Applied Complex Analysis (2021)

1 Lecture 22: Hermite polynomials

This lecture we overview features of Hermite polynomials, some of which also apply to Jacobi polynomials. This includes

1. Rodriguez formula
2. Approximation with Hermite polynomials
3. Eigenstates of Schrödinger equations with a quadratic well

1.1 Rodriguez formula

Because of the special structure of classical orthogonal weights, we have special Rodriguez formulae of the form

$$p_n(x) = \frac{1}{\kappa_n w(x)} \frac{\mathrm{d}^n}{\mathrm{d}x^n} w(x) \left[F(x)\right]^n$$

where $w(x)$ is the weight and $F(x) = (1 - x^2)$ (Jacobi), x (Laguerre) or 1 (Hermite) and κ_n is a normalization constant.

Proposition (Hermite Rodriguez)

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

Proof We first show that it's a degree n polynomial. This proceeds by induction:

$$H_0(x) = \mathrm{e}^{x^2} \frac{\mathrm{d}^0}{\mathrm{d}x^0} \mathrm{e}^{-x^2} = 1$$
$$H_{n+1}(x) = -\mathrm{e}^{x^2} \frac{\mathrm{d}}{\mathrm{d}x} \left[\mathrm{e}^{-x^2} H_n(x) \right] = 2x H_n(x) - H_n'(x)$$

and then we have

$$\frac{\mathrm{d}^n}{\mathrm{d}x^n} [p_m(x) \mathrm{e}^{-x^2}] = \frac{\mathrm{d}^{n-1}}{\mathrm{d}x^{n-1}} (p_m'(x) - 2x p_m(x)) \mathrm{e}^{-x^2}$$

Orthogonality follows from integration by parts:

$$\langle H_n, p_m \rangle_{\mathrm{H}} = (-1)^n \int_{-\infty}^{\infty} \frac{\mathrm{d}^n \mathrm{e}^{-x^2}}{\mathrm{d}x^n} p_m \mathrm{d}x = \int_{-\infty}^{\infty} \mathrm{e}^{-x^2} \frac{\mathrm{d}^n p_m}{\mathrm{d}x^n} \mathrm{d}x = 0$$

if $m < n$.

Now we just need to show we have the right constant. But we have

$$\frac{\mathrm{d}^n}{\mathrm{d}x^n} [\mathrm{e}^{-x^2}] = \frac{\mathrm{d}^{n-1}}{\mathrm{d}x^{n-1}} [-2x \mathrm{e}^{-x^2}] = \frac{\mathrm{d}^{n-2}}{\mathrm{d}x^{n-2}} [(4x^2 + O(x)) \mathrm{e}^{-x^2}] = \cdots = (-1)^n 2^n x^n$$

■

Note this tells us the Hermite recurrence: Here we have the simple expressions

$$H_n'(x) = 2nH_{n-1}(x) \qquad \text{and} \qquad \frac{\mathrm{d}}{\mathrm{d}x}[\mathrm{e}^{-x^2}H_n(x)] = -\mathrm{e}^{-x^2}H_{n+1}(x)$$

These follow from the same arguments as before since $w'(x) = -2xw(x)$. But using the Rodriguez formula, we get

$$2nH_{n-1}(x) = H_n'(x) = (-1)^n 2x\mathrm{e}^{x^2}\frac{\mathrm{d}^n}{\mathrm{d}x^n}\mathrm{e}^{-x^2} + (-1)^n\mathrm{e}^{x^2}\frac{\mathrm{d}^{n+1}}{\mathrm{d}x^{n+1}}\mathrm{e}^{-x^2} = 2xH_n(x) - H_{n+1}(x)$$

which means

$$xH_n(x) = nH_{n-1}(x) + \frac{H_{n+1}(x)}{2}$$

1.2 Approximation with Hermite polynomials

Hermite polynomials are typically used with the weight for approximation of functions: on the real line polynomial approximation is unnatural unless the function approximated is a polynomial as otherwise the behaviour at ∞ is inconsistent (polynomials blow up). Thus we can either use

$$f(x) = \mathrm{e}^{-x^2} \sum_{k=0}^{\infty} f_k H_k(x)$$

or

$$f(x) = \mathrm{e}^{-x^2/2} \sum_{k=0}^{\infty} f_k H_k(x)$$

** Demonstration **

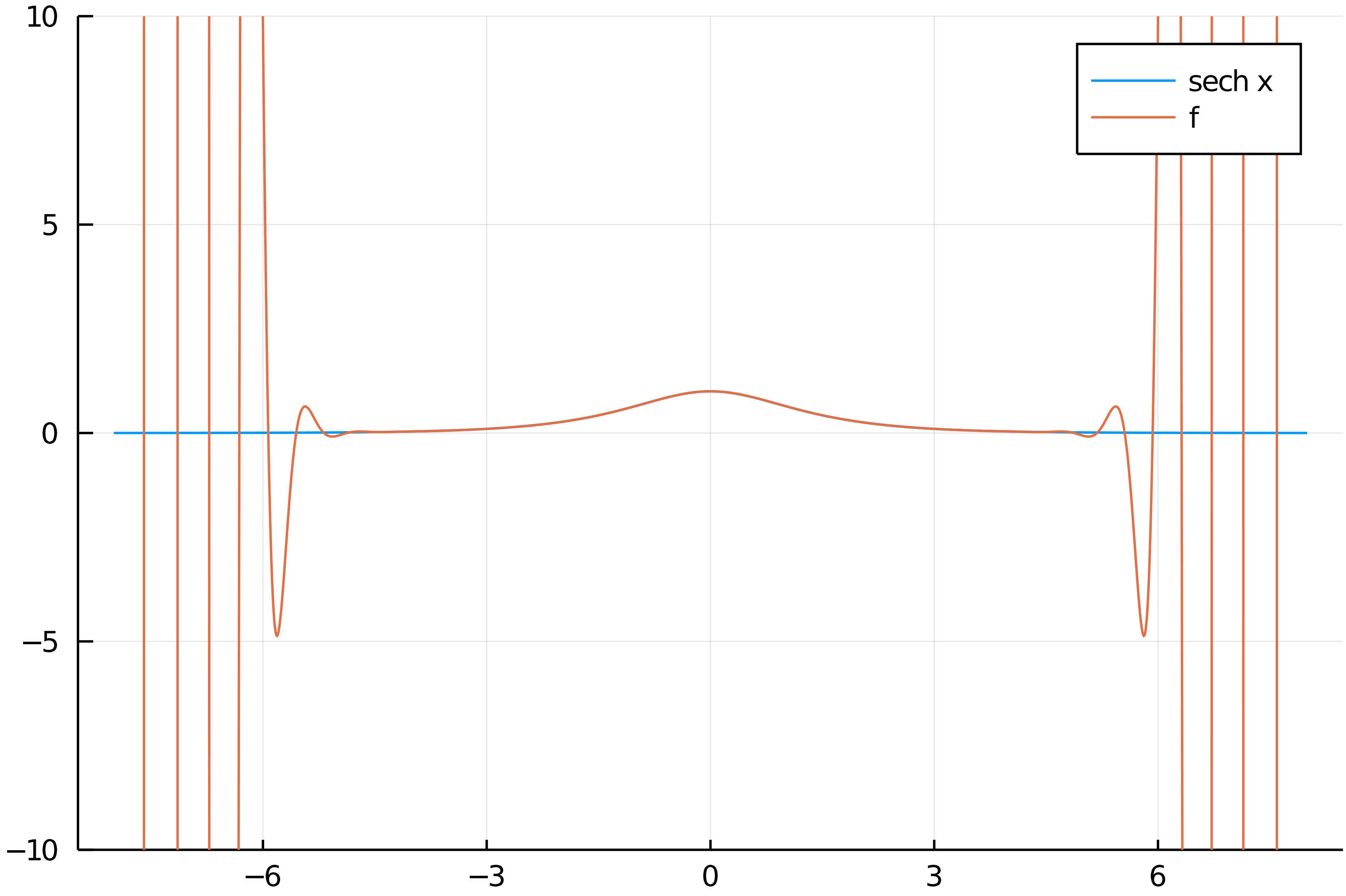
Depending on your problem, getting this wrong can be disasterous. For example, while we can certainly approximate polynomials with Hermite expansions:

```
using ApproxFun, Plots
f = Fun(x -> 1+x +x^2, Hermite())
f(0.10)
```

```
1.1099999999999997
```

We get nonsense when trying to approximating $\operatorname{sech}(x)$ by a degree 50 polynomial:

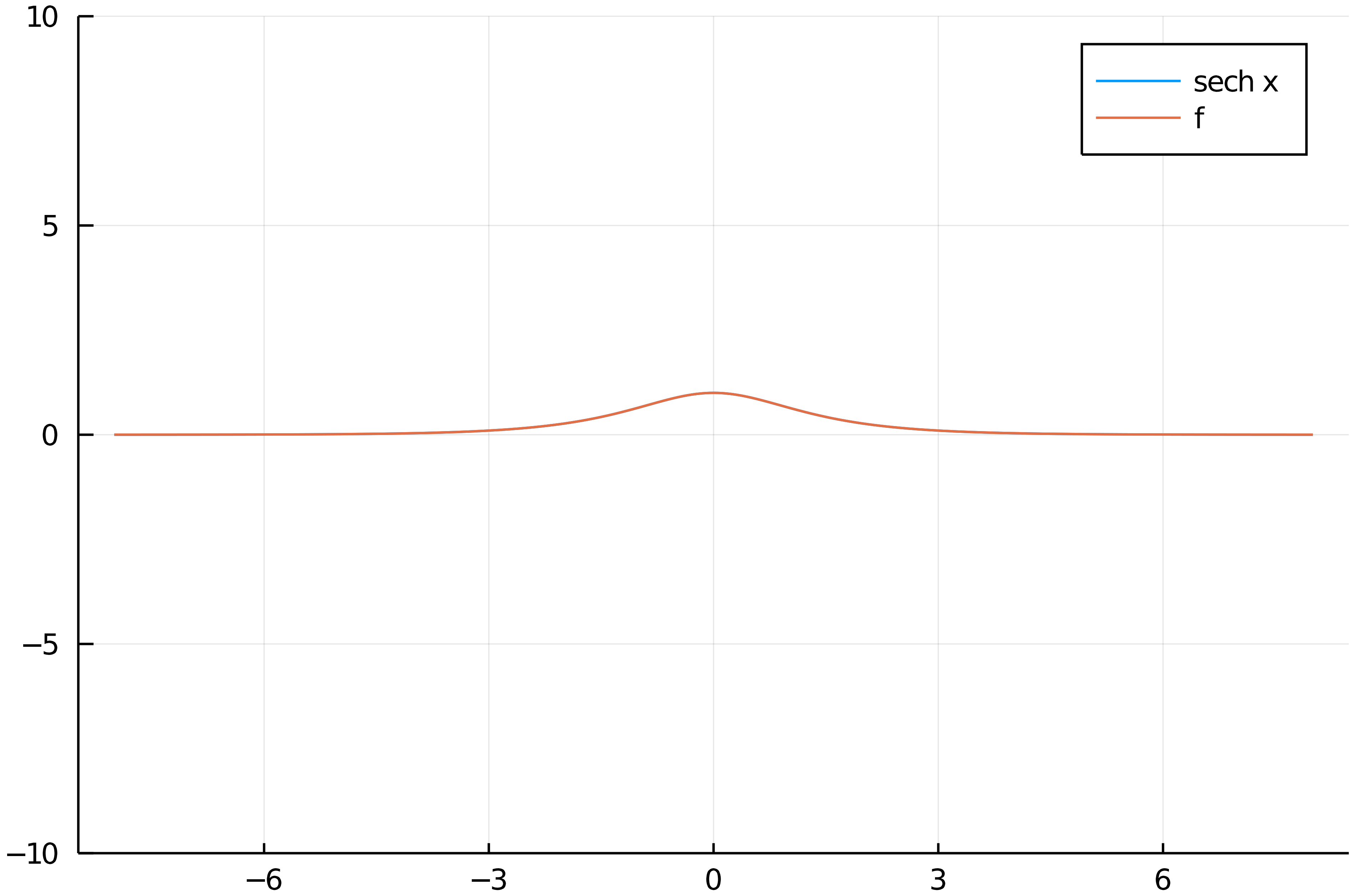
```julia
f = Fun(x -> sech(x), Hermite(), 51)
xx = -8:0.01:8
plot(xx, sech.(xx); ylims=(-10,10), label="sech x")
plot!(xx, f.(xx); label="f")
```



Incorporating the weight $\sqrt{w(x)} = \mathrm{e}^{-x^2/2}$ works:

```
f = Fun(x -> sech(x), GaussWeight(Hermite(),1/2),101)
```

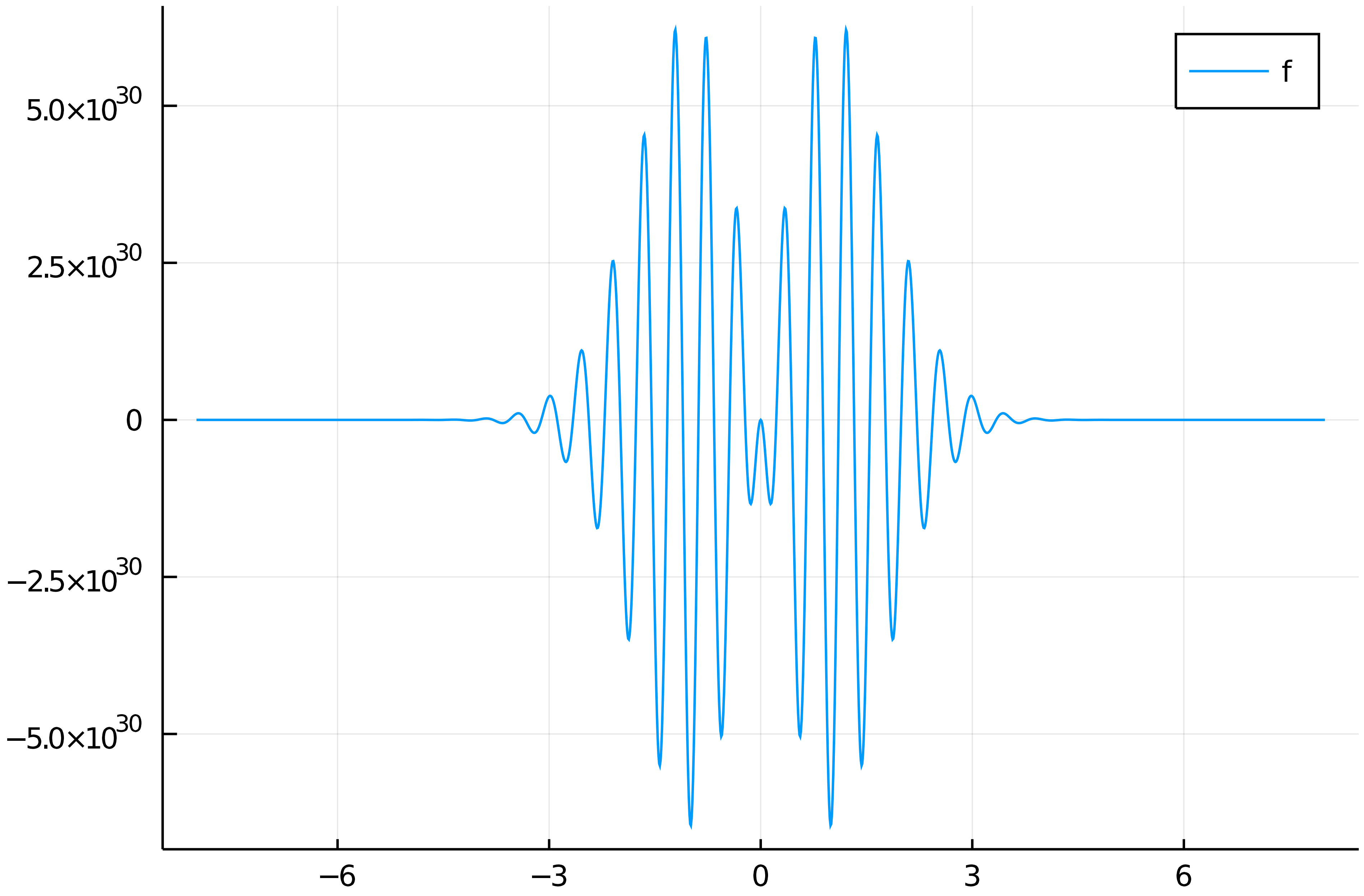
```julia
plot(xx, sech.(xx); ylims=(-10,10), label="sech x")
plot!(xx, f.(xx); label="f")
```



Weighted by w(x) = exp(-x^2) breaks again:

```
f = Fun(x -> sech(x), GaussWeight(Hermite()),101)
```

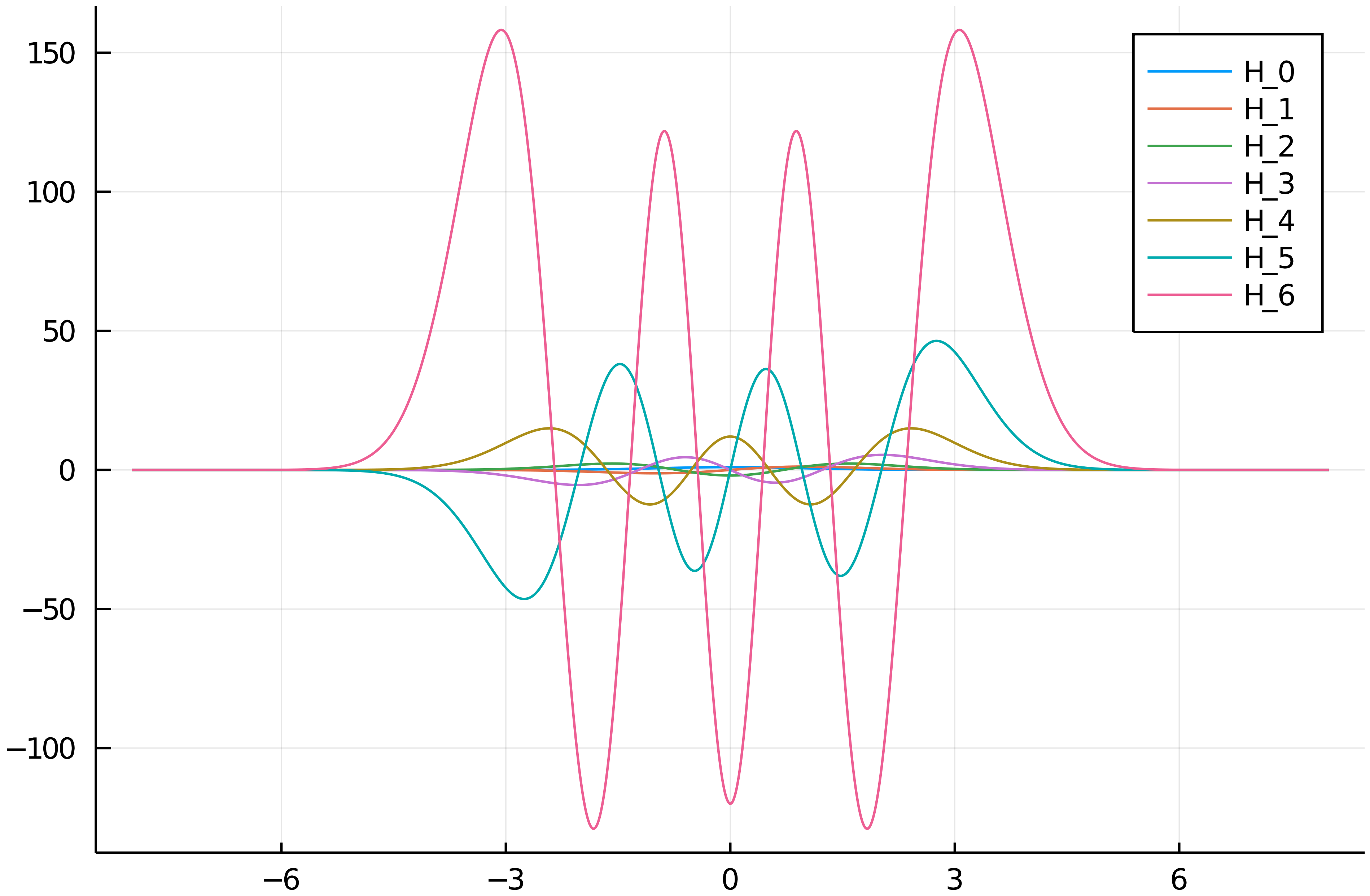
```
plot(xx, sech.(xx); ylims=(-10,10), label="sech x")
plot(xx, f.(xx); label="f")
```



Note that correctly weighted Hermite, that is, with $\sqrt{w(x)} = \mathrm{e}^{-x^2/2}$ look "nice":

```
p = plot()
```

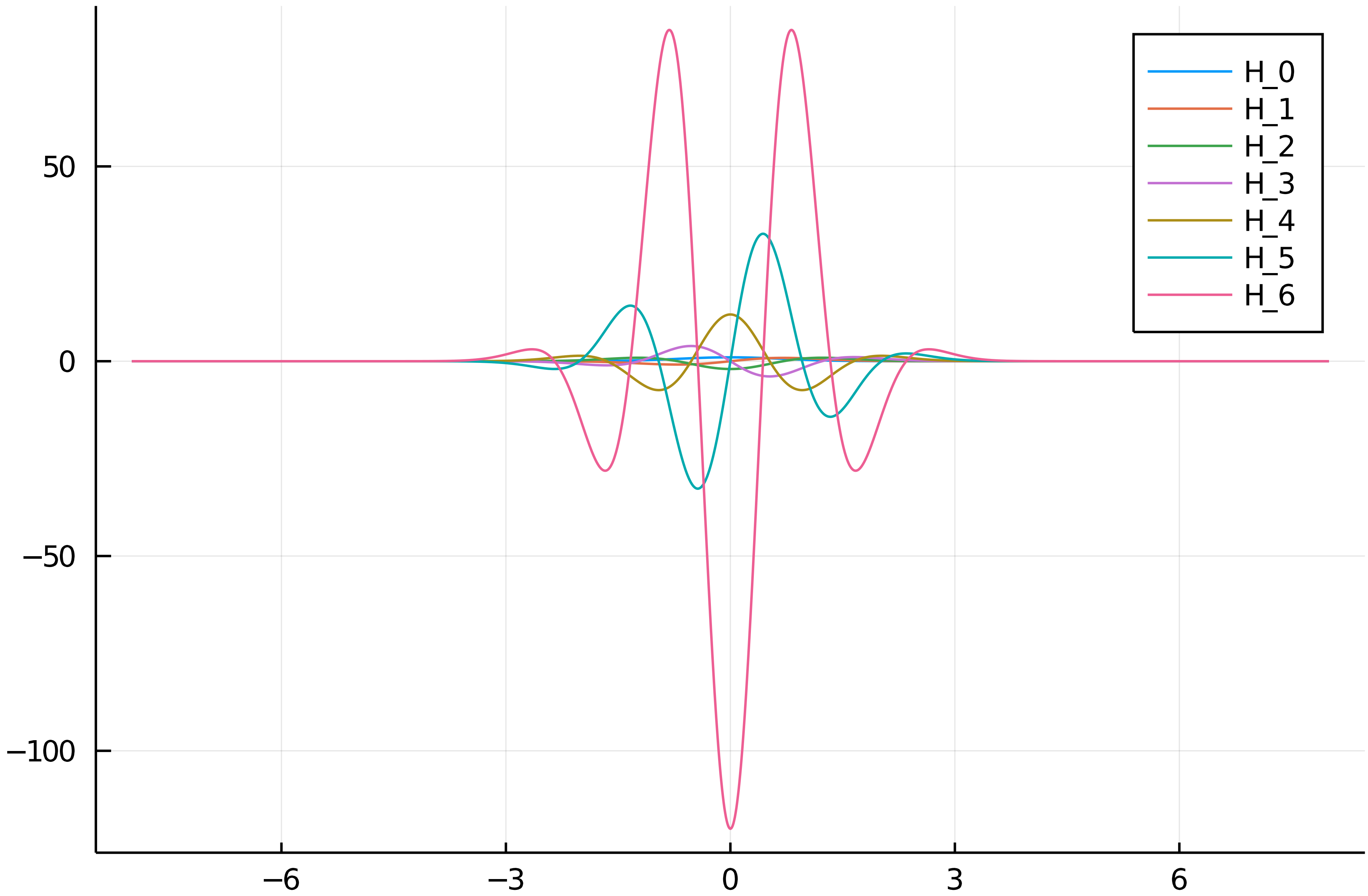
```julia
for k=0:6
    H_k = Fun(GaussWeight(Hermite(),1/2),[zeros(k);1])
    plot!(xx, H_k.(xx); label="H_$k")
end
p
```



Compare this to weighting by $w(x) = \mathrm{e}^{-x^2}$:

```
p = plot()
```

```julia
for k=0:6
    H_k = Fun(GaussWeight(Hermite()),[zeros(k);1])
    plot!(xx, H_k.(xx); label="H_$k")
end
p
```


50
0
−50
−100
−6
−3
0
3
6
H_0
H_1
H_2
H_3
H_4
H_5
H_6

1.3 Application: Eigenstates of Schrödinger operators with quadratic potentials

Using the derivative formulae tells us a Sturm–Liouville operator for Hermite polynomials:

$$\mathrm{e}^{x^2}\frac{\mathrm{d}}{\mathrm{d}x}\mathrm{e}^{-x^2}\frac{\mathrm{d}H_n}{\mathrm{d}x} = 2n\mathrm{e}^{x^2}\frac{\mathrm{d}}{\mathrm{d}x}\mathrm{e}^{-x^2}H_{n-1}(x) = -2nH_n(x)$$

or rewritten, this gives us

$$\frac{\mathrm{d}^2H_n}{\mathrm{d}x^2} - 2x\frac{\mathrm{d}H_n}{\mathrm{d}x} = -2nH_n(x)$$

We therefore have

$$\frac{\mathrm{d}^2}{\mathrm{d}x^2}[\mathrm{e}^{-\frac{x^2}{2}}H_n(x)] = \mathrm{e}^{-\frac{x^2}{2}}(H_n''(x) - 2xH_n'(x) + (x^2-1)H_n(x)) = \mathrm{e}^{-\frac{x^2}{2}}(x^2-1-2n)H_n(x)$$

In other words, for the Hermite function $\psi_n(x)$ we have

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

and therefore ψ_n are the eigenfunctions.

We want to normalize. In Schrödinger equations the square of the wave $\psi(x)^2$ represents a probability distribution, which should integrate to 1. Here's a trick: we know that

$$x\begin{pmatrix}H_0(x)\\H_1(x)\\H_2(x)\\\vdots\end{pmatrix}=\underbrace{\begin{pmatrix}0&\frac{1}{2}&&&\\1&0&\frac{1}{2}&&\\&2&0&\frac{1}{2}&\\&&3&0&\ddots\\&&&\ddots&\ddots\end{pmatrix}}_{J}\begin{pmatrix}H_0(x)\\H_1(x)\\H_2(x)\\\vdots\end{pmatrix}$$

We want to conjugate by a diagonal matrix so that

$$\begin{pmatrix}1&&&\\&d_1&&\\&&d_2&\\&&&\ddots\end{pmatrix}J\begin{pmatrix}1&&&\\&d_1^{-1}&&\\&&d_2^{-1}&\\&&&\ddots\end{pmatrix}=\begin{pmatrix}0&\frac{1}{2d_1}&&&\\d_1&0&\frac{d_1}{2d_2}&&\\&\frac{2d_2}{d_1}&0&\frac{d_2}{2d_3}&\\&&\frac{3d_3}{d_2}&0&\ddots\\&&&\ddots&\ddots\end{pmatrix}$$

becomes symmetric. This becomes a sequence of equations:

$$
\begin{aligned}
d_1 = \frac{1}{2d_1} &\Rightarrow d_1^2 = \frac{1}{2} \\
2d_2 d_1^{-1} = \frac{d_1}{2d_2} &\Rightarrow d_2^2 = \frac{d_1^2}{4} = \frac{1}{8} = \frac{1}{2^2 2!} \\
3d_3 d_2^{-1} = \frac{d_2}{2d_3} &\Rightarrow d_3^2 = \frac{d_2^2}{3 \times 2} = \frac{1}{2^3 3!} \\
&\vdots \\
d_n^2 = \frac{1}{2^n n!}
\end{aligned}
$$

Thus the norm of $d_n H_n(x)$ is constant. If we also normalize using

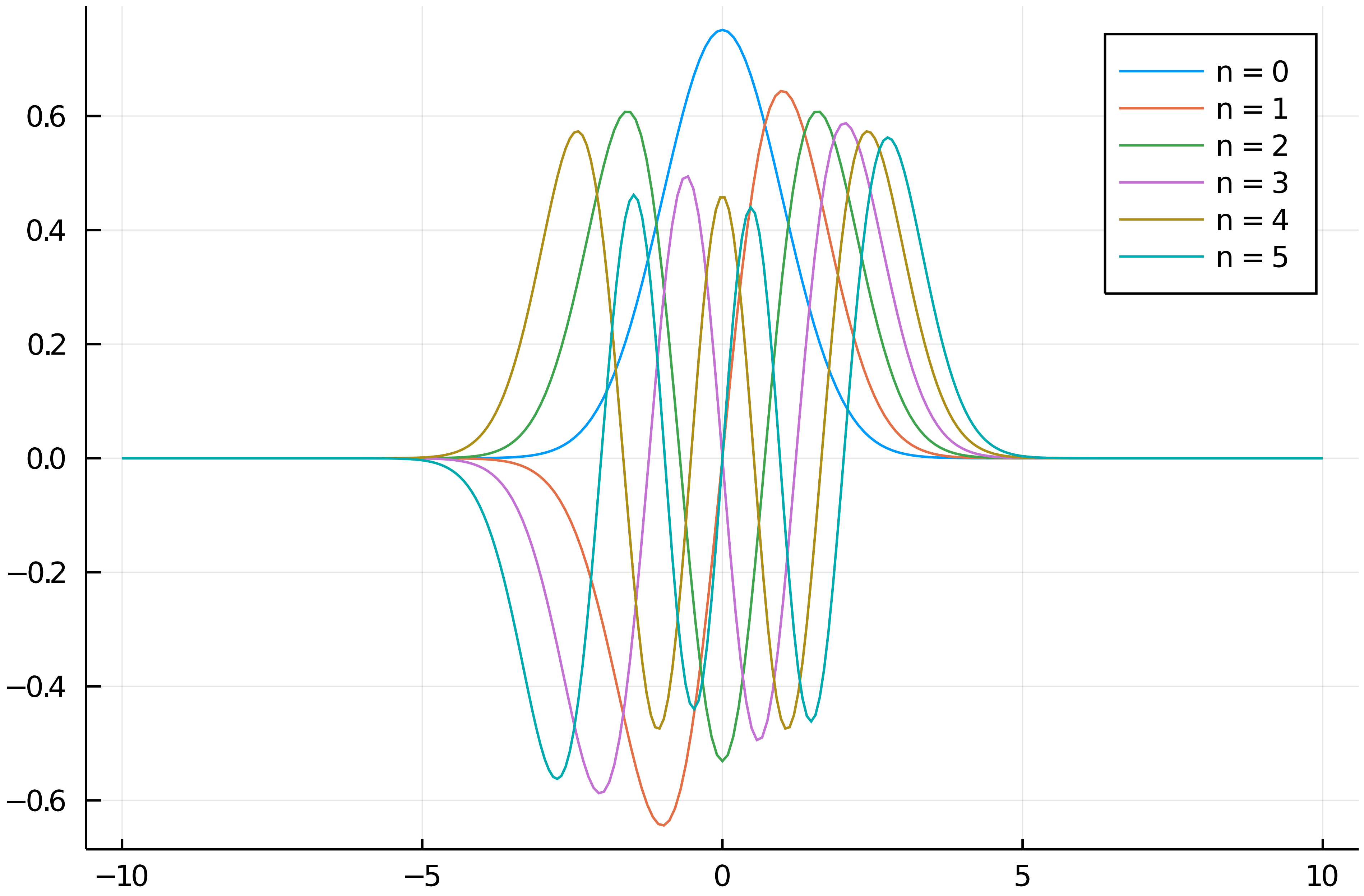
$$\int_{-\infty}^{\infty} \mathrm{e}^{-x^2} \mathrm{d}x = \sqrt{\pi}$$

we get the normalized eigenfunctions

$$\psi_n(x) = \frac{H_n(x)\mathrm{e}^{-x^2/2}}{\sqrt{\sqrt{\pi}2^n n!}}$$

```
p = plot()
for n = 0:5
    H = Fun(Hermite(), [zeros(n);1])
```

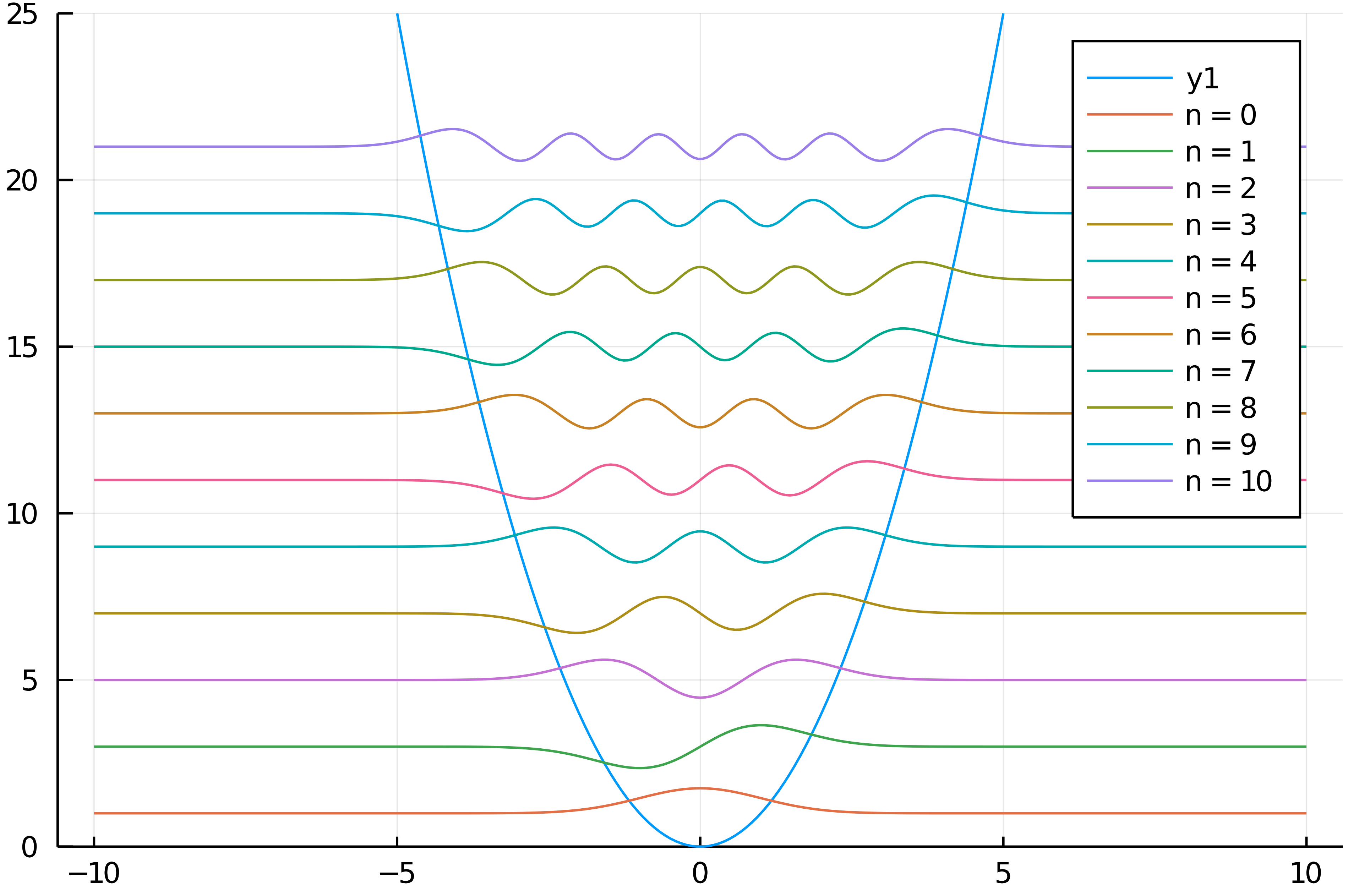
```
    ψ = Fun(x -> H(x)exp(-x^2/2), -10.0 ..
10.0)/sqrt(sqrt(π)*2^n*factorial(1.0n))
    plot!(ψ; label="n = $n")
end
p
```



It's convention to shift them by the eigenvalue:

```
p = plot(pad(Fun(x -> x^2, -10 .. 10), 100); ylims=(0,25))
```

```
for n = 0:10
    H = Fun(Hermite(), [zeros(n);1])
    ψ = Fun(x -> H(x)exp(-x^2/2), -10.0 ..
10.0)/sqrt(sqrt(π)*2^n*factorial(1.0n))
    plot!(ψ + 2n+1; label="n = $n")
end
p
```



25
20
15
10
5
0
−10
−5
0
5
10
y1
n = 0
n = 1
n = 2
n = 3
n = 4
n = 5
n = 6
n = 7
n = 8
n = 9
n = 10