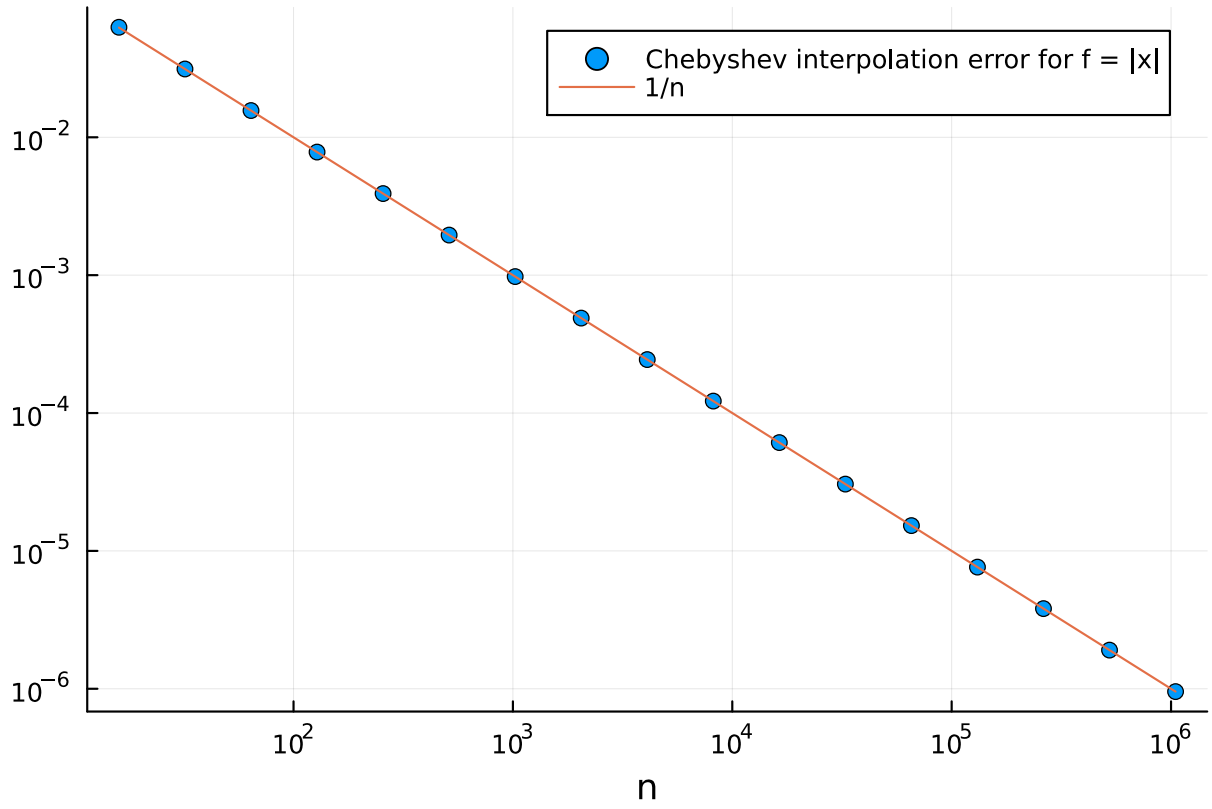
for $n \geq m$ because $q_{2n+1}(\theta) = g(\theta)$ and $p_{n+1}(x) = f(x)$ for $n \geq m$. Therefore, we also have

$$\|q^{(\nu)}_{2n+1}(\theta) - g^{(\nu)}(\theta)\|_\infty = \|p^{(\nu)}_{n+1}(x) - f^{(\nu)}(x)\|_\infty = 0$$

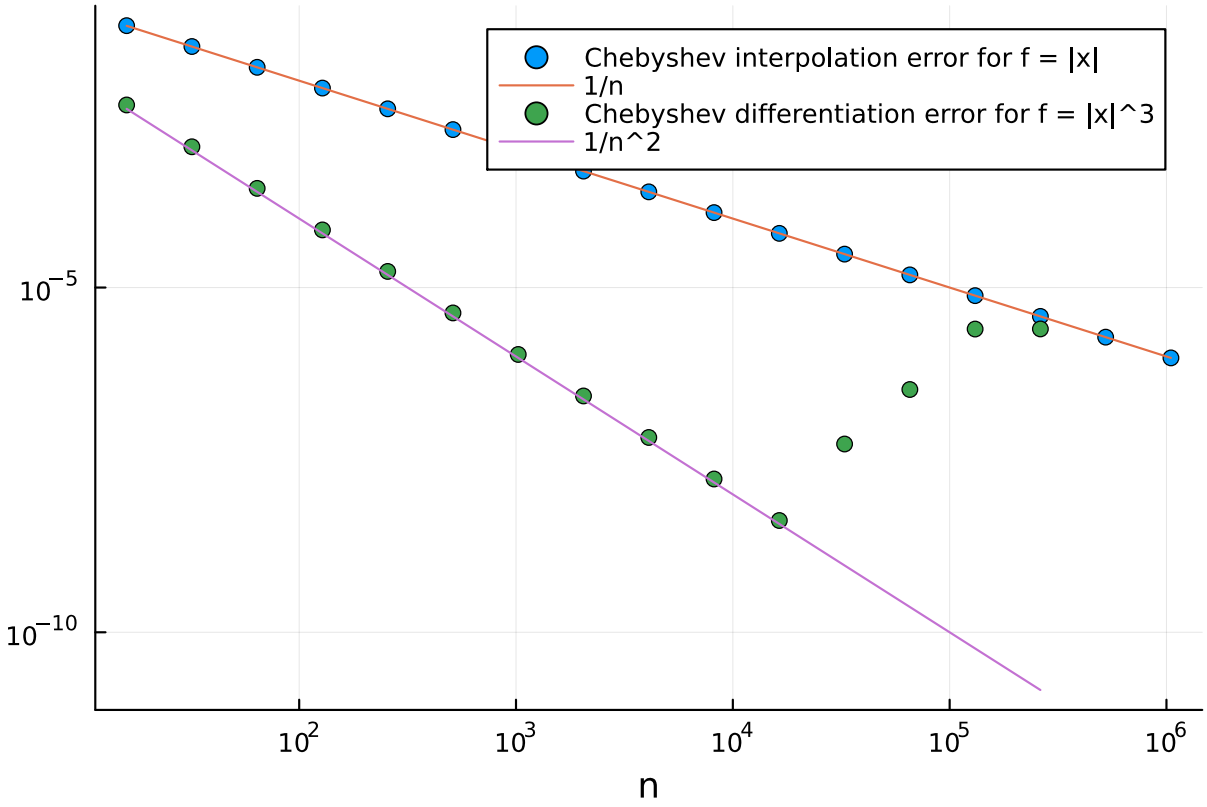
for $\nu = 0,1,\ldots$ if $n \geq m$.

Let's check whether the numerics agree with these predictions:

```
f_1 = x -> abs(x)
xx = range(-1,1;length=1001) # compute the error on this grid
nn = 2 .^(4:20)
S = Chebyshev()
# use ApproxFun to construct and evaluate Chebyshev interpolant
errs = [( p_n = Fun(f_1,S,n);
        norm(f_1.(xx) - p_n.(xx),Inf)  ) for n = nn ]
p=scatter(nn,errs;xscale=:log10, yscale=:log10,xlabel="n",
label="Chebyshev interpolation error for f = |x|")
plot!(nn,1 ./nn;label="1/n")
```

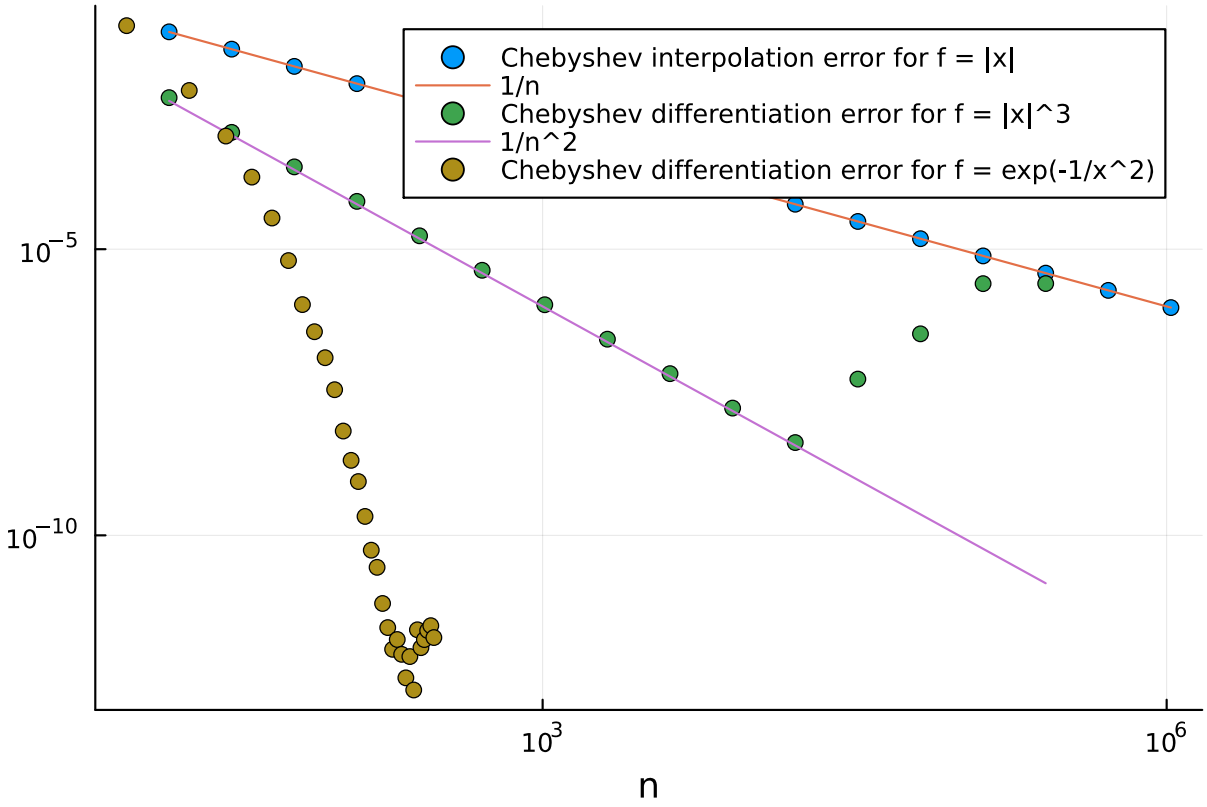



```
f_2 = x -> abs(x)^3
df_2 = x -> x < 0 ? -3x^2 : 3x^2
nn = 2 .^(4:18)
errs = [( x = cos.(π*(0:n)/n);
          norm(df_2.(x) - chebfft(f_2.(x)),Inf)) for n = nn]
p = scatter!(nn,errs;xscale=:log10, yscale=:log10,xlabel="n",
label="Chebyshev differentiation error for f = |x|^3")
plot!(nn,1 ./nn.^2;label="1/n^2")
```

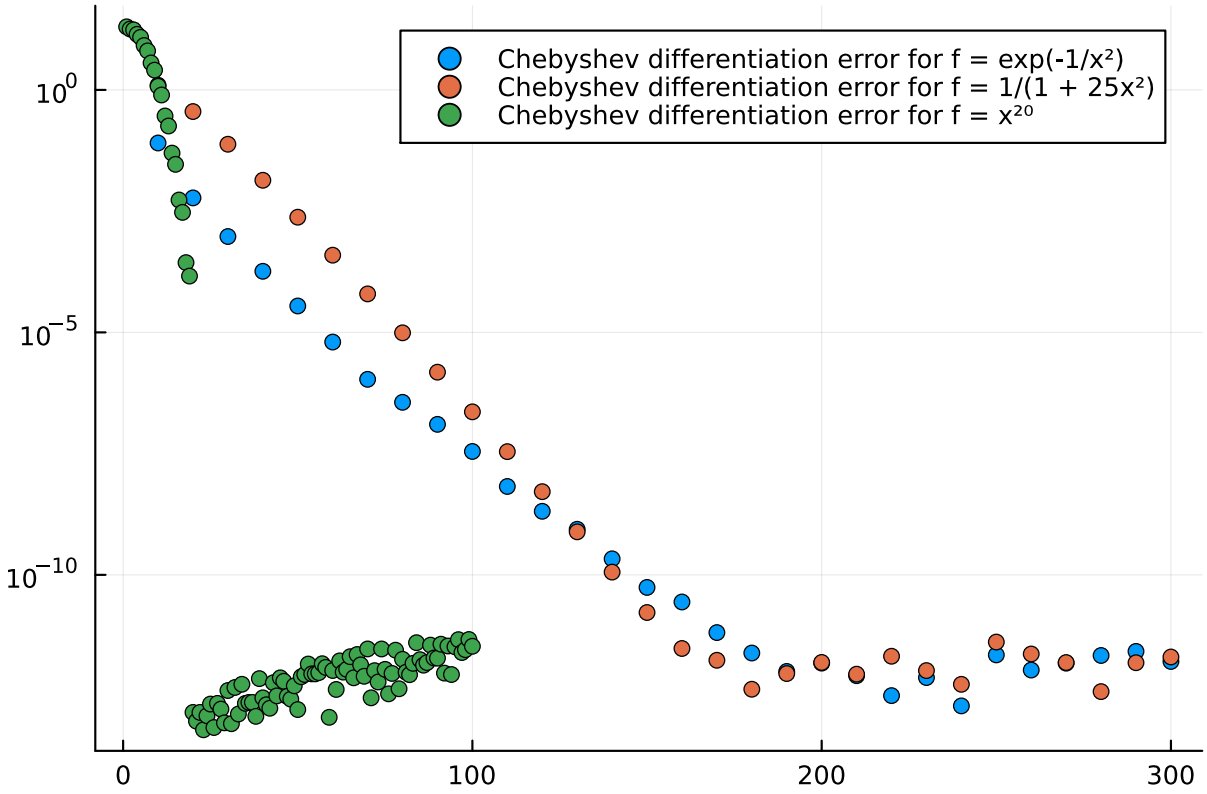


The increase of the error for Chebyshev differentiation is caused by a combination of rounding errors on the order of 10^{-15} and the $\mathcal{O}(n^2)$ growth in the formulae for Chebyshev differentiation at $x = \pm 1$ (see the function `chebfft`).

```
f_3 = x -> exp(-1/x^2)
df_3 = x -> x == 0 ? 0 : 2*exp(-1/x^2)/x^3
nn = 10:10:300
errs = [( x = cos.(π*(0:n)/n);
          norm(df_3.(x) - chebfft(f_3.(x)),Inf)) for n = nn]
p = scatter!(nn,errs;xscale=:log10, yscale=:log10,xlabel="n",
label="Chebyshev differentiation error for f = exp(-1/x^2)")
```




```
scatter(nn,errs;yscale=:log10,label="Chebyshev differentiation error for f = exp(-1/x2)")
f_4 = x -> 1/(1 + 25x^2)
df_4 = x ->  -50*x/(1 + 25x^2)^2
nn = 10:10:300
errs = [( x = cos.(π*(0:n)/n);
          norm(df_4.(x) - chebfft(f_4.(x)),Inf)) for n = nn]
scatter!(nn,errs;label="Chebyshev differentiation error for f = 1/(1 + 25x2)")
f_5 = x -> x^(20)
df_5 = x ->  20*x^19
nn = 1:100
errs = [( x = cos.(π*(0:n)/n);
          norm(df_5.(x) - chebfft(f_5.(x)),Inf)) for n = nn]
scatter!(nn,errs;label="Chebyshev differentiation error for f = x20")
```



Note that this a plot with semi-logarithmic axes whereas the previous plot is a log-log set of axes.