Solving the time dependent Schrodinger equation

$$\hbar i \frac{\partial \Psi(\vec{x},t)}{\partial t} = H\Psi(\vec{x},t), \quad H = -\frac{\hbar^2}{2m}\nabla^2 + V(\vec{x})$$

This has the formal solution

$$\Psi(\vec{x},t) = \mathrm{e}^{-itH/\hbar}\Psi(\vec{x},0)$$

For an eigenstate the time dependence is just a phase

$$\Psi_n(\vec{x},t) = \mathrm{e}^{-itE_n/\hbar}\Psi_n(\vec{x})$$

We can expand an arbitrary state in eigenstates, and thus

$$\Psi(\vec{x},t) = \sum_n C_n \mathrm{e}^{-itE_n/\hbar}\Psi_n(\vec{x})$$

But to use this we need to know all the eigenstates.
What to do if this is not possible?

Split-operator (Trotter approximation) method

Time evolution operator

$$U(t) = \mathrm{e}^{-iHt/\hbar} = \mathrm{e}^{-i(K+V)t/\hbar} \neq \mathrm{e}^{-iKt/\hbar}\mathrm{e}^{-iVt/\hbar}$$

For any operators A, B:

$$\mathrm{e}^{A+B} = \lim_{n\to\infty}\left(\mathrm{e}^{A/n}\mathrm{e}^{B/n}\right)^n$$

If we use this formula for large but finite n, there is a small error

$$\mathrm{e}^{A+B} = \left(\mathrm{e}^{A/n}\mathrm{e}^{B/n}\right)^n + [A,B]O(\Delta_t)^2$$

Introduce time step in time evolution

$$U(t) \approx \left(\mathrm{e}^{-iK\Delta_t/\hbar}\mathrm{e}^{-iV\Delta_t/\hbar}\right)^n + O(\Delta_t)^2$$

Why is this useful?

Only diagonal operations (multiplications) if we switch back and forth between real space and momentum space wave functions

The potential energy is diagonal in real space:

$$e^{-iV\Delta_t/\hbar}|\Psi\rangle = \int d\vec{x} e^{-iV(\vec{x})\Delta_t/\hbar}\Psi(\vec{x})|\vec{x}\rangle$$

The kinetic energy is diagonal in momentum space

$$e^{-iK\Delta_t/\hbar}|\Psi\rangle = \int d\vec{p} e^{-i(p^2/2m)\Delta_t/\hbar}\Psi(\vec{p})|\vec{p}\rangle$$

If we go back and forth between real and momentum space wave functions, the time evolution is obtained just by multiplications

Fourier transforms:

$$\Psi(\vec{p}) = \int d\vec{x} e^{-i\vec{p}\cdot\vec{x}}\Psi(\vec{x})$$

$$\Psi(\vec{x}) = \int d\vec{p} e^{i\vec{p}\cdot\vec{x}}\Psi(\vec{p})$$

We need to calculate a series of many Fourier integrals

How can the FT be carried out efficiently?

One dimension: discrete Fourier transform in periodic box

$$x = n\Delta_x, \quad \Delta_x = L/N, \quad n = -N/2+1, \ldots, N/2$$

$$k = m\Delta_k, \quad \Delta_k = 2\pi/L, \quad m = -N/2+1, \ldots, N/2$$

$$f(m\Delta_k) = \frac{1}{\sqrt{N}} \sum_n \mathrm{e}^{-i2\pi mn/N} g(n\Delta_x)$$

$$g(n\Delta_x) = \frac{1}{\sqrt{N}} \sum_m \mathrm{e}^{-i2\pi mn/N} f(m\Delta_k)$$

It looks like these transforms each require N^2 operations

Fast Fourier transform (FFT): only Nlog(N) operations

Read about how the FFT works (e.g., Numerical Recipes)
Available in Julia in package FFTW

Basic FFT functions available in FFTW

`FT: f=fft(g)`

$$f(m) = \sum_{n=1}^{N} \mathrm{e}^{-inm2\pi/N} g(n)$$

`Inverse FT: g=ifft(f)`

$$g(n) = \frac{1}{N} \sum_{m=1}^{N} \mathrm{e}^{+inm2\pi/N} f(m)$$

Not using conventional physics normalization
-i f initially is normalized to 1, g will be normalized to N
-but after the inverse transform, f is still normalized to 1

Note that the summation over n is for n=1,...,N.
The function is periodic:
n=N/2+1,...,N can correspond to negative function arguments.

Propagation of a Gaussian wave packet

Momentum space wave function corresponding to a Gaussian wave packet centered at x=0 with average momentum $\mathbf{k_0}$, width **a**

$$\Psi(k) \propto \mathrm{e}^{-a^2(k-k_0)^2/2} \;\rightarrow\; \Psi(x) \propto \mathrm{e}^{-x^2/a^2 - ik_0 x}$$

Start in momentum space (packet around x=0, momentum >0)

$$\Psi(k_m) \propto \mathrm{e}^{-a^2(k_m-k_0)^2/2},$$

$$k_m = m\Delta_k, \quad \Delta_k = 2\pi/L, \quad m = -N/2+1, \ldots, N/2$$

- nn x-points, box from –ll to ll (length L = 2*ll)

```
psi=Vector{ComplexF64}(undef,nn)
dk=pi/ll
for i=-div(nn,2)+1:div(nn,2)
    k=i*dk
    i>0 ? j=i : j=i+nn
    psi[j]=exp(-(a0*(k-k0)/2)^2)
end
psi=ifft(psi)
psi=normalize(nn,ll,psi)
```

Prepare time evolution operators:
- time-step `dt`,
- potential implemented in a function

```
vdt=Vector{ComplexF64}(undef,nn)
kdt=Vector{ComplexF64}(undef,nn)
dx::Float64=2*ll/nn
dk::Float64=2*pi/(2*ll)
for i=-div(nn,2)+1:div(nn,2)
    x::Float64=i*dx
    k::Float64=i*dk
    i>0 ? j=i : j=i+nn
    vdt[j]=exp(-dt*im*potential(x,ll))
    kdt[j]=exp(-dt*im*0.5*k^2)
end
```

Fourier transform psi to real space:

Evolve with potential factor: `psi .= psi .* vdt`

Fourier transform psi to momentum space:

Evolve with kinetic factor: `psi .= psi .* kdt`

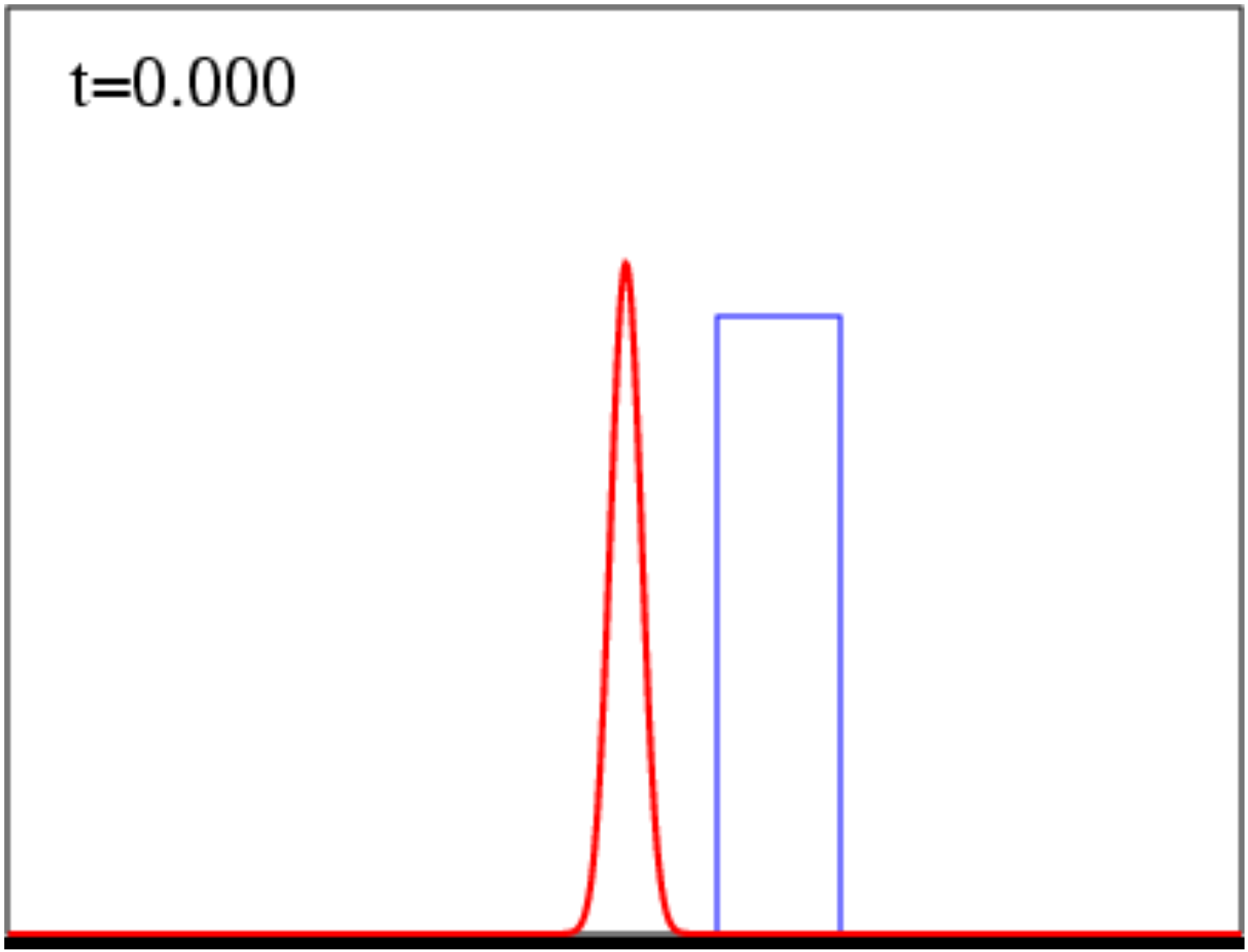
```
function evolvestep(vdt,kdt,psi)
    psi .*= vdt
    psi=fft(psi)
    psi .*= kdt
    psi=ifft(psi)
    return psi
 end
```

Repeat as many times as desired

Convergence checks
- as a function of space/momentum discretization
- as a function of time-step

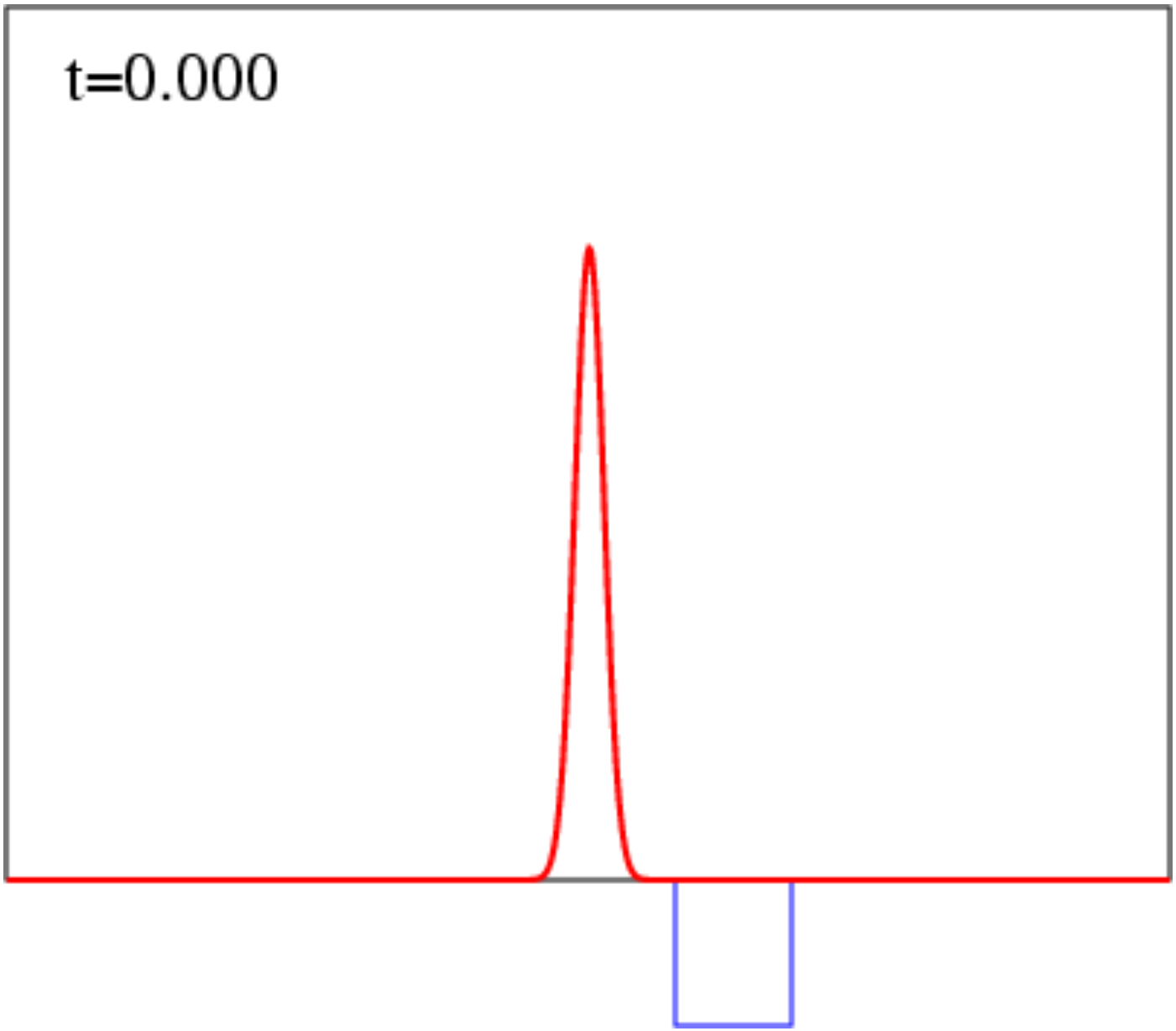
Wave packet incident on repulsive square potential barrier

L=40, N=1024, V=50 for 3 < x<7, a=1, k_0=10, Δ_t=0.001



Wave packet incident on attractive square potential barrier

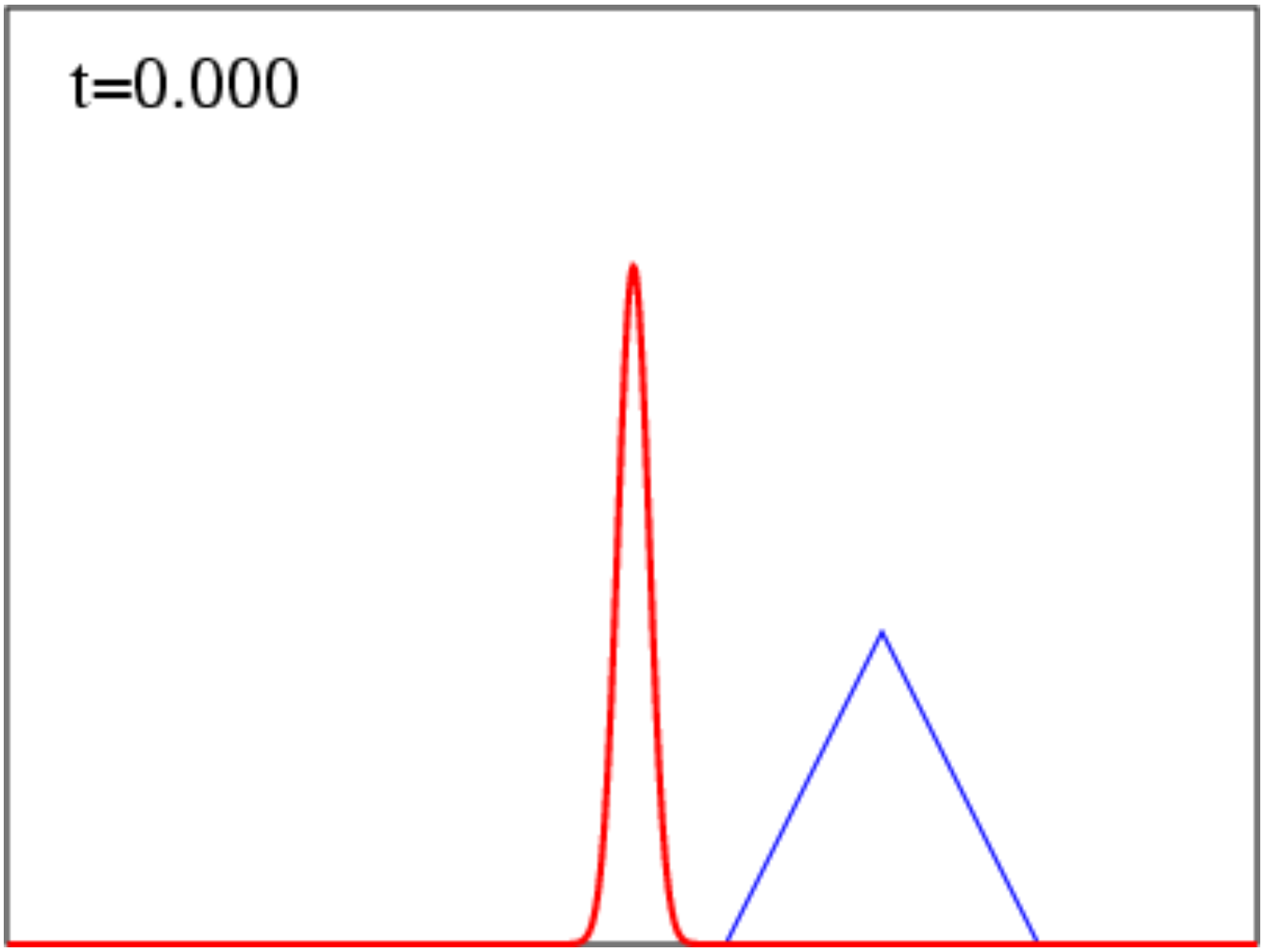
L=40, N=1024, V=-50 for 3 < x<7, a=1, k_0=10, Δ_t=0.001



Wave packet incident on a repulsive triangular potential barrier

L=40, N=1024, a=1, k_0=10, Δ_t=0.001

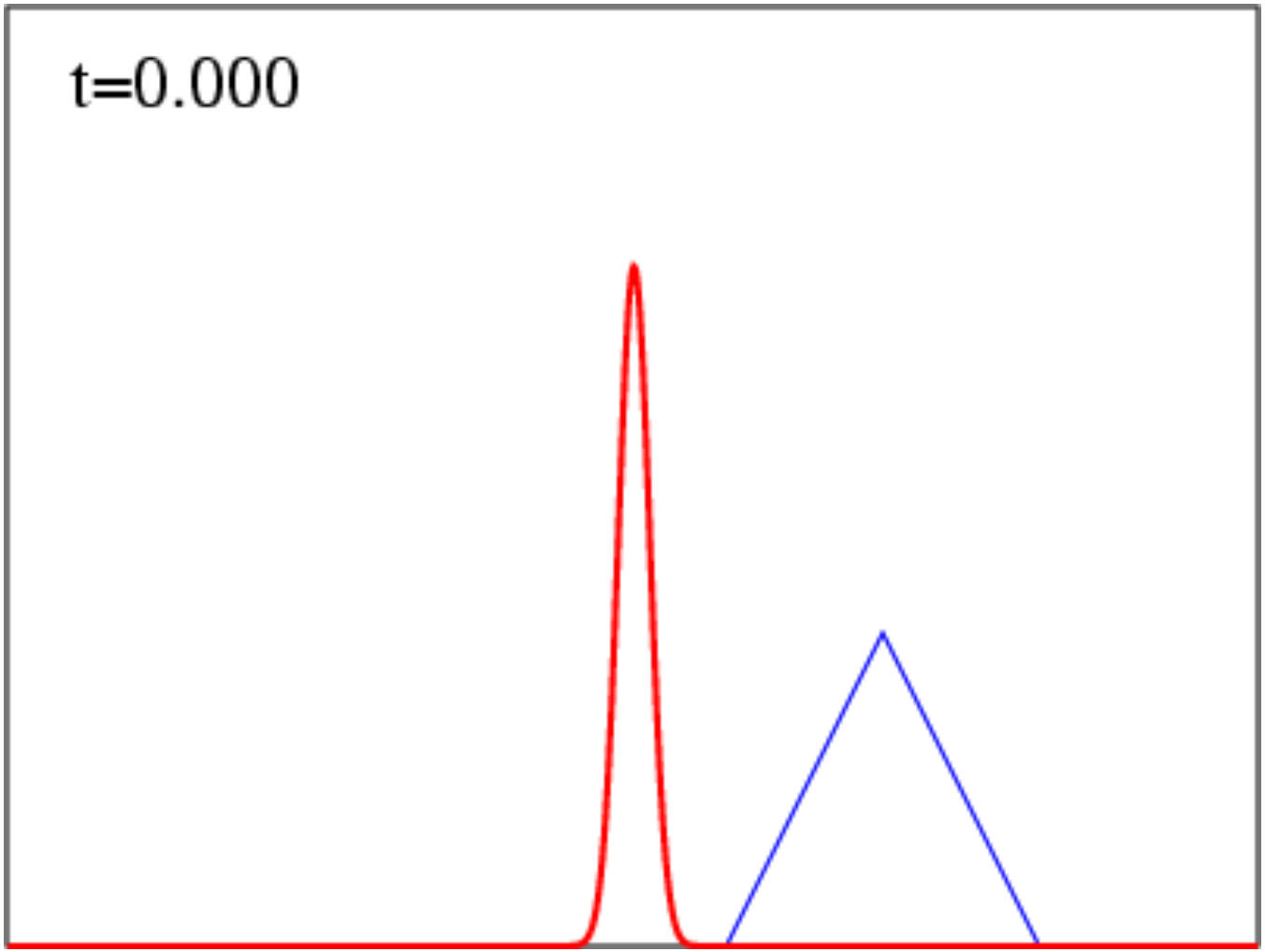
potential between x=3 and x=13, Vmax=10



Wave packet incident on a repulsive triangular potential barrier

$L=40, N=1024, k_0=10, \Delta_t=0.001$

potential between x=3 and x=13, Vmax=5



Wave packet incident on a repulsive triangular potential barrier

L=40, N=1024, V=-50 for 3 < x<7, a=1, k_0=10, Δ_t=0.001

potential between x=3 and x=13, Vmax=7.5