Chapter 3

Optimization

The second cooperative control problem we introduce is distributed optimization. Optimization is an important subject across mathematics, science, and engineering. Motivation of performing optimization over networked systems in a distributed fashion is driven by one or several combined factors including large scales, decentralized data collections, distributed computing technologies, and privacy concerns. One example of distributed optimization is large-scale machine learning, where big image/video data are collected and stored at different data centers, and multiple workstations in these centers perform optimization computation for global data classification or model prediction. Another example is economic dispatching in grid-connected smart buildings, where individual buildings process data of local energy generation and consumption which may be privacy-sensitive, and these buildings perform optimization computation for minimizing grid-wide generation costs subject to the constraint of meeting all consumption demands. Other application domains include power networks, smart grids, smart cities, transportation networks, and the Internet of Things (IoT).

In this chapter, we show that a necessary graphical condition to achieve distributed optimization is that the digraph is *strongly connected*. This is the same as the necessary condition for distributed averaging in the preceding chapter. Indeed, distributed optimization requires tracking the average value of the iteratively updated local optima, which intuitively demands that every agent possess a direct or indirect 'channel' in order to receive information from every other agent.

Owing to this close relation to averaging, we design a distributed optimization algorithm based on the surplus-based algorithm presented for achieving averaging over strongly connected digraphs (which need not be balanced). Further, we will relate the distributed optimization problem to a widely studied problem of distributed resource allocation. Hence the latter may also be solved by the same distributed optimization algorithm.

3.1 Problem Statement

Consider a network of n (> 1) agents. Each agent i ($\in [1, n]$) has a *state* variable $x_i(k) \in \mathbb{R}$, and a local cost function $f_i : \mathbb{R} \to \mathbb{R}$.[1] The goal of distributed optimization is that the agents cooperatively solve the following problem:

$$\min_{x_1,\ldots,x_n \in \mathbb{R}} \sum_{i=1}^{n} f_i(x_i) \qquad (3.1)$$
$$\text{subject to } x_1 = \cdots = x_n.$$

Let $F(\xi) := \sum_{i=1}^{n} f_i(\xi)$ be the global cost function for the multi-agent network. Thus problem (3.1) means that every agent minimizes the global cost function. We shall restrict our attention to the case where F has a unique optimal solution $\xi^* \in \mathbb{R}$. Denote the optimal value by

$$F^* := F(\xi^*) = \min_{\xi \in \mathbb{R}} F(\xi).$$

Under the following assumption, F indeed admits a unique optimal solution ξ^* (see Lemma 3.8 in Appendix) and a reasonable rate of convergence to the solution ξ^* is ensured.

Assumption 3.1 *Every local cost function f_i ($i \in [1, n]$)*

- *is* continuously differentiable *with gradient ∇f_i (which is derivative for one-dimensional f_i);*
- *is* strongly convex *with parameter $m_i > 0$ (or simply m_i-strongly convex), i.e.*

$$(\forall \xi_1, \xi_2 \in \mathbb{R}) f_i(\xi_1) \geq f_i(\xi_2) + \nabla f_i(\xi_2)(\xi_1 - \xi_2) + \frac{m_i}{2}\|\xi_1 - \xi_2\|_2^2; \qquad (3.2)$$

- *has a* Lipschitz-continuous gradient *with parameter $l_i > 0$ (or l_i-smooth), i.e.*

$$(\forall \xi_1, \xi_2 \in \mathbb{R}) \|\nabla f_i(\xi_1) - \nabla f_i(\xi_2)\|_2 \leq l_i \|\xi_1 - \xi_2\|_2. \qquad (3.3)$$

A straightforward characterization of the latter two conditions in Assumption 3.1 in the case that the inverse of the Hessian $\nabla^2 f_i$ (which is the reciprocal of the second derivative for f_i with one-dimensional domain) exists is:

$$m_i \leq \nabla^2 f_i \leq l_i.$$

[1]The choice of one-dimensional domain $\mathbb{R}$ of function f is made deliberately for simplicity of presentation, and the essential ideas and techniques are the same for functions of multi-dimensional domain $\mathbb{R}^N$.

Namely, strong convexity and smoothness provide respectively lower and upper bounds on $\nabla^2 f_i$. As a result, $m_i \leq l_i$ always holds. Let

$$\bar{l} := \max_{i\in[1,n]} l_i, \quad l := \sum_{i=1}^{n} l_i, \quad m := \sum_{i=1}^{n} m_i. \tag{3.4}$$

Then under Assumption 3.1, the global cost function F is m-strongly convex and l-smooth, with the *condition number* $Q := \frac{l}{m} \geq 1$.

Optimization Problem:

Consider a network of n agents interconnected through a digraph $\mathcal{G}$. Suppose that Assumption 3.1 holds and ξ^* is the (unique) optimal solution to $\min_{\xi\in\mathbb{R}} F(\xi)$. Design a distributed algorithm to update the agents' states $x_i(k)$, $i = 1, \ldots, n$, such that

$$(\forall i \in [1, n])(\forall x_i(0) \in \mathbb{R}) \lim_{k\to\infty} x_i(k) = \xi^*.$$

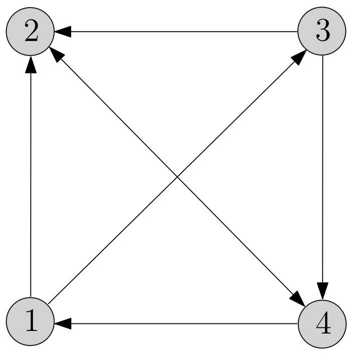

Figure 3.1: Illustrating example of optimization problem with four agents

Example 3.1 *We provide an example to illustrate the optimization problem. As displayed in Fig. 3.1, four agents are interconnected through a digraph $\mathcal{G}$. The neighbor sets of the agents are $\mathcal{N}_1 = \{4\}$, $\mathcal{N}_2 = \{1, 3, 4\}$, $\mathcal{N}_3 = \{1\}$, $\mathcal{N}_4 = \{2, 3\}$; and the out-neighbor sets are $\mathcal{N}_1^o = \{2, 3\}$, $\mathcal{N}_2^o = \{4\}$, $\mathcal{N}_3^o = \{2, 4\}$, $\mathcal{N}_4^o = \{1, 2\}$.*

Let the local cost functions of the agents be

$$\begin{aligned} f_1(\xi) &= \log(1+\mathrm{e}^{-\xi}) + 2\xi^2 \\ f_2(\xi) &= 3\log(1+\mathrm{e}^{-\xi}) + \xi^2 \\ f_3(\xi) &= 2\log(1+\mathrm{e}^{-\xi}) + 2\xi^2 + 4 \\ f_4(\xi) &= \log(1+\mathrm{e}^{-\xi}) + \xi^2 + \xi. \end{aligned}$$

Compute $\nabla^2 f_1(\xi) = \frac{\mathrm{e}^{\xi}}{(\mathrm{e}^{\xi}+1)^2} + 4$, *which lies in the interval* $(4, 4.25]$*; thus* f_1 *is* 4.05*-strongly convex and* 4.25*-smooth. Similarly,* f_2 *is* 2.05*-strongly convex and* 2.75*-smooth;* f_3 *is* 4.05*-strongly convex and* 4.5*-smooth; and* f_4 *is* 2.05*-strongly convex and* 2.25*-smooth. Hence Assumption 3.1 holds.*
The global cost function F *is*

$$F(\xi) = \sum_{i=1}^{4} f_i(\xi) = 7\log(1+\mathrm{e}^{-\xi}) + 6\xi^2 + \xi + 4$$

which is 12.05*-strongly convex and* 13.75*-smooth. The unique optimal solution to* $\min_{\xi\in\mathbb{R}} F(\xi)$ *is* $\xi^* = 0.1819$*, and the optimal value is* $F^* = 8.6247$*.*
Suppose that the initial states of the agents are $x_1(0) = 1$*,* $x_2(0) = 2$*,* $x_3(0) = 3$*,* $x_4(0) = 4$*. The optimization problem is to design a distributed algorithm such that each agent's state asymptotically converges to the optimal solution* $\xi^* = 0.1819$*.*

A necessary graphical condition for solving the optimization problem is that the digraph $\mathcal{G}$ is strongly connected (this is the same as that for solving the averaging problem).

Proposition 3.1 *Suppose that there exists a distributed algorithm that solves the optimization problem. Then the digraph* $\mathcal{G}$ *is strongly connected.*

Proof. The proof is by contradiction. Suppose that the digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is *not* strongly connected. Then at least one node (agent) in $\mathcal{V}$ is not a root of $\mathcal{G}$. Let $\mathcal{R}$ denote the set of roots. Then $\mathcal{R} \neq \mathcal{V}$. We consider two cases separately: $\mathcal{R} = \emptyset$ and $\mathcal{R} \neq \emptyset$.

If $\mathcal{R} = \emptyset$, i.e. $\mathcal{G}$ does not contain a spanning tree, then it follows from Theorem 1.1 that $\mathcal{G}$ has at least two (distinct) closed strong components (say) $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1), \mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$. In this case, consider local cost functions f_i and an initial condition such that the agents in $\mathcal{G}_1$ have initial state $c_1 \in \mathbb{R}$ that minimizes $\sum_{i\in\mathcal{V}_1} f_i(\cdot)$, those in $\mathcal{G}_2$ have $c_2 \in \mathbb{R}$ that minimizes $\sum_{i\in\mathcal{V}_2} f_i(\cdot)$, and $c_1 \neq c_2$. Since $\mathcal{G}_1$ and $\mathcal{G}_2$ are closed (i.e. information cannot be communicated from one to the other) and the agents in $\mathcal{G}_1$ (resp. $\mathcal{G}_2$) have the same state value that minimizes $\sum_{i\in\mathcal{V}_1} f_i(\cdot)$ (resp. $\sum_{i\in\mathcal{V}_2} f_i(\cdot)$), there does not exist any distributed algorithm that can update the states of the agents in $\mathcal{G}_1$ or $\mathcal{G}_2$. Consequently, no distributed algorithm can solve the optimization problem.

It is left to consider $\mathcal{R} \neq \emptyset$. In this case, $\mathcal{G}$ contains a spanning tree, and again by Theorem 1.1 that the induced digraph by $\mathcal{R}$ is the unique closed strong component in $\mathcal{G}$. Consider local cost functions f_i and an initial condition such that all agents in $\mathcal{R}$ have the same state $c \in \mathbb{R}$, which minimizes $\sum_{i\in\mathcal{R}} f_i(\cdot)$; but $c \neq \xi^*$ where ξ^* is the optimal solution for $\sum_{i\in\mathcal{V}} f_i(\cdot)$. Since $\mathcal{R}$ is closed (i.e. information cannot be communicated from $\mathcal{V}\setminus\mathcal{R}$ to $\mathcal{R}$) and the agents therein have the same state value that minimizes $\sum_{i\in\mathcal{R}} f_i(\cdot)$, there does not exist any distributed algorithm that can update the states of the agents in $\mathcal{R}$. Consequently, no distributed algorithm can solve the optimization problem. $\square$

Owing to Proposition 3.1, we shall henceforth assume that the digraph $\mathcal{G}$ is strongly connected.

Assumption 3.2 *The digraph $\mathcal{G}$ modeling the interconnection structure of the networked agents is strongly connected.*

3.2 Distributed Algorithm

Example 3.2 *Consider again Example 3.1. To converge to the optimal solution ξ^*, a natural idea is that each agent employs* gradient descent *with respect to its local cost function, while iteratively computes the average of the state values received from neighbors. Namely, for $i \in [1,4]$*

$$x_i(k+1) = x_i(k) + \sum_{j\in\mathcal{N}_i} \frac{1}{|\mathcal{N}_i|+1}(x_j(k) - x_i(k)) - \varepsilon\nabla f_i(x_i(k))$$

where $\varepsilon > 0$ is a (small or diminishing) stepsize. In vector form we have

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \\ x_4(k+1) \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \end{bmatrix} - \begin{bmatrix} \varepsilon & 0 & 0 & 0 \\ 0 & \varepsilon & 0 & 0 \\ 0 & 0 & \varepsilon & 0 \\ 0 & 0 & 0 & \varepsilon \end{bmatrix} \begin{bmatrix} \nabla f_1(x_1(k)) \\ \nabla f_2(x_2(k)) \\ \nabla f_3(x_3(k)) \\ \nabla f_4(x_4(k)) \end{bmatrix} \tag{3.5}$$

Denote by L the standard Laplacian matrix of the weighted digraph $\mathcal{G}$ in Fig. 3.1. Note that the first matrix above is $I - L$, which is row-stochastic but is not column-stochastic. The four eigenvalues of $I - L$ are:

$$1, 0.1667, 0.125 \pm 0.2602j$$

namely there is a simple eigenvalue 1 *and other eigenvalues lie within the unit circle. Thus the spectral radius of* $I-L$ *is* $\rho(I-L)=1$. *The (normalized) left eigenvector corresponding to the simple eigenvalue* 1 *is:* $\pi_l := [0.4615\ 0.3077\ 0.4615\ 0.6923]^\top$; *thus* $\pi_l^\top(I-L)=\pi_l^\top$. *Multiplying* $\pi_l^\top$ *on both sides of (3.5) above yields:*

$$\sum_{i=1}^{4}\pi_i x_i(k+1)=\sum_{i=1}^{4}\pi_i x_i(k)-\varepsilon\sum_{i=1}^{4}\pi_i\nabla f_i(x_i(k)).$$

This is a gradient descent algorithm for a different global function $F'(\xi) := \sum_{i=1}^{4}\pi_i f_i(\xi)$, *weighted by the left eigenvector* π_l *(for a different global state* $x' := \sum_{i=1}^{4}\pi_l x_i$*). Hence the above scheme does not solve the optimization of* $F(\xi)=\sum_{i=1}^{4} f_i(\xi)$, *i.e. the states do not asymptotically converge to the optimal solution of* F. *This is illustrated in Fig. 3.2; here* $\varepsilon=0.1$ *and the states converge to a vector* $[0.1035\ 0.2331\ 0.1599\ 0.0911]^\top$, *no component of which equals the optimal solution* $\xi^*=0.1819$.

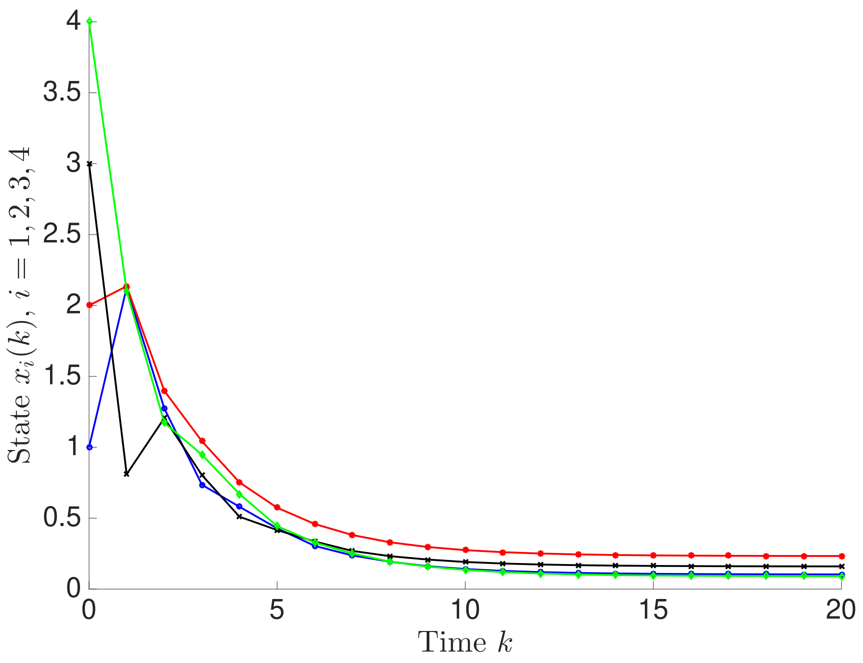

Figure 3.2: States fail to converge to the optimal solution of global cost function

Since our global function $F(\xi)=\sum_{i=1}^{n} f_i(\xi)$ is equally weighted over the local cost functions, if the left eigenvector π_l with respect to eigenvalue 1 of $I-L$ was $\mathbf{1}$ (the vector of all ones), then the scheme in Example 3.2 would have worked. In general, however, $\pi_l \neq \mathbf{1}$ for strongly connected digraphs (unless weight-balanced); instead we resort again to using surplus variables to achieve the same effect of uniform weights. Specifically, we equip each agent i with a surplus variable $s_i(k)$ to record the changes in the gradient of the local cost function, i.e. $\nabla f_i(x_i(k))$. At $k = 0$, we set $s_i(0) = \nabla f_i(x_i(0))$ for all i.

In the following, we describe a distributed algorithm that updates the state $x_i(k)$ and the surplus $s_i(k)$.

Surplus-based Optimization Algorithm (SOA):

Every agent i has a state variable $x_i(k)$ whose initial value is an arbitrary real number, and a surplus variable $s_i(k)$ whose initial value is $\nabla f_i(x_i(0))$. At each time $k \geq 0$, every agent i performs three operations:

1) Agent i sends its state $x_i(k)$ and weighted surplus $a_{ji}s_i(k)$ to each out-neighbor $j \in \mathcal{N}_i^o$. The weights a_{ji} satisfy $\sum_{j\in\mathcal{N}_i^o} a_{ji} < 1$.

2) Agent i receives the state $x_j(k)$ and weighted surplus $a_{ij}s_j(k)$ from each (in-)neighbor $j \in \mathcal{N}_i$. The weights a_{ij} satisfy $\sum_{j\in\mathcal{N}_i} a_{ij} < 1$.

3) Agent i updates its state $x_i(k)$ and surplus $s_i(k)$ as follows:

$$x_i(k+1) = x_i(k) + \sum_{j\in\mathcal{N}_i} a_{ij}(x_j(k) - x_i(k)) - \varepsilon s_i(k) \tag{3.6}$$

$$s_i(k+1) = (1 - \sum_{j\in\mathcal{N}_i^o} a_{ji})s_i(k) + \sum_{j\in\mathcal{N}_i} a_{ij}s_j(k) + \Big(\nabla f_i(x_i(k+1)) - \nabla f_i(x_i(k))\Big). \tag{3.7}$$

The parameter ε in (2.2) is a positive real number, i.e. $\varepsilon > 0$. The weights may be chosen as in Remark 2.2 to satisfy the two conditions $\sum_{j\in\mathcal{N}_i^o} a_{ji} < 1$ and $\sum_{j\in\mathcal{N}_i} a_{ij} < 1$.

Remark 3.1 *In SOA, (3.6) is the state update equation by the gradient descent scheme as described in Example 3.2, treating $s_i(k)$ as the estimate of gradient of the local cost function. On the other hand, (3.7) is the surplus update equation where the first two terms represent sending (resp. receiving) surplus to out-neighbors (resp. from neighbors), and the third term records the change in gradients. Summing up (3.7) from $i = 1$ to n on both sides, we derive*

$$\sum_{i=1}^n s_i(k+1) = \sum_{i=1}^n \left((1 - \sum_{j\in\mathcal{N}_i^o} a_{ji})s_i(k) + \sum_{j\in\mathcal{N}_i} a_{ij}s_j(k) \right) + \sum_{i=1}^n \Big(\nabla f_i(x_i(k+1)) - \nabla f_i(x_i(k))\Big)$$

$$\Rightarrow \sum_{i=1}^n s_i(k+1) - \sum_{i=1}^n s_i(k) = \sum_{i=1}^n \nabla f_i(x_i(k+1)) - \sum_{i=1}^n \nabla f_i(x_i(k)).$$

Since $s_i(0) = \nabla f_i(x_i(0))$*, we conclude that for every* $k \geq 0$*,*

$$\sum_{i=1}^{n} s_i(k) = \sum_{i=1}^{n} \nabla f_i(x_i(k)).$$

Thus the sum of surplus variables $s_i(k)$ *is the sum of gradients of the local cost functions at time* k*.*

Remark 3.2 *(Relation with SAA) Consider* (i) *the special quadratic cost function* $f_i(x_i) := \frac{1}{2}x_i^2$ *(thus* $\nabla f_i(x_i) = x_i$*); and* (ii) *change of variable* $\hat{s}_i := -s_i$*. Substituting these into SOA, we obtain SAA with surplus variable* $\hat{s}_i$*. Note that* $s_i \to 0$ *if and only if* $\hat{s}_i \to 0$*. Owing to this relation, SOA is a generalization of SAA.*

Remark 3.3 *Let*

$$x := \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n, \quad s := \begin{bmatrix} s_1 \\ \vdots \\ s_n \end{bmatrix} \in \mathbb{R}^n, \quad \nabla f(x) := \begin{bmatrix} \nabla f_1(x_1) \\ \vdots \\ \nabla f_n(x_n) \end{bmatrix} \in \mathbb{R}^n$$

be respectively the aggregated state, surplus, and gradient of the networked agents. Then SOA is written compactly as follows:

$$\begin{aligned} x(k+1) &= (I - L)x(k) - \varepsilon s(k) \\ s(k+1) &= (I - L^o)s(k) + (\nabla f(x(k+1)) - \nabla f(x(k))) \end{aligned} \tag{3.8}$$

where $I - L$ *is row-stochastic and* $I - L^o$ *column-stochastic. The initial conditions are* $x(0) \in \mathbb{R}^n$ *(arbitrary) and* $s(0) = \nabla f(x(0))$*.*

Example 3.3 *Let us revisit Example 3.2. It is checked that the weights* a_{ij} *satisfy the two conditions* $\sum_{j \in \mathcal{N}_i^o} a_{ji} < 1$ *and* $\sum_{j \in \mathcal{N}_i} a_{ij} < 1$*. Then SOA in vector form is:*

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \\ x_4(k+1) \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \end{bmatrix} - \begin{bmatrix} \varepsilon & 0 & 0 & 0 \\ 0 & \varepsilon & 0 & 0 \\ 0 & 0 & \varepsilon & 0 \\ 0 & 0 & 0 & \varepsilon \end{bmatrix} \begin{bmatrix} s_1(k) \\ s_2(k) \\ s_3(k) \\ s_4(k) \end{bmatrix}$$

$$\begin{bmatrix} s_1(k+1) \\ s_2(k+1) \\ s_3(k+1) \\ s_4(k+1) \end{bmatrix} = \begin{bmatrix} \frac{1}{4} & 0 & 0 & \frac{1}{2} \\ \frac{1}{4} & \frac{2}{3} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{2} & 0 & \frac{5}{12} & 0 \\ 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{4} \end{bmatrix} \begin{bmatrix} s_1(k) \\ s_2(k) \\ s_3(k) \\ s_4(k) \end{bmatrix} + \begin{bmatrix} \nabla f_1(x_1(k+1)) - \nabla f_1(x_1(k)) \\ \nabla f_2(x_2(k+1)) - \nabla f_2(x_2(k)) \\ \nabla f_3(x_3(k+1)) - \nabla f_3(x_3(k)) \\ \nabla f_4(x_4(k+1)) - \nabla f_4(x_4(k)) \end{bmatrix}.$$

Fig. 3.3 displays the case in which all states converge to the optimal solution $\xi^* = 0.1819$ when the parameter $\varepsilon = 0.1$; while Fig. 3.4 shows that when $\varepsilon = 0.2$, convergence does not occur. Hence similar to SAA for the averaging problem, the parameter ε needs to be carefully chosen (to be small enough) so as to ensure convergence.

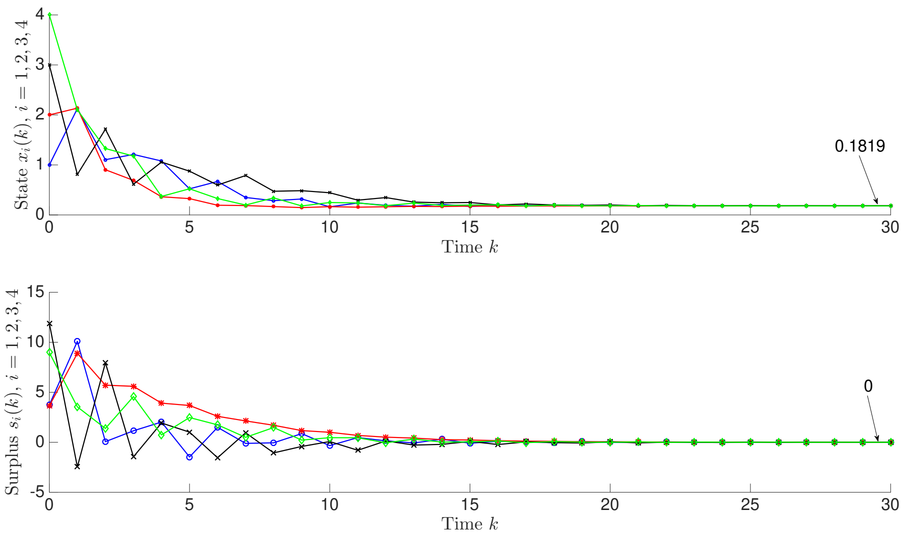

Figure 3.3: Convergence to optimal solution when $\varepsilon = 0.1$

3.3 Convergence Result

The following is the main result of this section.

Theorem 3.1 *Suppose that Assumptions 3.1 and 3.2 hold. If the parameter $\varepsilon > 0$ is sufficiently small, then SOA solves the optimization problem.*

Consider the two matrices $I - L$ and $I - L^o$. Under Assumption 3.2 and by Lemma 2.1, the spectral radius $\rho(I - L) = 1$ is a simple eigenvalue with a positive left-eigenvector π_l such that

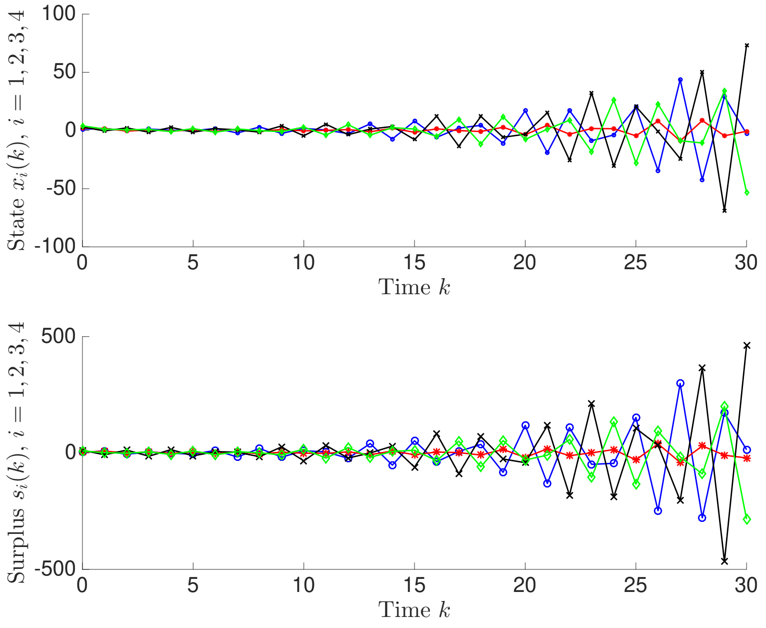

Figure 3.4: Failure to converge when $\varepsilon = 0.2$

$\pi_l^\top \mathbf{1} = 1$; and $\rho(I - L^o) = 1$ is also a simple eigenvalue with a positive eigenvector π_r such that $\pi_r^\top \mathbf{1} = 1$. Write $\Pi_l := \mathbf{1}\pi_l^\top$ and $\Pi_r := \pi_r \mathbf{1}^\top$. The proof of Theorem 3.1 is structured into the following three steps. First, we construct two special vector norms $\|\cdot\|_{\Pi_l}, \|\cdot\|_{\Pi_r}$ with which $I - L$ and $I - L^o$ have a special contraction property. Second, when the parameter $\varepsilon > 0$ satisfies a certain bound, we bound several relevant norms to derive the following inequality:

$$\begin{bmatrix} \|x(k+1) - \Pi_l x(k+1)\|_{\Pi_l} \\ \|\Pi_l x(k+1) - \xi^* \mathbf{1}\|_2 \\ \|s(k+1) - \Pi_r s(k+1)\|_{\Pi_r} \end{bmatrix} \le C \begin{bmatrix} \|x(k) - \Pi_l x(k)\|_{\Pi_l} \\ \|\Pi_l x(k) - \xi^* \mathbf{1}\|_2 \\ \|s(k) - \Pi_r s(k)\|_{\Pi_r} \end{bmatrix} \tag{3.9}$$

where C is a nonnegative matrix. Finally, we prove for small $\varepsilon > 0$ that the spectral radius of C satisfies $\rho(C) < 1$. Hence all three eigenvalues of C lie within the unit circle; thereby

$$\begin{bmatrix} \|x(k) - \Pi_l x(k)\|_{\Pi_l} \\ \|\Pi_l x(k) - \xi^* \mathbf{1}\|_2 \\ \|s(k) - \Pi_r s(k)\|_{\Pi_r} \end{bmatrix} \to 0.$$

In particular $x(k) \to \xi^* \mathbf{1}$, meaning that all the states converge to the optimal solution ξ^* of the