```julia
using LinearAlgebra, DSP, Plots, LaTeXStrings
Σ = sum

N = 200 # number of samples
𝐑 = I(2) # correlation matrix
𝐩 = [1, 1.6] # cross-correlation vector between the desired and input signals
𝐱 = randn(N) # input vector
𝐡 = [1, 1.6] # filter coefficients
μ = .1

𝐝 = rand(N)
for n ∈ 2:N
    𝐱₍ₙ₎ = [𝐱[n], 𝐱[n-1]] # input vector at the instant n
    𝐝[n] = 𝐡 ⋅ 𝐱₍ₙ₎ # d(n)
end

# steepest descent
𝐰₍ₙ₎ = rand(2) # initial guess of the coefficient vector
𝐲 = rand(N) # output signal
𝔼e² = zeros(N) # error signal
for n ∈ 2:N
    𝐱₍ₙ₎ = [𝐱[n], 𝐱[n-1]] # input vector at the instant n
    𝐲[n] = 𝐰₍ₙ₎ ⋅ 𝐱₍ₙ₎ # y(n)
    𝔼e²[n] = ((n-2)*𝔼e²[n-1] + (𝐝[n] - 𝐲[n])^2)/(n-1)
    𝐠₍ₙ₎ = -2𝐩 + 2𝐑*𝐰₍ₙ₎ # deterministic gradient
    global 𝐰₍ₙ₎ -= μ*𝐠₍ₙ₎
end
p1 = plot([𝐲 𝐝], title="Steepest descent", label=[L"\mathbf{w}(n) = \mathbf{w}(n) - \mu \mathbf{g}(n)" L"d(n)"])
e1 = plot(𝔼e², title="MSE of the Steepest descent", label=L"\mathbb{E}[e^2(n)]")

# Newton's algorithm
𝐰₍ₙ₎ = rand(2) # initial guess of the coefficient vector
𝐲 = rand(N) # output signal
𝔼e² = zeros(N) # error signal
𝐇 = 2𝐑 # the Hessian
for n ∈ 2:N
    𝐱₍ₙ₎ = [𝐱[n], 𝐱[n-1]] # input vector at the instant n
    𝐲[n] = 𝐰₍ₙ₎ ⋅ 𝐱₍ₙ₎ # y(n)
    𝔼e²[n] = ((n-2)*𝔼e²[n-1] + (𝐝[n] - 𝐲[n])^2)/(n-1)
    𝐠₍ₙ₎ = -2𝐩 + 2𝐑*𝐰₍ₙ₎ # deterministic gradient
    Δ𝐰₍ₙ₊₁₎ = -inv(𝐇)*𝐠₍ₙ₎
    global 𝐰₍ₙ₎ += Δ𝐰₍ₙ₊₁₎
end
p2 = plot([𝐲 𝐝], title="Newton's algorithm" , label=[L"\mathbf{w}(n) = \mathbf{w}(n) + \mu \mathbf{H}^{-1}\mathbf{g}(n)" L"d(n)"])
e2 = plot(𝔼e², title="MSE of the Newton's algorithm", label=L"\mathbb{E}[e^2(n)]")

# Least-Mean Squares (LMS) algorithm
𝐰₍ₙ₎ = rand(2) # initial guess of the coefficient vector
𝐲 = rand(N) # output signal
𝔼e² = zeros(N) # error signal
for n ∈ 2:N
    𝐱₍ₙ₎ = [𝐱[n], 𝐱[n-1]] # input vector at the instant n
    𝐲[n] = 𝐰₍ₙ₎ ⋅ 𝐱₍ₙ₎ # y(n)
    e₍ₙ₎ = 𝐝[n] - 𝐲[n]
```

```julia
    𝔼e²[n] = ((n-2)*𝔼e²[n-1] + e₍ₙ₎^2)/(n-1)
    𝐠̂₍ₙ₎ = -2e₍ₙ₎*𝐱₍ₙ₎ # stochastic gradient
    global 𝐰₍ₙ₎ -= μ*𝐠̂₍ₙ₎
end
p3 = plot([𝐲 𝐝], title="LMS algorithm", label=[L"\mathbf{w}(n) = \mathbf{w}(n) + 2\mu e(n)\mathbf{x}(n)" L"d(n)"])
e3 = plot(𝔼e², title="MSE of the LMS algorithm", label=L"\mathbb{E}[e^2(n)]")

# normalized LMS algorithm
𝐰₍ₙ₎ = rand(2) # initial guess of the coefficient vector
γ = 1
𝐲 = rand(N) # output signal
𝔼e² = zeros(N) # error signal
for n ∈ 2:N
    𝐱₍ₙ₎ = [𝐱[n], 𝐱[n-1]] # input vector at the instant n
    𝐲[n] = 𝐰₍ₙ₎ ⋅ 𝐱₍ₙ₎ # y(n)
    e₍ₙ₎ = 𝐝[n] - 𝐲[n]
    𝔼e²[n] = ((n-2)*𝔼e²[n-1] + e₍ₙ₎^2)/(n-1)
    𝐠̂₍ₙ₎ = -2e₍ₙ₎*𝐱₍ₙ₎ # stochastic gradient
    global 𝐰₍ₙ₎ -= μ*𝐠̂₍ₙ₎/(𝐱₍ₙ₎⋅𝐱₍ₙ₎ + γ)
end
p4 = plot([𝐲 𝐝], title="Normalized LMS algorithm", label=[L"\mathbf{w}(n) = \mathbf{w}(n) + \frac{2\mu e(n)\mathbf{x}(n)}{\mathbf{x}^\mathrm{T}(n)\mathbf{x}(n) + \gamma}" L"d(n)"])
e4 = plot(𝔼e², title="MSE of the normalized LMS algorithm", label=L"\mathbb{E}[e^2(n)]")

fig = plot(p1, p2, p3, p4, layout=(4,1), size=(1200,800))
savefig(fig, "figs/q4.png")

fig = plot(e1, e2, e3, e4, layout=(4,1), size=(1200,800))
savefig(fig, "figs/q4-error-evolution.png")
```