POLITECNICO DI MILANO

Adaptive filtering: the steepest descent algorithm

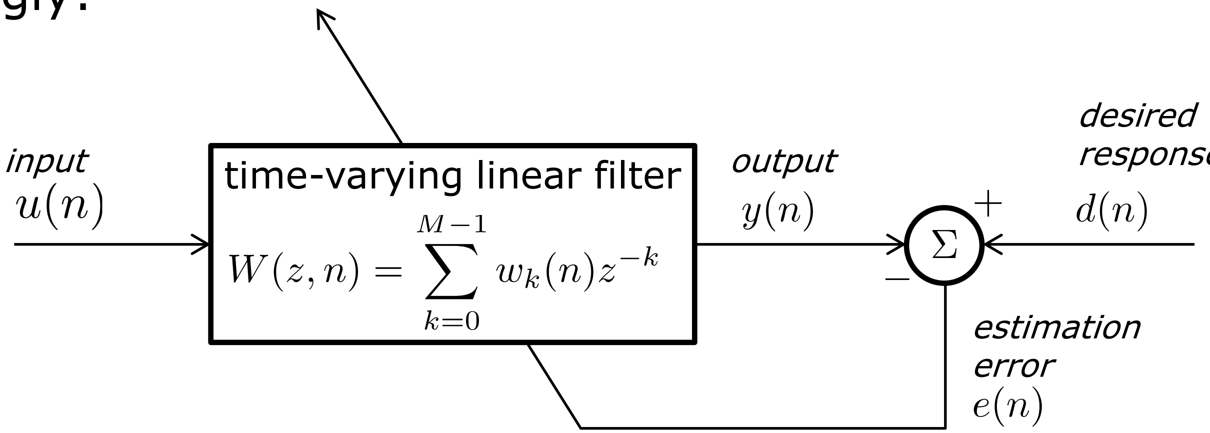
Antonio Canclini
Augusto Sarti

Motivations

- Solving the Wiener-Hopf equations may be computationally inefficient
  - matrix inversion is involved, complexity is $O(M^3)$
  - it may happen that the auto-correlation matrix is almost singular, thus causing numerical issues in the inversion

- Adaptive algorithms can be used to overcome these problems

- The error signal is used in closed-loop as an instantaneous measurement of the performance, and the filter taps are adjusted accordingly:



Overview

- The steepest descent method is a gradient-based adaptation technique
- It is an iterative algorithm that, starting from some initial (arbitrary) value for the tap-weight vector, improves with the increased number of iterations
- When the input signal and the desired response are stationary, the method of steepest descent converges to the optimum Wiener solution
- Procedure:
  1. Start with an initial guess of the filter coefficients, $\mathbf{w}(n=0)$
  2. Compute the gradient of the cost function $\boldsymbol{\nabla} J[w(n)]$
  3. Update the filter coefficients following the steepest descent of the gradient
  4. Go back to step 2 and repeat

Notation

- Differently from the classical Wiener filtering method, which provides a closed form solution, the steepest descent provides a time-varying solution
- Therefore, we stress the dependency on time of the involved quantities:

input vector $$\mathbf{u}(n) = [u(n), u(n-1), \dots u(n-M+1)]^T$$

tap-weight vector $$\mathbf{w}(n) = [w_0(n), w_1(n), \dots w_{M-1}(n)]^T$$

cost function $$J(n) = \sigma_d^2 - \mathbf{w}^H(n)\mathbf{p} - \mathbf{p}^H\mathbf{w}(n) + \mathbf{w}^H(n)\mathbf{R}\mathbf{w}(n)$$

- Note that, if the tap-input vector and the desired response are jointly stationary, then the cross-correlation vector $\mathbf{p}$ and the auto-correlation matrix $\mathbf{R}$ are constant, and the cost function takes the form in the above expression

The algorithm

1. Begin with an initial value $\mathbf{w}(0)$ for the tap-weight vector
   - it provides an initial guess as to where the minimum point of the error-performance surface may be located; unless some prior knowledge is available, $\mathbf{w}(0)$ is usually set to the null vector

2. Using the current guess $\mathbf{w}(n)$ of the tap-weights, compute the gradient vector

$$\nabla J(n) = \begin{bmatrix} \frac{\partial J(n)}{\partial a_0(n)} + j\frac{\partial J(n)}{\partial b_0(n)} \\ \frac{\partial J(n)}{\partial a_1(n)} + j\frac{\partial J(n)}{\partial b_1(n)} \\ \vdots \\ \frac{\partial J(n)}{\partial a_{M-1}(n)} + j\frac{\partial J(n)}{\partial b_{M-1}(n)} \end{bmatrix} = -2\mathbf{p} + 2\mathbf{R}\mathbf{w}(n)$$

3. Compute the next guess at the tap-weight vector, following the direction opposite to that of the gradient vector (i.e., steepest descent of the error-performance surface):

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{1}{2}\mu[-\nabla J(n)]$$

4. Repeat the process from step 2

The algorithm

- The update equation can be rewritten as

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu[\mathbf{p} - \mathbf{R}\mathbf{w}(n)], \quad n = 0, 1, 2, \ldots$$

- The parameter μ is a positive real-valued constant, which controls the size of the incremental correction applied to the tap-weights
  - we refer to μ as the **step-size parameter**

- We also observe that $\mathbf{p} - \mathbf{R}\mathbf{w}(n) = E\{\mathbf{u}(n)e^*(n)\}$

- Note that the update rule involves a feedback loop; what about stability of the algorithm?

Stability of steepest descent method

- The feedback loop is given by the term $\mu\mathbf{R}\mathbf{w}(n)$, thus the stability performance is determined by two factors:
  - the step-size parameter μ
  - the correlation matrix $\mathbf{R}$

- To determine the condition for the stability, we examine the **natural modes of the algorithm**, i.e. we exploit the canonical form of the cost function

- We define the **weight-error vector** at time n as

$$\mathbf{c}(n) \triangleq \mathbf{w}(n) - \mathbf{w}_o$$

where $\mathbf{w}_o$ is the optimum filter (i.e., the Wiener solution)

- After some math, we can rewrite the update rule for the weight-error vector as

$$\mathbf{c}(n+1) = (\mathbf{I} - \mu\mathbf{R})\,\mathbf{c}(n)$$

Stability of steepest descent method

- Recall the eigenvalue decomposition of the auto-correlation matrix

$$\mathbf{R} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^H \,, \qquad \mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_M \end{bmatrix}$$

the auto-correlation matrix is guaranteed to be positive definite, implying that $\lambda_1, \lambda_2, \ldots \lambda_M \in \mathbb{R}^+$

- Recalling also the definition of the transformed tap-weights

$$\begin{aligned} \mathbf{v}(n) &\triangleq \mathbf{Q}^H \left[\mathbf{w}(n) - \mathbf{w}_o\right] \\ &= \mathbf{Q}^H \mathbf{c}(n) \end{aligned}$$

we can rewrite again the updating rule, obtaining

$$\mathbf{v}(n+1) = \left(\mathbf{I} - \mu\mathbf{\Lambda}\right)\mathbf{v}(n)$$

- The initial value of $\mathbf{v}(n)$ is given by $\mathbf{v}(0) = \mathbf{Q}^H \left[\mathbf{w}(0) - \mathbf{w}_o\right] = -\mathbf{Q}^H \mathbf{w}_o$, assuming that $\mathbf{w}(0) = [0, 0, \ldots, 0]^T$

Stability of steepest descent method

- As the update rule of $\mathbf{v}(n)$ involves a diagonal matrix, we can treat each component separately:

$$v_k(n+1) = (1 - \mu\lambda_k)v_k(n) \quad k = 1, 2, \ldots, M$$

this equation describes the kth natural mode of the steepest descent algorithm

- Knowing the initial value $v_k(0)$, we readily obtain the solution

$$v_k(n) = (1 - \mu\lambda_k)^n v_k(0) \quad k = 1, 2, \ldots, M$$

  - all the eigenvalues are positive and real, implying no oscillations in the response of $v_k(n)$
  - the equation represent a geometric series (geometric ratio $1 - \mu\lambda_k$), which is convergent if

$$|1 - \mu\lambda_k| < 1$$

Stability of steepest descent method

- For **stability** or **convergence** of the steepest descent algorithm, the last constraint must be satisfied for all the natural modes, i.e.

$$|1 - \mu\lambda_k| < 1 \, , \quad \forall k$$

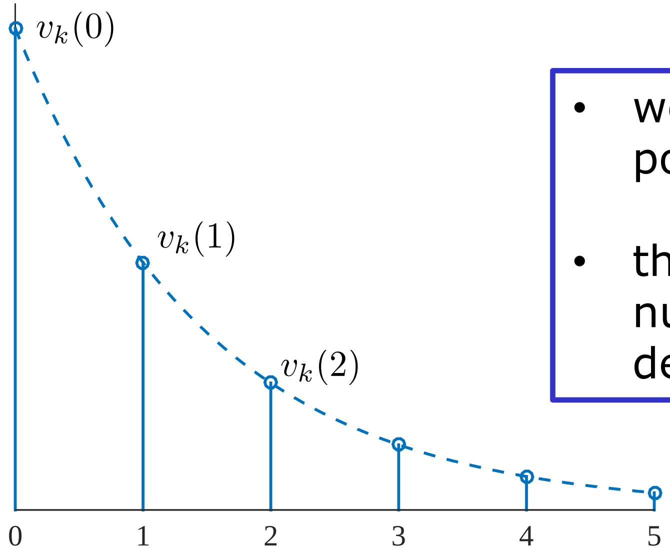
- If so, as the number of iterations approaches infinity, all the natural modes of the algorithm die out, irrespective of the initial conditions. Formally:

$$\lim_{n \to \infty} \mathbf{w}(n) = \mathbf{w}_o$$

- The **necessary and sufficient condition for the convergence** or stability is governed by the largest eigenvalue $\lambda_{\max}$

$$\boxed{0 < \mu < \frac{2}{\lambda_{\max}}}$$

- When the stability condition is satisfied, the natural modes of the algorithm exhibit an exponential decay



- we can fit an exponential envelope posing $1 - \mu\lambda_k = e^{-1/\tau_k}$
- the time constant τ_k defines the number of iterations required for decaying to $v_k(0)/e$

- Hence, the kth time constant τ_k can be expressed in terms of the related eigenvalue λ_k and the step-size parameter μ:

$$\tau_k = \frac{-1}{\ln(1 - \mu\lambda_k)} \approx \frac{1}{\mu\lambda_k} \quad \text{for} \quad \mu \ll 1$$

Transient behaviour

- We are ready to analyze the transient behaviour of the mean-squared error

$$J(n) = J_{\min} + \sum_{k=1}^{M} \lambda_k |v_k(n)|^2$$
$$= J_{\min} + \sum_{k=1}^{M} \lambda_k (1-\mu\lambda_k)^{2n} |v_k(0)|^2$$

  - The cost function consists of a sum of exponentials, each of them corresponding to a natural mode of the algorithm

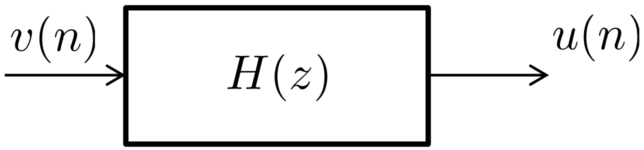
- In going from the initial value $J(0)$ to the final value $J_{\min}$, the exponential decay for the kth mode has a time constant equal to

$$\tau_{k,J} = \frac{-1}{2\ln(1-\mu\lambda_k)} \approx \frac{1}{2\mu\lambda_k}$$

- It follows than the convergence time is governed by the smaller eigenvalue $\lambda_{\min}$: the global time constant is given by

$$\boxed{\tau_J \approx \frac{1}{2\mu\lambda_{\min}}}$$

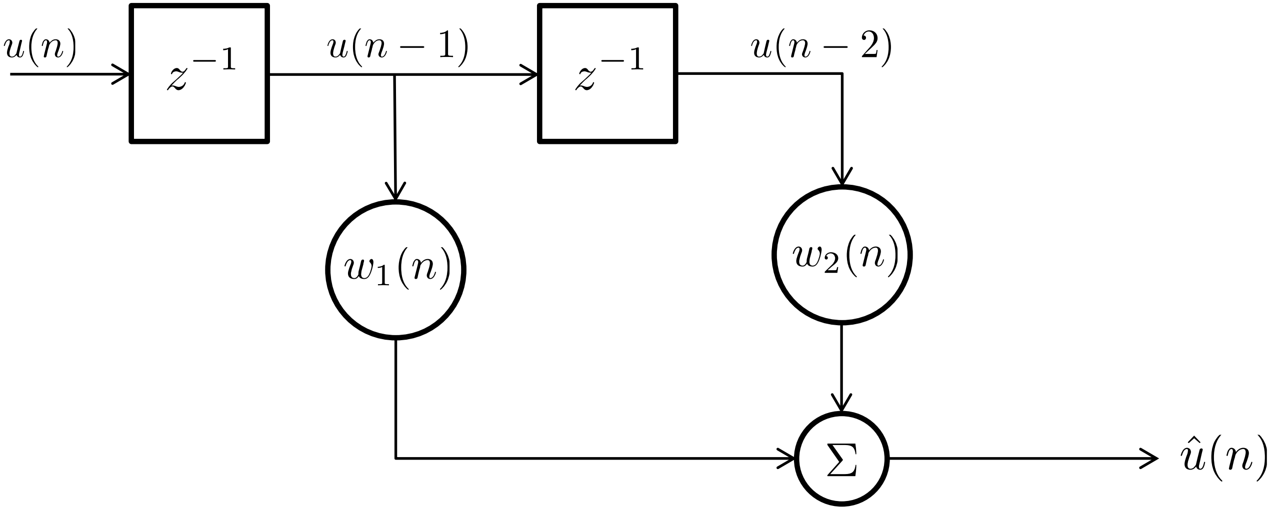
- Consider a 2nd order real-valued Auto-Regressive (AR) process that generates the input signal:



$$H(z) = \frac{1}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

$$u(n) + a_1 u(n-1) + a_2 u(n-2) = v(n)$$

- Goal: design a linear predictor of order 2, to estimate the value $u(n)$ through linear combination of $M = 2$ past samples



Final considerations

- The stability of the algorithm depends on the largest eigenvalue
- The convergence rate depends on the smaller eigenvalue
- The overall performances are governed by the ratio $\lambda_{\max}/\lambda_{\min}$, (i.e., by the conditioning number of the auto-correlation matrix $\mathbf{R}$)
- A convergence rate problem arises when the conditioning number is high (e.g., when $\lambda_{\min} \to 0$):
  - slow convergence to the optimum
  - in the limit of $\lambda_{\min} = 0$ the convergence time is infinite; this happens when the auto-correlation matrix is singular and thus not invertible

Numerical example

- Auto-correlation matrix of $u(n)$ (e.g., computed through Yule-Walker equations):

$$\mathbf{R} = \begin{bmatrix} r(0) & r(1) \\ r(1) & r(0) \end{bmatrix}$$

$$r(0) = \sigma_u^2$$

$$r(1) = -\frac{a_1}{1+a_2}\sigma_u^2$$

- Computing the eigenvalue decomposition $\mathbf{R} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^H$, we obtain the following eigenvalues:

$$\lambda_1 = \left(1 - \frac{a_1}{1+a_2}\right)\sigma_u^2$$

$$\lambda_2 = \left(1 + \frac{a_1}{1+a_2}\right)\sigma_u^2$$

- The optimum 2-taps Wiener filter corresponds to

$$\mathbf{w}_o = [w_{o1} = -a_1\, , w_{o2} = -a_2]^T$$

Numerical example

- We test the steepest descent algorithm varying some parameters, as summarized in this table:

	AR parameters		eigenvalues		eig. ratio	min. MSE
case	a_1	a_2	λ_1	λ_2	λ_1/λ_2	$J_{\min} = \sigma_v^2$
1	-0.1950	0.95	1.1	0.9	1.22	0.0965
2	-0.9750	0.95	1.5	0.5	3	0.0731
3	-1.5955	0.95	1.818	0.182	10	0.0322
4	-1.9114	0.95	1.957	0.0198	100	0.0038

- Note that σ_v^2 is tuned differently for each test, so that $\sigma_u^2 = 1$

Numerical example

- We analyze the minimization process, showing the contour lines of the error performance surface at each iteration

- Note that the canonical form of the cost function is

$$J(n) = J_{\min} + \lambda_1 v_1^2(n) + \lambda_2 v_2^2(n)$$

- Thus, the contour lines (extracted fixing the value of n) are ellipses in the plane (v_1, v_2), centered at the origin:

$$J(n) - J_{\min} = \lambda_1 v_1^2(n) + \lambda_2 v_2^2(n) \Longrightarrow$$

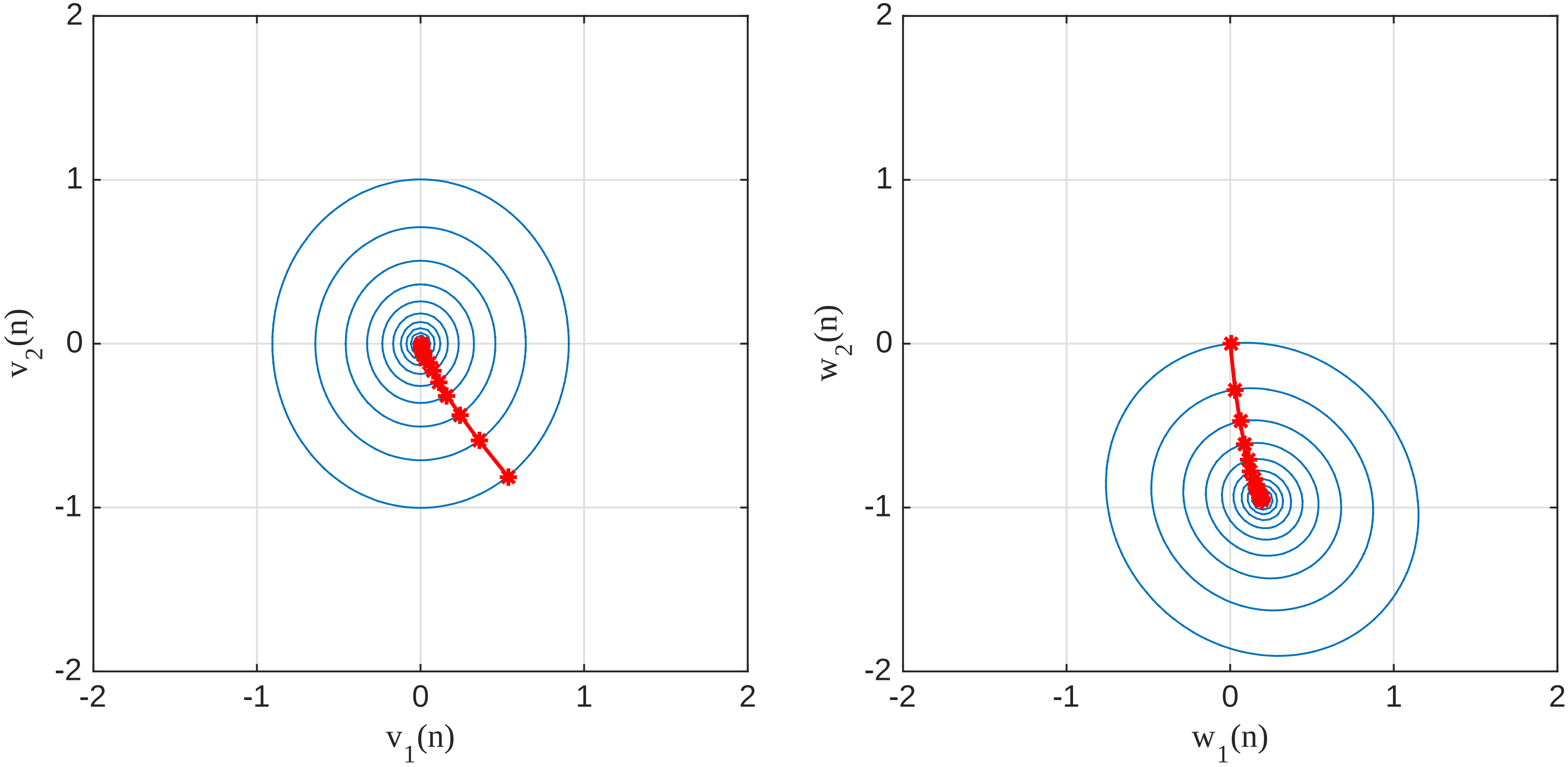
ellipse equation, with the two axis oriented as v_1 and v_2

- The pairs $v_1(0), v_2(0)\ ;\ v_1(1), v_2(1)\ ;\ \ldots v_1(n), v_2(n)$ define a trajectory in the plane (v_1, v_2), converging to the origin

- Similarly, the contour lines also in the plane (w_1, w_2) are ellipses centered at the optimum solution (w_{o1}, w_{o2})

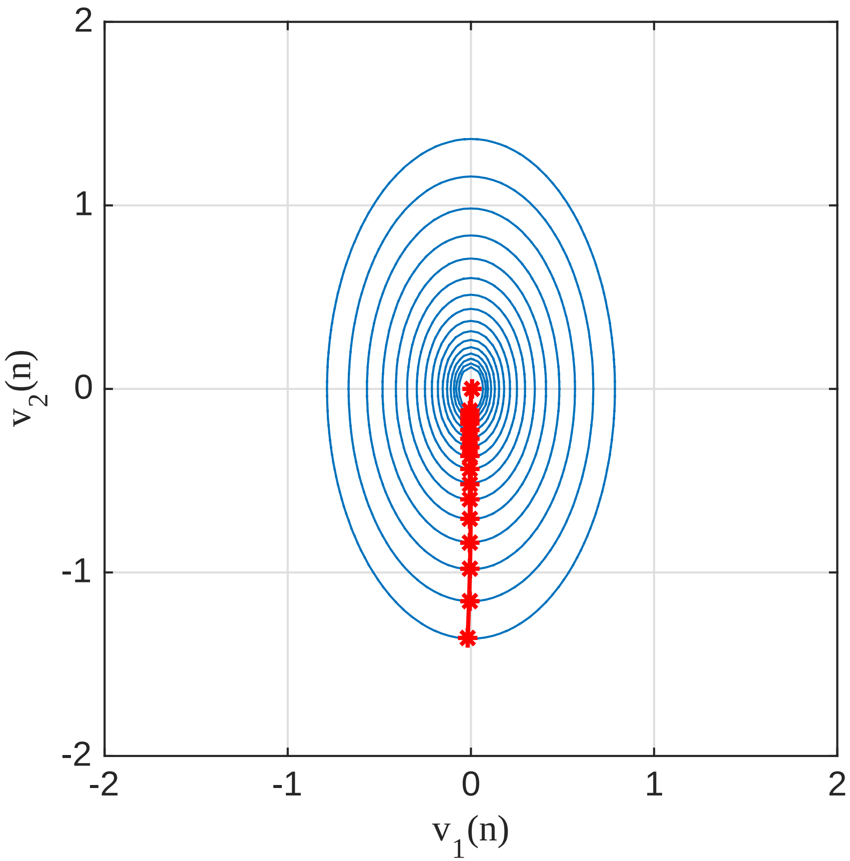
Experiment 1: varying the eigenvalue ratio

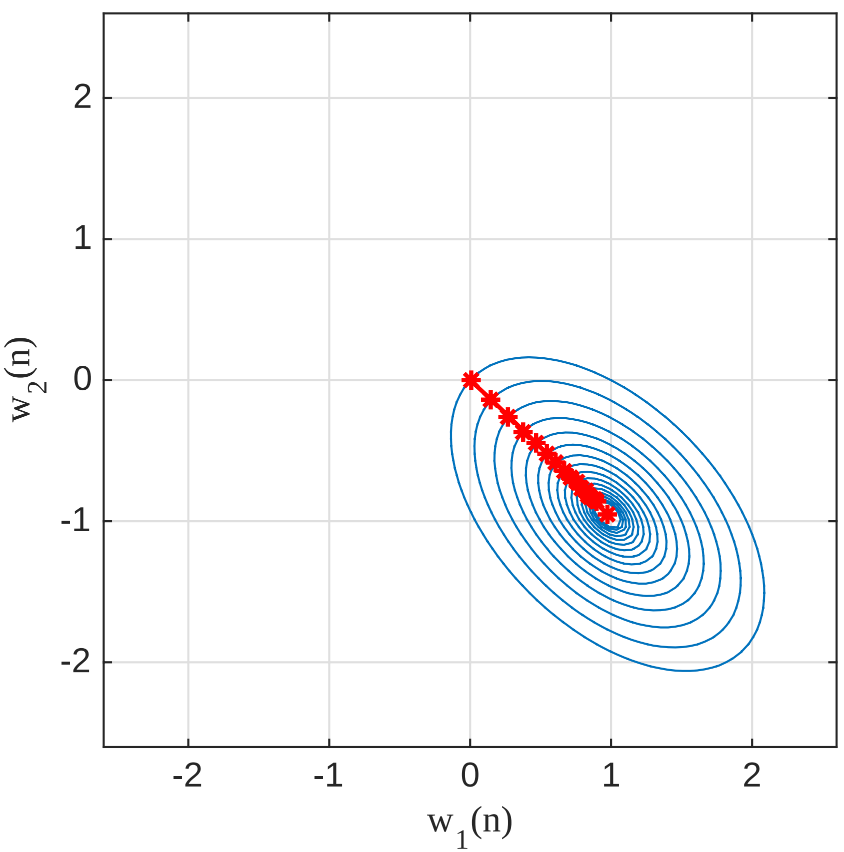
- Testing conditions:
  - $\mu = 0.3 \quad (\mu_{max} = 1.818)$
  - $\lambda_1/\lambda_2 = 1.22$



Experiment 1: varying the eigenvalue ratio

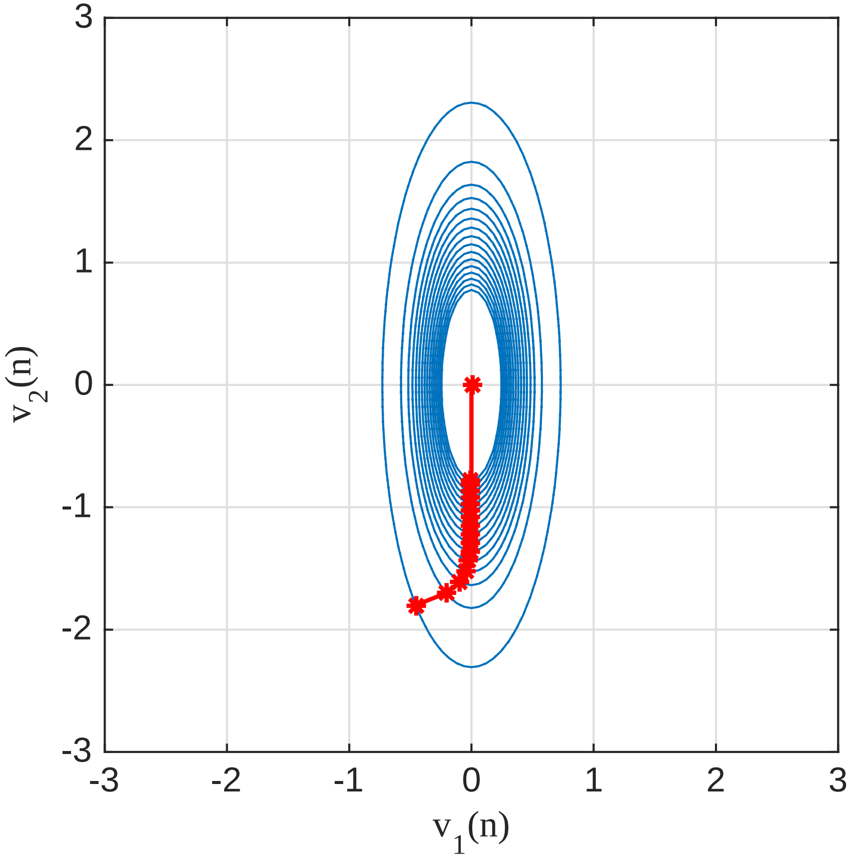
- Testing conditions:
  - $\mu = 0.3 \quad (\mu_{max} = 1.33)$
  - $\lambda_1/\lambda_2 = 3$

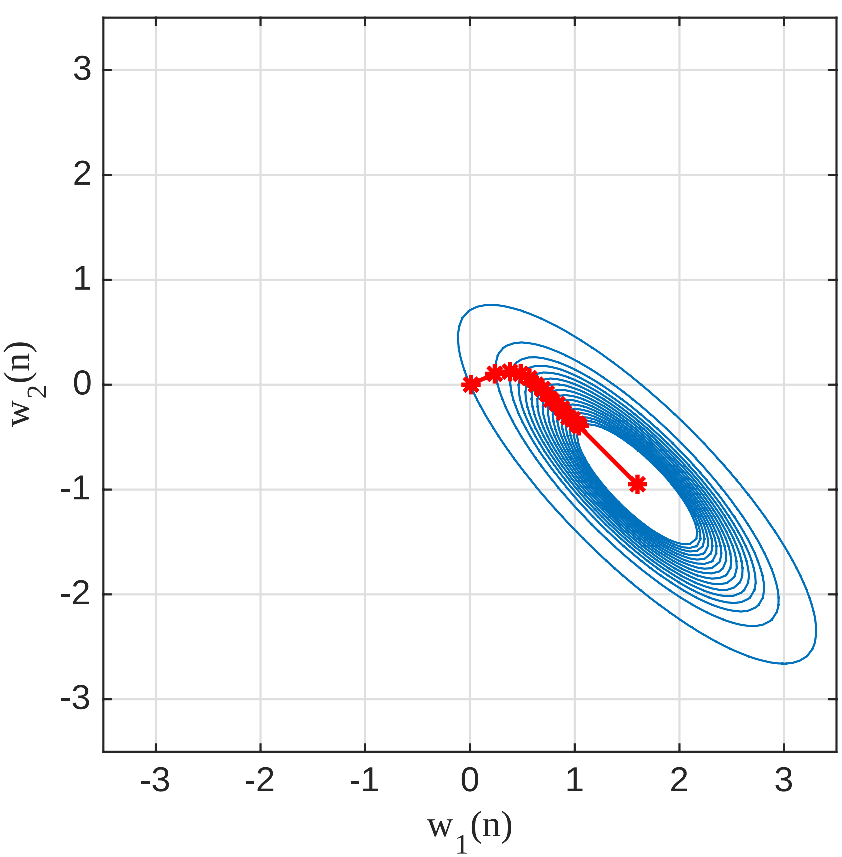




Experiment 1: varying the eigenvalue ratio

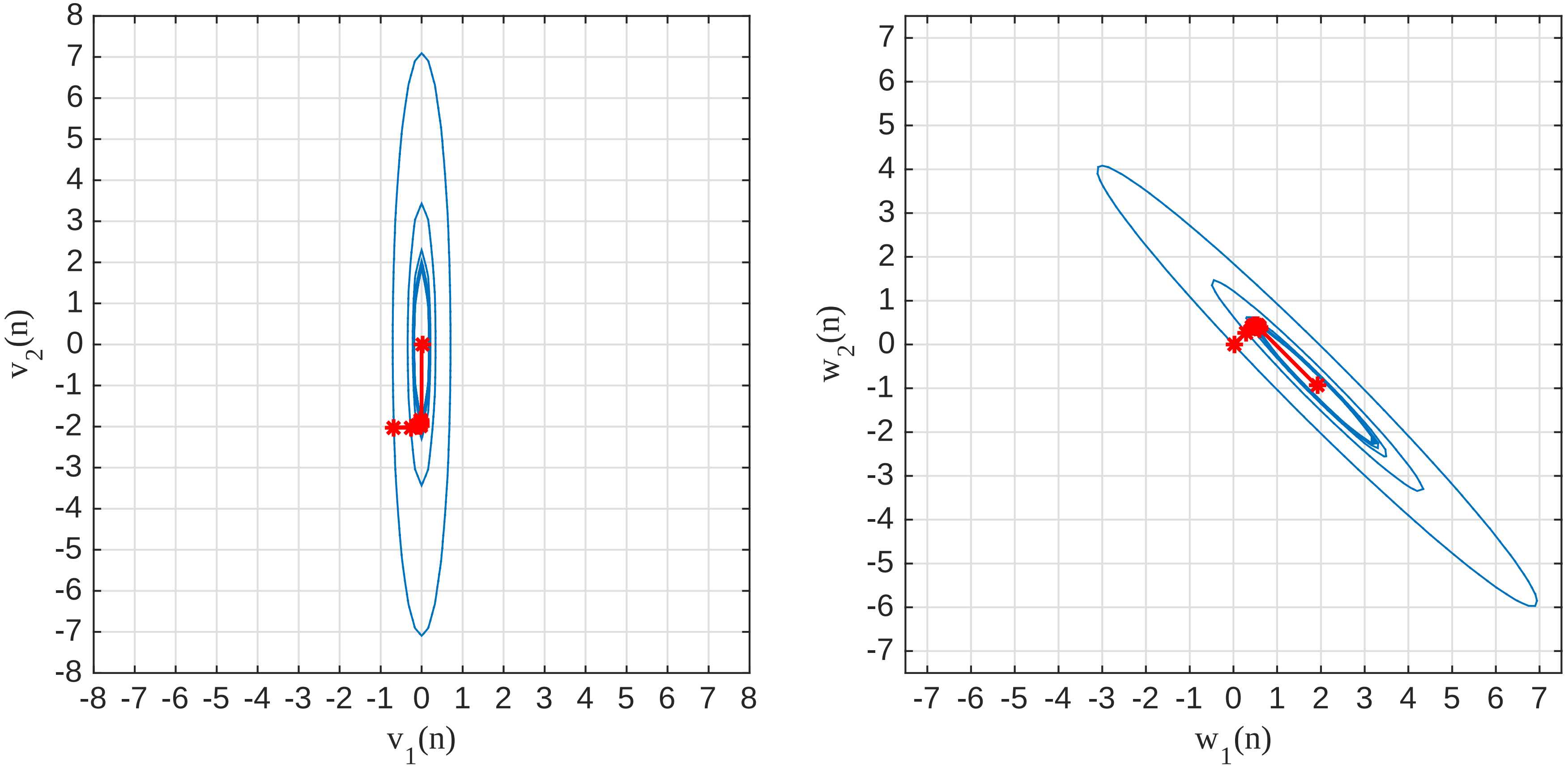
- Testing conditions:
  - $\mu = 0.3 \quad (\mu_{max} = 1.1)$
  - $\lambda_1/\lambda_2 = 10$





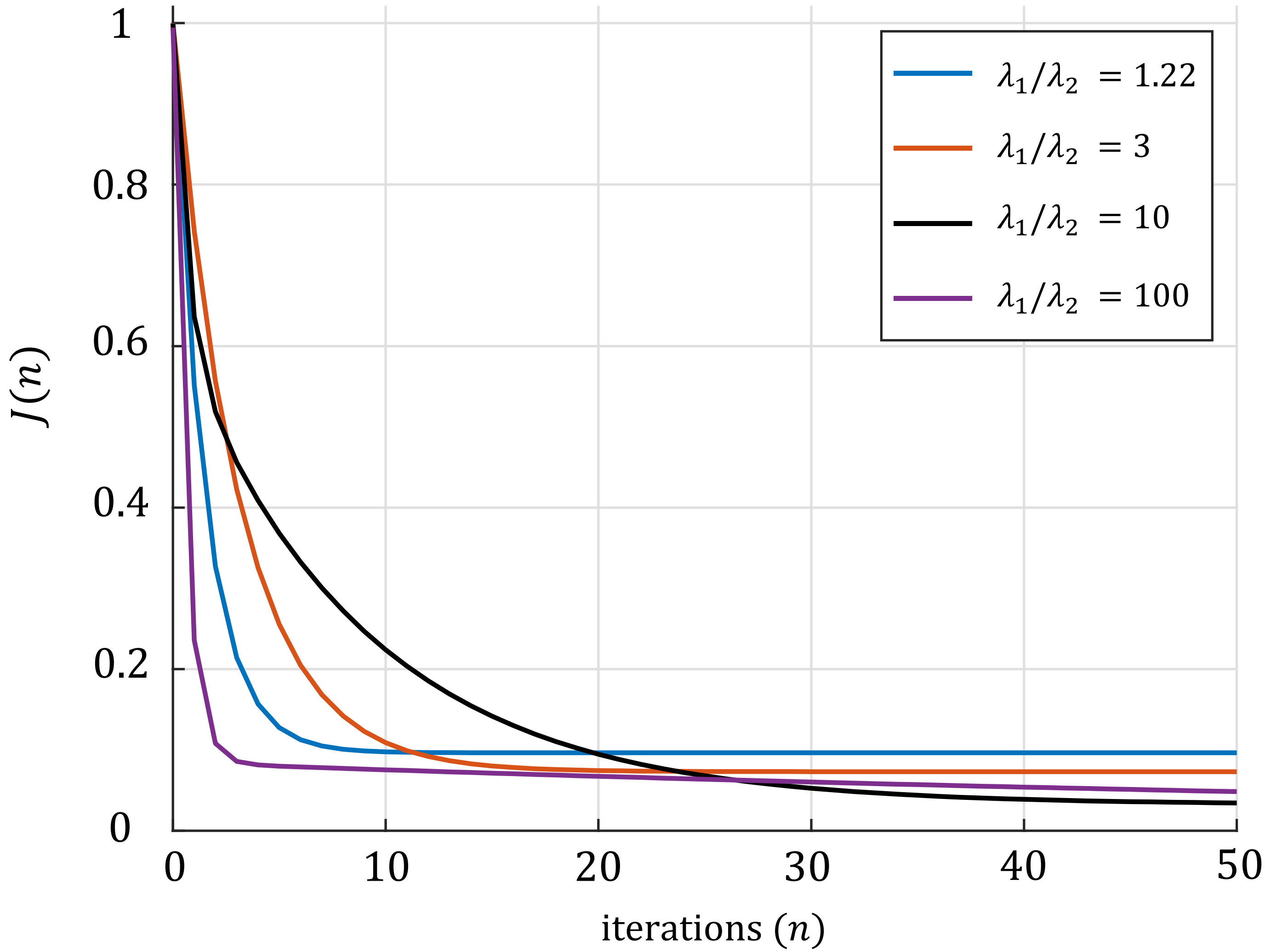
Experiment 1: varying the eigenvalue ratio

- Testing conditions:
  - $\mu = 0.3 \quad (\mu_{max} \approx 1)$
  - $\lambda_1/\lambda_2 = 100$



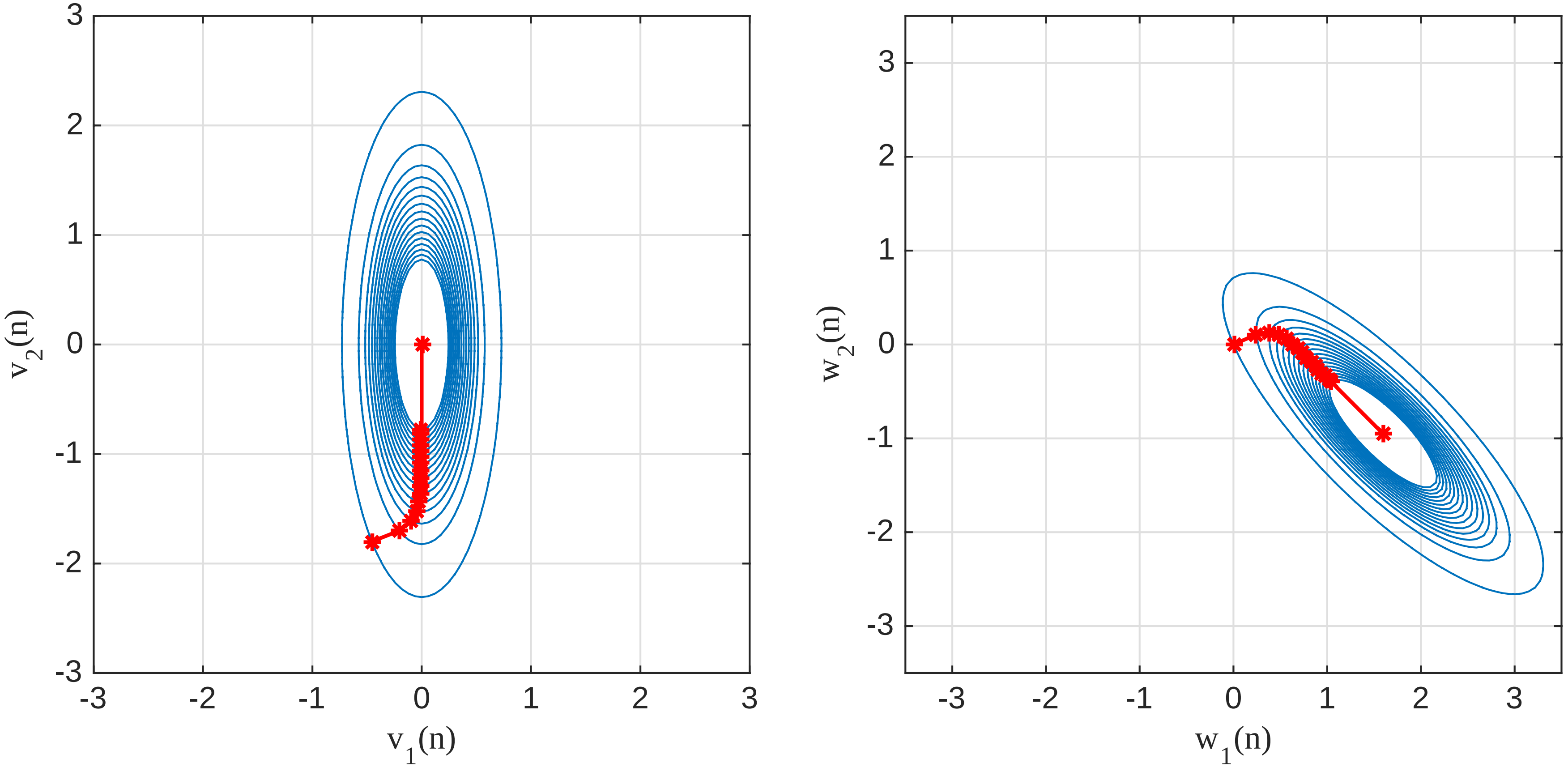
Experiment 1: varying the eigenvalue ratio

- The convergence rate can be inferred looking at the following plot:



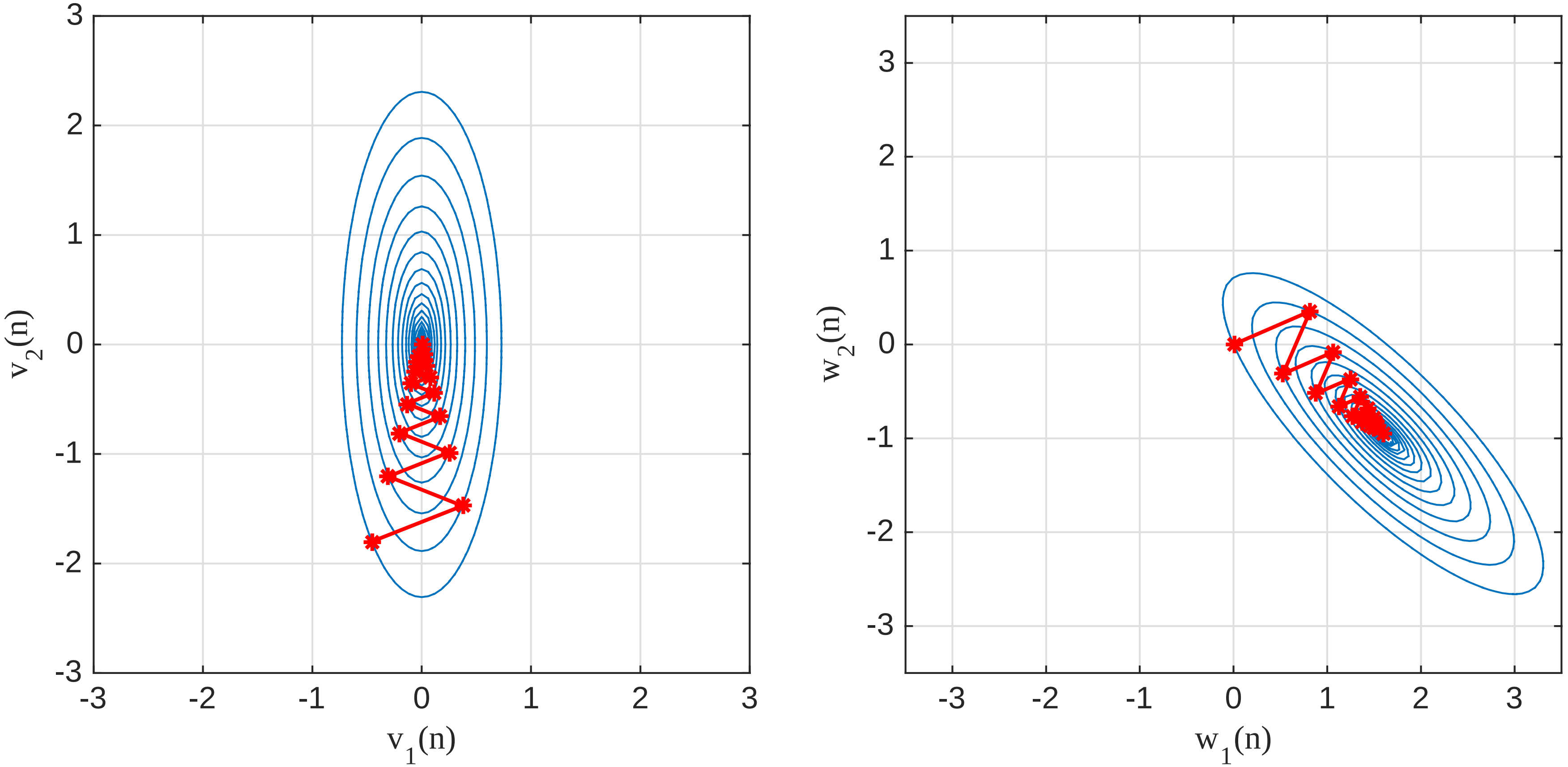
Experiment 2: varying the step-size parameter

- Testing conditions:
  - $\mu = 0.3 \ \ (\mu_{\max} = 1.1)$
  - $\lambda_1/\lambda_2 = 10$



Experiment 2: varying the step-size parameter

- Testing conditions:
  - $\mu = 1 \quad (\mu_{\max} = 1.1)$
  - $\lambda_1/\lambda_2 = 10$



Observations

- When $\lambda_1 \approx \lambda_2$, the trajectory on the planes (v_1, v_2) and (w_1, w_2) follow an almost straight line
  - this corresponds to the shortest path to reach the optimum

- This also happens when either $v_1(0) = 0$ or $v_2(0) = 0$
  - it means a right choice in the initial conditions

- In other cases, the trajectory follows a curved path
  - the more the eigenvalues ratio is large, the more the path is curved and convergence takes more time

- If the step-size parameter μ is too small, the transient behaviour is **overdampened**

- When μ approaches the maximum allowable value, the transient behaviour is **underdampened**, i.e., the trajectory exhibits oscillations