

```

1 using LinearAlgebra, DSP, Plots, LaTeXStrings
2 Σ = sum
3
4 N = 200 # number of samples
5 R = I(2) # correlation matrix
6 p = [1, 1.6] # cross-correlation vector between the desired and input
   signals
7 x = randn(N) # input vector
8 h = [1, 1.6] # filter coefficients
9 μ = .1
10
11 d = rand(N)
12 for n ∈ 2:N
13     x(n) = [x[n], x[n-1]] # input vector at the instant n
14     d[n] = h · x(n) # d(n)
15 end
16
17 # steepest descent
18 w(n) = rand(2) # initial guess of the coefficient vector
19 y = rand(N) # output signal
20 Ee² = zeros(N) # error signal
21 for n ∈ 2:N
22     x(n) = [x[n], x[n-1]] # input vector at the instant n
23     y[n] = w(n) · x(n) # y(n)
24     Ee²[n] = ((n-2)*Ee²[n-1] + (d[n] - y[n])²)/(n-1)
25     g(n) = -2p + 2R*w(n) # deterministic gradient
26     global w(n) -= μ*g(n)
27 end
28 p1 = plot([y d], title="Steepest descent", label=[L"\mathbf{w}(n) =
   \mathbf{w}(n) - \mu \mathbf{g}(n)" L"d(n)"])
29 e1 = plot(Ee², title="MSE of the Steepest descent", label=L"\mathbb{E}
   [e²(n)]")
30
31 # Newton's algorithm
32 w(n) = rand(2) # initial guess of the coefficient vector
33 y = rand(N) # output signal
34 Ee² = zeros(N) # error signal
35 H = 2R # the Hessian
36 for n ∈ 2:N
37     x(n) = [x[n], x[n-1]] # input vector at the instant n
38     y[n] = w(n) · x(n) # y(n)
39     Ee²[n] = ((n-2)*Ee²[n-1] + (d[n] - y[n])²)/(n-1)
40     g(n) = -2p + 2R*w(n) # deterministic gradient
41     Δw(n+1) = -inv(H)*g(n)
42     global w(n) += Δw(n+1)
43 end
44 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)"])
45 e2 = plot(Ee², title="MSE of the Newton's algorithm", label=L"\mathbb{E}
   [e²(n)]")
46
47 # Least-Mean Squares (LMS) algorithm
48 w(n) = rand(2) # initial guess of the coefficient vector
49 y = rand(N) # output signal
50 Ee² = zeros(N) # error signal
51 for n ∈ 2:N
52     x(n) = [x[n], x[n-1]] # input vector at the instant n
53     y[n] = w(n) · x(n) # y(n)
54     e(n) = d[n] - y[n]

```

```

55     Ee²[n] = ((n-2)*Ee²[n-1] + e_(n)²)/(n-1)
56     ĝ_(n) = -2e_(n)*x_(n) # stochastic gradient
57     global w_(n) -= μ*ĝ_(n)
58 end
59 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)"])
60 e3 = plot(Ee², title="MSE of the LMS algorithm", label=L"\mathbb{E}
[e²(n)]")
61
62 # normalized LMS algorithm
63 w_(n) = rand(2) # initial guess of the coefficient vector
64 y = 1
65 y = rand(N) # output signal
66 Ee² = zeros(N) # error signal
67 for n ∈ 2:N
68     x_(n) = [x[n], x[n-1]] # input vector at the instant n
69     y[n] = w_(n) · x_(n) # y(n)
70     e_(n) = d[n] - y[n]
71     Ee²[n] = ((n-2)*Ee²[n-1] + e_(n)²)/(n-1)
72     ĝ_(n) = -2e_(n)*x_(n) # stochastic gradient
73     global w_(n) -= μ*ĝ_(n)/(x_(n)·x_(n) + γ)
74 end
75 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)"])
76 e4 = plot(Ee², title="MSE of the normalized LMS algorithm",
label=L"\mathbb{E}[e²(n)]")
77
78 fig = plot(p1, p2, p3, p4, layout=(4,1), size=(1200,800))
79 savefig(fig, "figs/q4.png")
80
81 fig = plot(e1, e2, e3, e4, layout=(4,1), size=(1200,800))
82 savefig(fig, "figs/q4-error-evolution.png")

```