Institut für Theoretische
Physik
Universität zu Köln

Prof. Dr. Simon Trebst
Jan Attig

# Computerphysik
# Vorlesung — Programmiertechniken
## Streuung am Potentialtopf

### Sommersemester 2019

**Website:** http://www.thp.uni-koeln.de/trebst/Lectures/2019-CompPhys.shtml (http://www.thp.uni-koeln.de/trebst/Lectures/2019-CompPhys.shtml)

## 0. Fourier-Trafo

In [ ]:
```
]add FFTW
```

## 1. Zeitaufgelöste Streuung an einem Potential

In [1]:
```
# Pakete einbinden
using PyCall
using PyPlot
using LinearAlgebra
using FFTW
pygui(true);
```

In [2]:
```
# Diskretisierungen
# h_bar = m = 1.0 per Definition

dt = 0.01

N = 2^11
dx = 0.1
xs = dx .* (collect(1:N) .- 0.5*N)

dk = 2 * pi / (N*dx)
ks = -0.5 * N * dk .+ dk .* collect(1:N);
```

In [3]:
```
# Definition des Potential
function set_potential(V0, V_width, V_distance=0)
    V = zeros(length(xs))

    ### single potential
    if(V_distance == 0)
        for i in 1:N
            if( abs(xs[i])<V_width/2 ) V[i] = V0 end
        end
    end

    ### two potentials
    if(V_distance != 0)
        for i in 1:N
            if( abs(xs[i])<V_distance/2 + V_width && abs(xs[i])>V_dist
        end
    end

    return V
end
```

Out[3]: set_potential (generic function with 2 methods)

In [4]:
```
# Definition der Wellenfuntion
function set_wavefunction(k0)
    # Gaussian
    sigma = 12.0 / k0

    # "linkes" Wellenpaket
    x0 = -30
    psi_x0 = (1/sqrt((sigma * sqrt(pi))) .* exp.(complex.(-0.5.*((xs .

    # "rechtes" Wellenpaket
    # x1 = 30
    # psi_x1 = (1/sqrt((sigma * sqrt(pi))) .* exp.(complex.(-0.5.*((xs

    return psi_x0 # + psi_x1
end
```

Out[4]: set_wavefunction (generic function with 1 method)

In [5]:
```
# Hauptroutine
function scatter(psi_x0, V)
    # display
    gcf().clf()
    gcf().set_facecolor("lightgray")
    axis("off")

    # limits
    ylim([-0.03,0.28])
    xlim([-100,100])

    line_plot = plot(xs, zeros(length(xs)), color="black", linewidth=3
    V_plot    = plot(xs, V,                 color="blue", linewidth=1)
    psi_plot  = plot(xs, psi_x0,            color="#850000", linewidth

    # discrete Fourier transformation
    psi_mod_x = psi_x0 .* exp.(complex.(0, -ks[1] .* xs)) .* dx ./ (sq

    ### core iteration
    step  = 1
    while step < 2000 && (abs.(psi_x0).^2)[1] < 0.001 && (abs.(psi_x0)
        # update step
        for j in 1:20
            psi_mod_x .*= exp.( complex.(0, -0.5*V*dt))
            psi_mod_k = fft(psi_mod_x)
            psi_mod_k .*= exp.( complex.(0, -0.5 * (ks .* ks) * dt))
            psi_mod_x = ifft(psi_mod_k)
            psi_mod_x .*= exp.( complex.(0, -0.5*V*dt))
        end

        # discrete Fourier transformation
        psi_x0 = psi_mod_x .* exp.(complex.(0, ks[1] .* xs)) .* sqrt(2

        # update plot
        psi_plot.set_data(xs, abs.(psi_x0).^2)

        # delay for plotting
        pause(0.001)
        step += 1
    end
end
```

Out[5]:
```
scatter (generic function with 1 method)
```

In [6]:
```
k0 = 1.5
V0 = 1.0
V_width = 5.0
V_distance = 0.0

V = set_potential(V0, V_width,V_distance)
psi_x0 = set_wavefunction(k0)

scatter(psi_x0, V)
```

In [ ]:
```

```