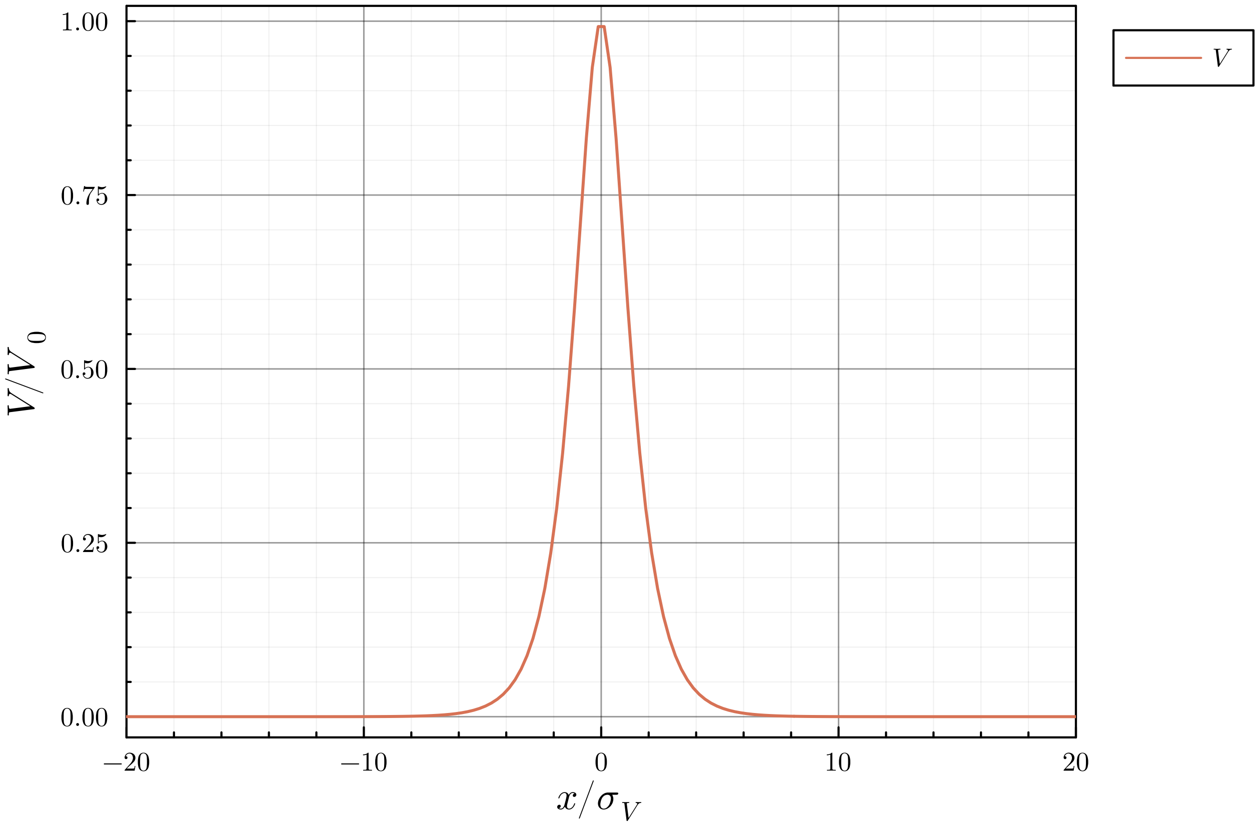
```julia
using DifferentialEquations , LinearAlgebra , Plots , StatsPlots ,
LaTeXStrings , PlotThemes , PlutoUI ; theme(:dao)
```

Classical vs. quantum tunneling

CAS PY 452, Fall 2022

Emmy Blumenthal

```julia
# define problem parameters
begin
    ħ = 0.2;
    m = 1;
    V₁ = 0.06;
    σᵥ = 1.0;
    V(x) = V₁ / cosh(x / σᵥ);
    F(x) = (V₁/σᵥ)*sech(x/σᵥ)*tanh(x/σᵥ)
    nPts=4000;
    L = 1000;
    Δx = L/(nPts-1);
    x=collect((-L/2):Δx:(L/2));
    Vx = V.(x);
end;
```



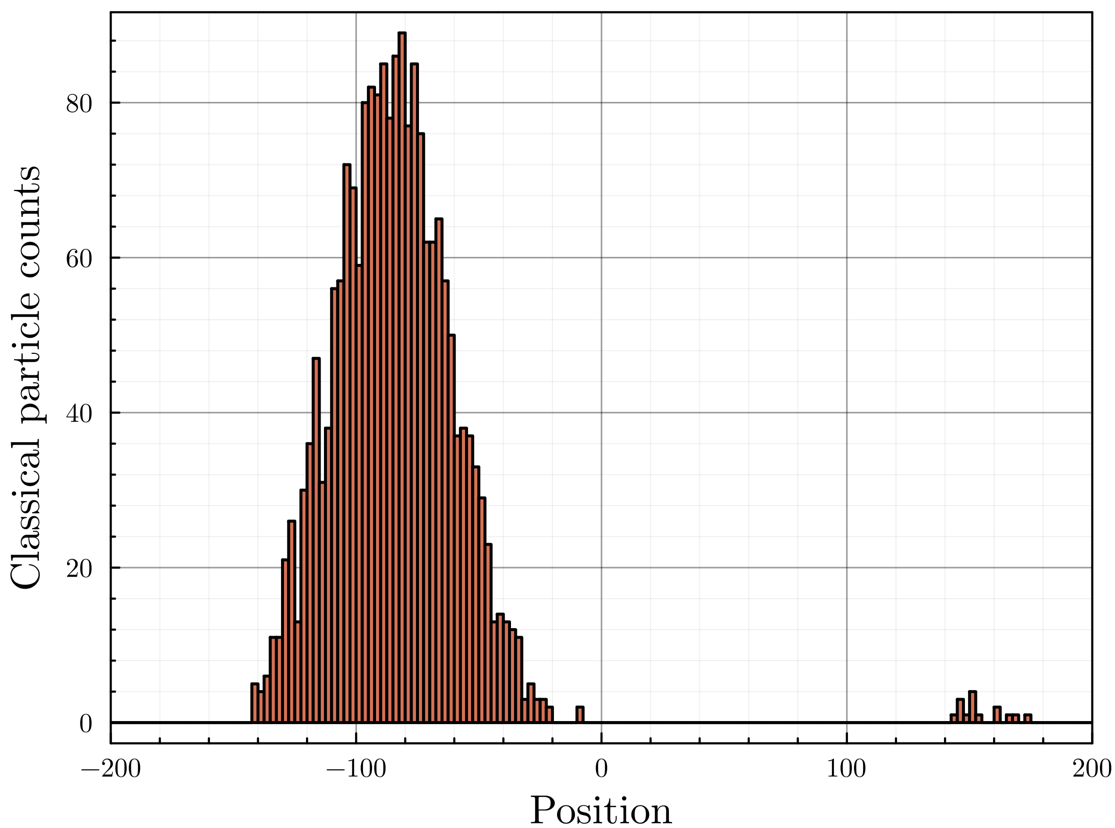
Classical simulations:

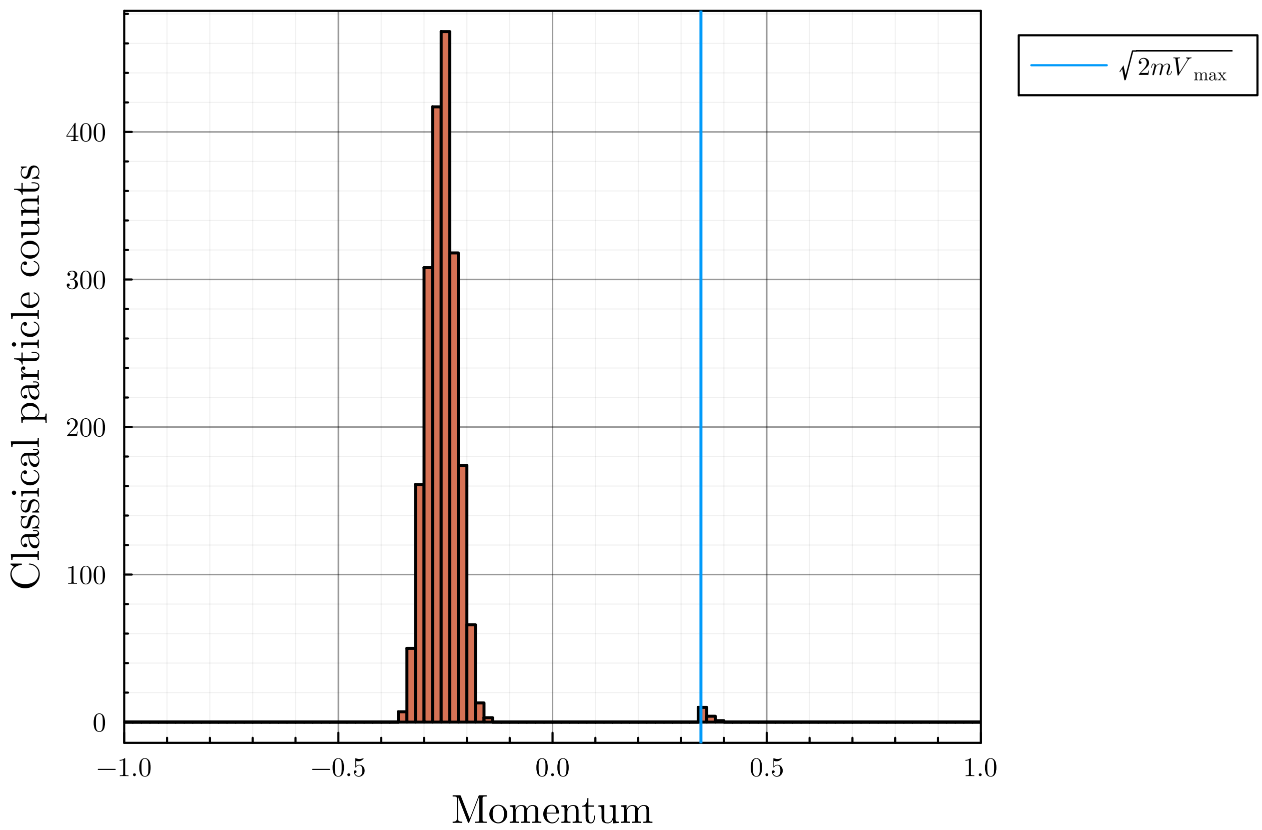
```
# Newton's second law
function eom!(du,u,params,t)
    du[1] = u[2]
    du[2] = F(u[1])/m
end;
```

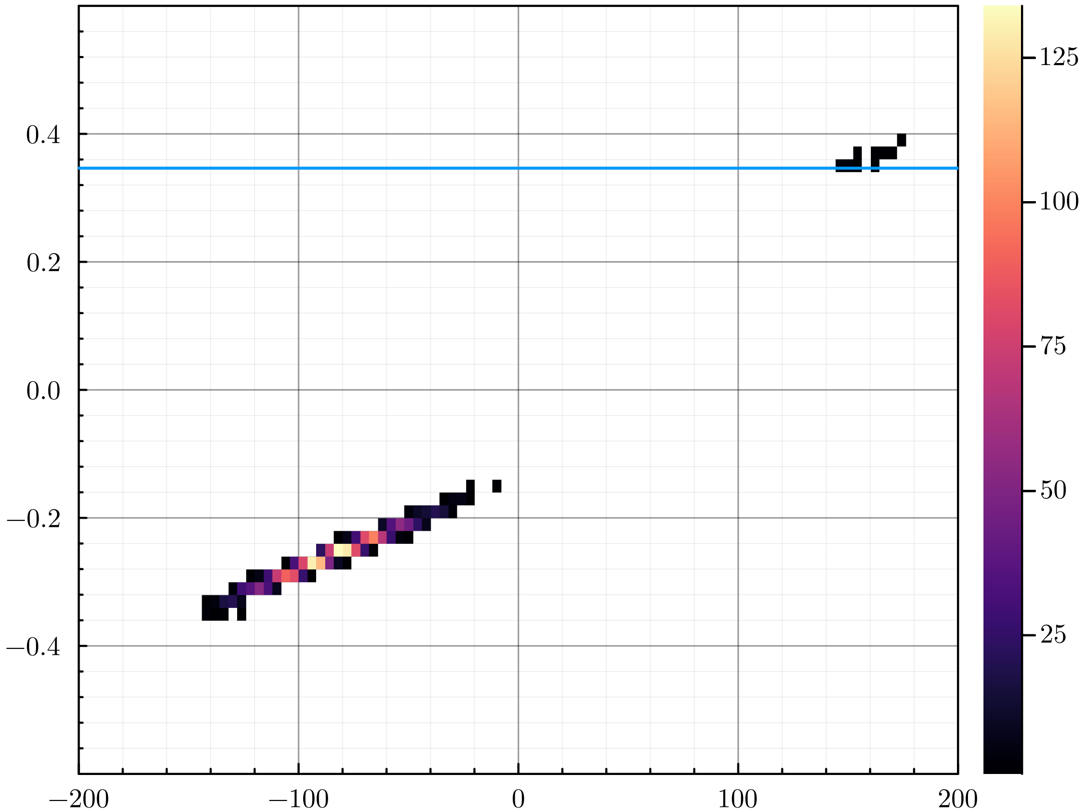
```
# Get a solution given initial position and momentum
function trajec(x0,p0)
    u0 = [x0,p0/m];
    tspan = (0.0,1000.0);
    prob = ODEProblem(eom!,u0,tspan);
    sol = solve(prob,dtmax=1,saveat=4,reltol=1e-8,abstol=1e-8);
    return sol
end;
```

```
# Choose parameters for initial ensemble distribution
x₀ = -100; σₓ₀ = 3.0; p₀ = 0.26; σₚ₀ = ħ/(2σₓ₀);
```

```
begin #solve differential equations for 1000 particles
    ensemblepstemp = []; ensemblexstemp = []; tsclassical = 0:2:1000;
    for _ in 1:2000
        let p = p₀ + σp₀*randn(), x = x₀ + σx₀*randn()
            atrajec = trajec(x,p);
            push!(ensemblexstemp,atrajec(tsclassical,idxs=1).u)
            push!(ensemblepstemp,atrajec(tsclassical,idxs=2).u)
        end
    end
    ensemblexs = hcat(ensemblexstemp...)
    ensembleps = hcat(ensemblepstemp...)
end;
```




Classical particle counts
0
100
200
300
400
−1.0
−0.5
0.0
0.5
1.0
Momentum
$\sqrt{2mV_{\max}}$




```julia
begin
    histogram2d(ensemblexs[first(eachindex(tsclassical)[tsclassical .==
    plottclassical]),:],ensembleps[first(eachindex(tsclassical)[tsclassical .==
    plottclassical]),:],xlim=(-200,200),ylim=(-0.6,0.6),bins=
    (-200:4:200,-0.6:0.02:0.6))
    hline!([sqrt(2*m*V₁)],label=:none)
end
```

Observe that a few particles do get through... They are exactly those that have an initial momentum greater than $\sqrt{2mV_{\text{max}}}$

Quantum simulations

```julia
# Define quantum operators:
begin
    ∇² =Δx^(-2)*SymTridiagonal(-2ones(nPts),ones(nPts-1)); #Discrete Laplacian
    V̂ = Diagonal(Vx);
    H₀ = (-ħ^2/(2m))∇²; # Hamiltonian w/ no barrier
    H = (-ħ^2/(2m))∇² + V̂; # Hamiltonian
end;
```

```julia
# compute energies and stationary states:
begin
    # Es, sψs = eigen(H)
    eigs = eigen(H);
    Es = eigs.values;
    sψs = eigs.vectors .* Δx^(-1/2);
    eigfrees = eigen(H₀);
    Efrees = eigfrees.values;
    sψfrees = eigfrees.vectors .* Δx^(-1/2);
end;
```

```julia
# initial wave function
ψ₀ = normalize(exp.(-(x .- x₀).^2 ./(4 * σₓ₀^2) + 1.0im * p₀ * x /ħ)) * Δx^(-1/2);
```

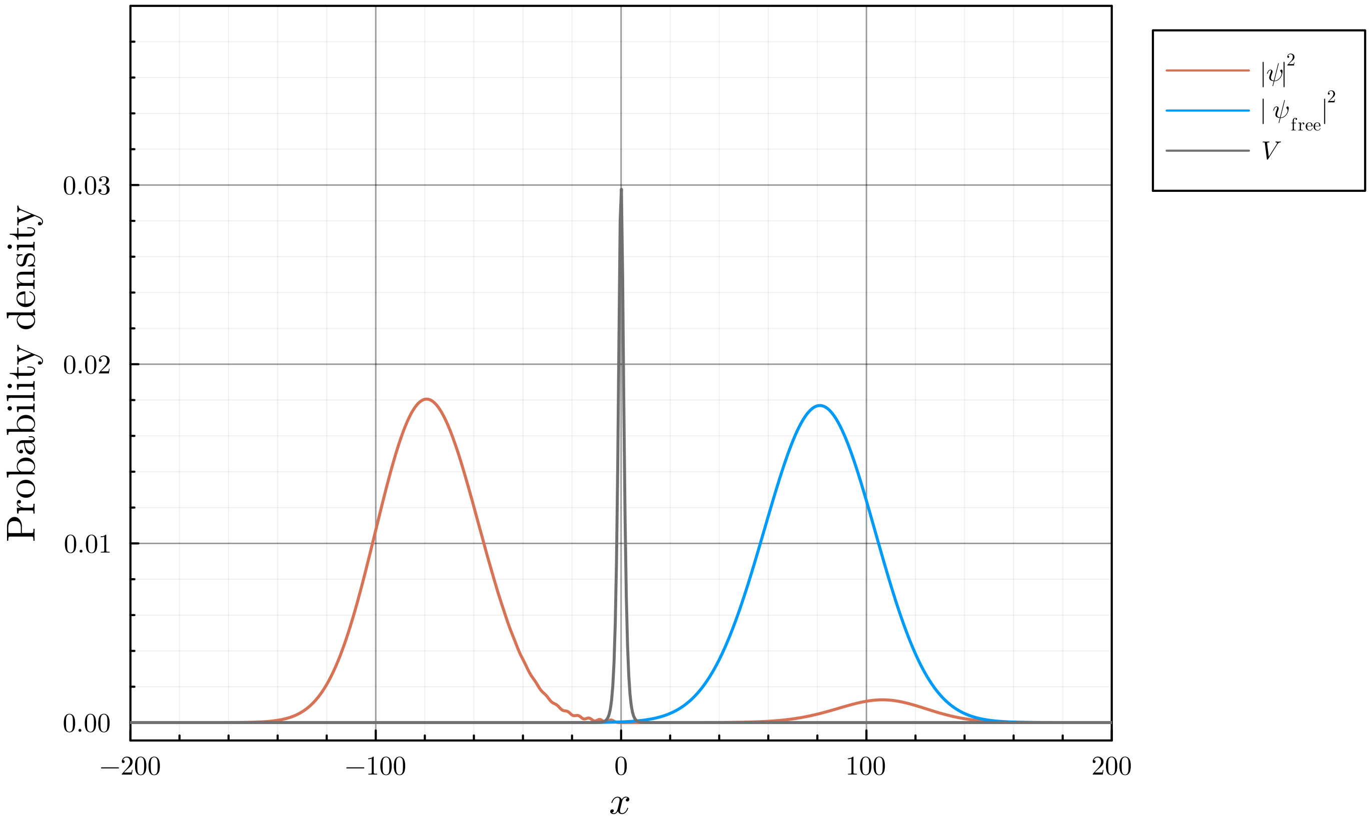
```julia
# time evolution by attaching phase factors
begin
    ψ₀eigenbasis = sψs' * ψ₀; # convert to stationary state basis
    ψ₀freeeigenbasis = sψfrees' * ψ₀;
    ψ(t) = sψs * Diagonal(exp.(-1.0im * t*Es/ħ )) * ψ₀eigenbasis * Δx;
    ψfree(t) = sψfrees * Diagonal(exp.(-1.0im * t*Efrees/ħ ))  *ψ₀freeeigenbasis * Δx;
end;
```

```julia
begin # create vector of time-evolved states
    tmax = 1000; tstep=2.5;
    ψs = ψ.(0:tstep:tmax)
    PDFs = (X->abs.(X).^2).(ψs)
end;
```

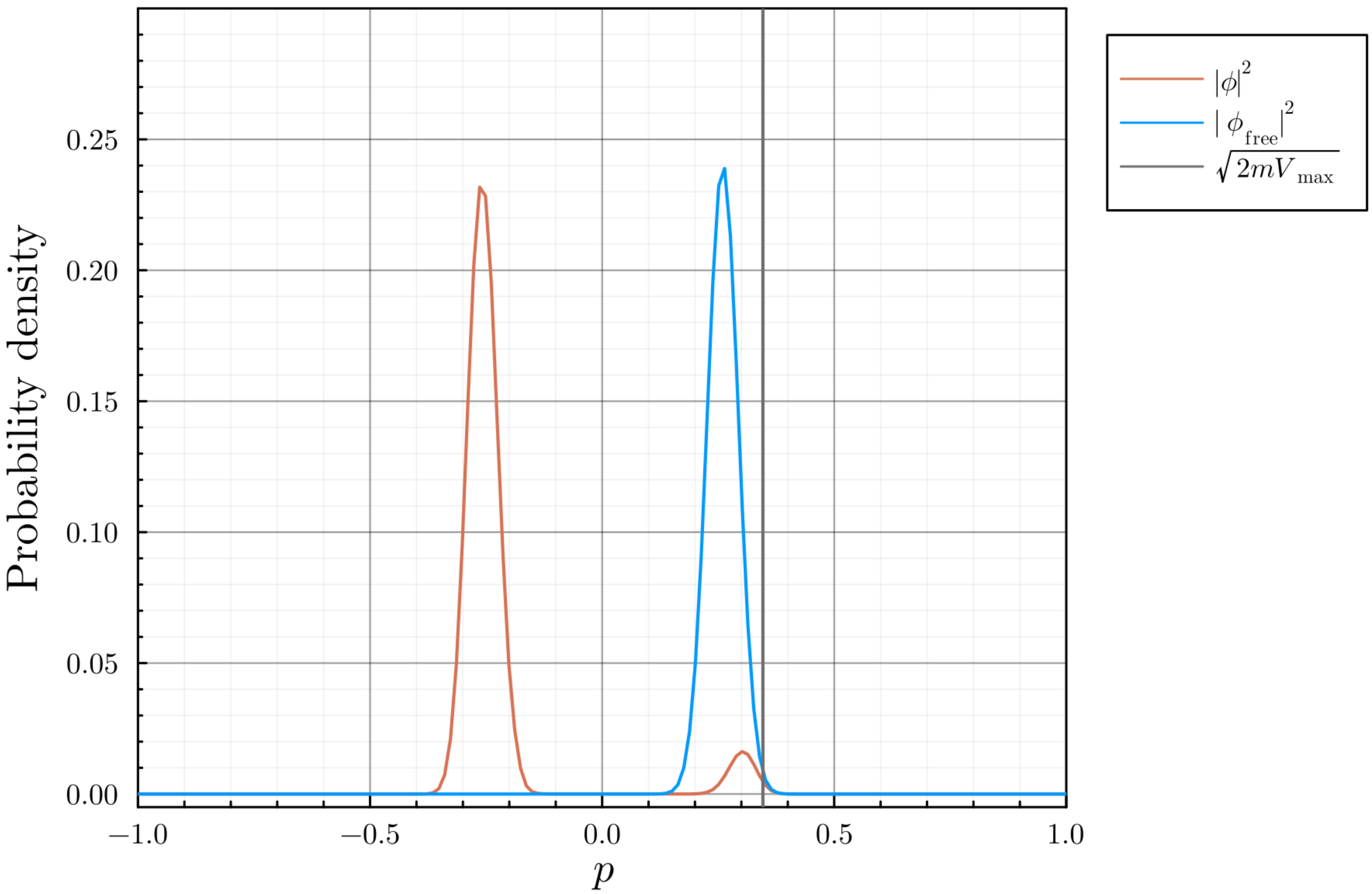
```julia
begin # same for free states
    ψfrees = ψfree.(0:tstep:tmax)
    PDFfrees = (X->abs.(X).^2).(ψfrees)
end;
```

```julia
begin # create a momentum basis
    k = (-2π/Δx ):(2π/(L/2 + Δx/2)):(2π/Δx);
    pBasis = hcat(Δx^(-1/2)*normalize.(eachcol(exp.(1.0im * x * k' /ħ)))...);
end;
```

```julia
begin # evolution in momentum basis
    moms = (X->abs.(pBasis' * X).^2).(ψs);
    momsfree = (X->abs.(pBasis' * X).^2).(ψfrees);
end;
```



$|\psi|^2$
$|\psi_{\mathrm{free}}|^2$
V
Probability density
0.03
0.02
0.01
0.00
−200
−100
0
100
200
x



We see that there is a significant probability that at long times, there will be particles which pass the barrier and have momentum less than $\sqrt{2mV_{\text{max}}}$! This is tunneling!

We can see how the phase-space dsitribution evolves by using the Wigner quasi-probability distribution. There is a nice visualization on Wikipedia here here.