Chapter 2

Averaging

The first cooperative control problem we introduce is distributed averaging. Averaging is simple and useful in many contexts of networked systems. One example is *load balancing*: say there are five interconnected machines and ten jobs, having each machine process two jobs is the most efficient. Another example is environment measuring by *sensor networks*: if each sensor has measured an environment parameter, say temperature, contaminated by white noise, then the average of these measurements is the unbiased, minimum mean-squared error estimate of the true temperature. Other examples include cyclic pursuit, clock synchronization, and social influencing.

Networked systems and the interactions among component agents (via sensing or communication) are naturally modeled by digraphs. In this chapter, we show that a necessary graphical condition to achieve distributed averaging is that the digraph is *strongly connected*, namely every agent is reachable from every other agent. This is intuitively evident, as for locally computing the global average, each agent needs a 'channel', direct or indirect, to receive information from every other agent.

If the digraph is furthermore *balanced*, meaning roughly that each agent receives equal amount of in-flow information and out-going information, then averaging is easily solvable by a distributed algorithm (the consensus algorithm to be introduced in Chapter 4). However, *balanced* is neither a mild graphical condition nor a necessary condition for averaging. Hence we will assume only strongly connected digraphs (possibly unbalanced), and design a distributed algorithm that achieves averaging.

2.1 Problem Statement

Consider a network of n (> 1) agents. Each agent i ($\in [1, n]$) has a *state* variable $x_i(k) \in \mathbb{R}$, where $k \geq 0$ is a nonnegative integer and denotes the *discrete* time.

We model the interconnection structure of the networked agents by a digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$: Each node in $\mathcal{V} = \{1, ..., n\}$ stands for an agent, and each (directed) edge (j, i) in $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes that agent j communicates to agent i (namely, the information flow is from j to i). The (in-)neighbor set of agent i is $\mathcal{N}_i := \{j \in \mathcal{V} \mid (j, i) \in \mathcal{E}\}$, while the out-neighbor set $\mathcal{N}_i^o := \{j \in \mathcal{V} \mid (i, j) \in \mathcal{E}\}$.

We say that an algorithm is *distributed* if every agent i updates its state $x_i(k)$ based only on the information received from $\mathcal{N}_i$, and sends information only to $\mathcal{N}_i^o$.

Averaging Problem:

Consider a network of n agents interconnected through a digraph $\mathcal{G}$. Design a distributed algorithm to update the agents' states $x_i(k)$, $i = 1, \ldots, n$, such that

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

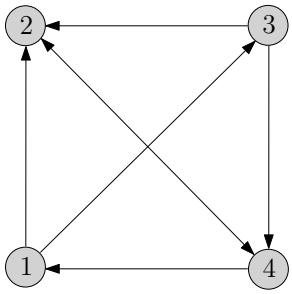

Figure 2.1: Illustrating example of averaging problem with four agents

Example 2.1 *We provide an example to illustrate the averaging problem. As displayed in Fig. 2.1, 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 $\mathcal{N}_1^o = \{2, 3\}$, $\mathcal{N}_2^o = \{4\}$, $\mathcal{N}_3^o = \{2, 4\}$, $\mathcal{N}_4^o = \{1, 2\}$.*
Suppose that the initial states of the agents are $x_1(0) = 1$, $x_2(0) = 2$, $x_3(0) = 3$, $x_4(0) = 4$. Then the average is 2.5. The averaging problem is to design a distributed algorithm such that each agent's state asymptotically converges to the average value 2.5.

A necessary graphical condition for solving the averaging problem is given below.

Proposition 2.1 *Suppose that there exists a distributed algorithm that solves the averaging 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) $\mathcal{G}_1, \mathcal{G}_2$. In this case, consider an initial condition

such that the agents in $\mathcal{G}_1$ have initial state $c_1 \in \mathbb{R}$, those in $\mathcal{G}_2$ have $c_2 \in \mathbb{R}$, and $c_1 \neq c_2$. Since $\mathcal{G}_1$ and $\mathcal{G}_2$ are closed, information cannot be communicated from one to the other. Consequently, there exists no distributed algorithm that can solve the averaging 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 subdigraph by $\mathcal{R}$ is the unique closed strong component in $\mathcal{G}$. Consider an initial condition such that all agents in $\mathcal{R}$ have initial state $c \in \mathbb{R}$, those in $\mathcal{V} \setminus \mathcal{R}$ have $c' \in \mathbb{R}$, and $c \neq c'$. Since $\mathcal{R}$ is closed, information cannot be communicated from $\mathcal{V} \setminus \mathcal{R}$ to $\mathcal{R}$. Consequently, there exists no distributed algorithm that can solve the averaging problem. □

Owing to Proposition 2.1, we shall henceforth assume that the digraph $\mathcal{G}$ is strongly connected.

Assumption 2.1 *The digraph $\mathcal{G}$ modeling the interconnection structure of the networked agents is strongly connected.*

2.2 Distributed Algorithm

Example 2.2 *Consider again Example 2.1. To achieve averaging, a natural idea is that each agent iteratively computes the (local) average of the state values received from neighbors and its own state value. Namely, for $i \in [1, 4]$*

$$\begin{aligned} x_i(k+1) &= \frac{1}{|\mathcal{N}_i|+1}(x_i(k) + \sum_{j \in \mathcal{N}_i} x_j(k)) \\ &= x_i(k) + \sum_{j \in \mathcal{N}_i} \frac{1}{|\mathcal{N}_i|+1}(x_j(k) - x_i(k)). \end{aligned}$$

For the initial states of the agents $x_1(0) = 1$, $x_2(0) = 2$, $x_3(0) = 3$, $x_4(0) = 4$, let us compute by the above equation the new states at $k = 1$:

$$\begin{aligned} x_1(1) &= x_1(0) + \frac{1}{2}(x_4(0) - x_1(0)) = \frac{1}{2}x_1(0) + \frac{1}{2}x_4(0) = 2.5 \\ x_2(1) &= x_2(0) + \frac{1}{4}(x_1(0) - x_2(0)) + \frac{1}{4}(x_3(0) - x_2(0)) + \frac{1}{4}(x_4(0) - x_2(0)) = 2.5 \\ x_3(1) &= x_3(0) + \frac{1}{2}(x_1(0) - x_3(0)) = \frac{7}{3} \\ x_4(1) &= x_4(0) + \frac{1}{3}(x_2(0) - x_4(0)) + \frac{1}{3}(x_3(0) - x_4(0)) = 3. \end{aligned}$$

Observe that the state sum at time $k = 1$ is $\sum_{i=1}^4 x_i(1) = \frac{31}{3}$, while the initial state sum $\sum_{i=1}^4 x_i(0) = 10$. The state sum has changed (by $\frac{1}{3}$) after one update, and this is in fact due to unbalanced structure of the digraph $\mathcal{G}$ in Fig. 2.1. Indeed, let $a_{ij} = \frac{1}{|\mathcal{N}_i|+1}$ be the

(positive) weight of edge $(j,i) \in \mathcal{E}$; then the weighted degrees are $d_1 = \frac{1}{2}$, $d_2 = \frac{3}{4}$, $d_3 = \frac{1}{2}$, $d_4 = \frac{2}{3}$, while the weighted out-degrees $d_1^o = \frac{3}{4}$, $d_2^o = \frac{1}{3}$, $d_3^o = \frac{7}{12}$, $d_4^o = \frac{3}{4}$ — the weighted digraph is thus not weight-balanced.
Note that the adjacency matrix and standard Laplacian matrix of the weighted digraph $\mathcal{G}$ are:

$$A = \begin{bmatrix} 0 & 0 & 0 & \frac{1}{2} \\ \frac{1}{4} & 0 & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{3} & \frac{1}{3} & 0 \end{bmatrix}, \quad L = \begin{bmatrix} \frac{1}{2} & 0 & 0 & -\frac{1}{2} \\ -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & -\frac{1}{3} & -\frac{1}{3} & \frac{2}{3} \end{bmatrix}.$$

Hence the above state-update scheme may be written in vector form:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \\ x_4(k+1) \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \end{bmatrix} = (I - L) \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \end{bmatrix}.$$

The matrix $I - L$ is nonnegative and every row sums up to one; thus $I - L$ is a row-stochastic *matrix. On the other hand, not every column of $I - L$ sums up to one, so $I - L$ is not* column-stochastic *(and this is caused by non-weight-balancedness of the digraph $\mathcal{G}$). This means that the initial sum is not kept invariant during each state update, and consequently asymptotic convergence to the initial average is not achievable. This is illustrated in Fig. 2.2.*

The problem illustrated by Example 2.2 suggests a plausible remedy: equip each agent i with an additional variable $s_i(k)$ to record the changes in state $x_i(k)$, such that the sum of $x_i(k)$ and $s_i(k)$ is a constant, i.e.

$$(\forall k \geq 0) \sum_{i=1}^{n} (x_i(k+1) + s_i(k+1)) = \sum_{i=1}^{n} (x_i(k) + s_i(k)).$$

We call $s_i(k)$ the *surplus* variable of agent i at time k. At $k = 0$, set $s_i(0) = 0$ for all i; this is intuitive because there is no change yet in state $x_i(0)$ to be recorded. Hence for every $k \geq 0$, there holds

$$\sum_{i=1}^{n} (x_i(k) + s_i(k)) = \sum_{i=1}^{n} (x_i(0) + s_i(0)) = \sum_{i=1}^{n} x_i(0). \tag{2.1}$$

Namely the initial state sum is kept invariant using the surplus variables.

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

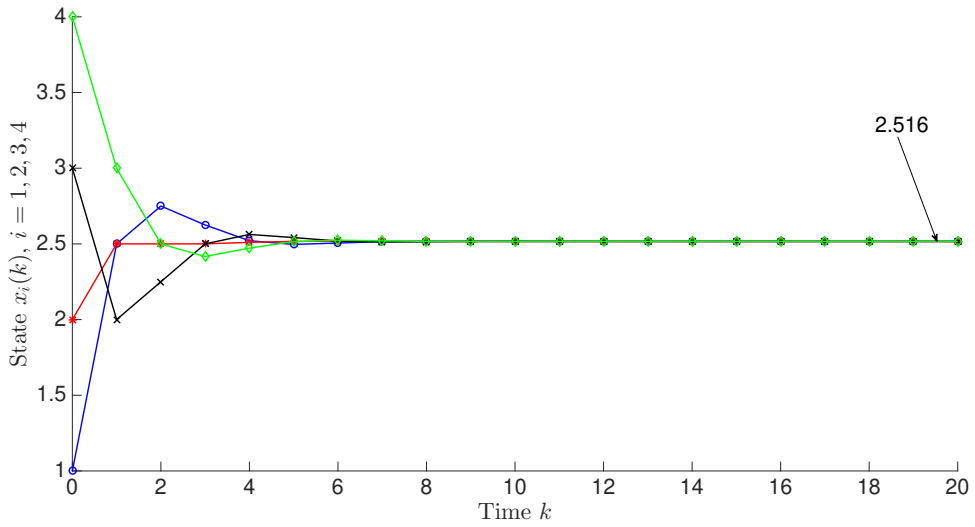

Figure 2.2: Failure to achieve averaging

$s_i(k)$ such that (2.1) holds.

Surplus-based Averaging Algorithm (SAA):

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 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:

$$x_i(k+1) = x_i(k) + \sum_{j\in\mathcal{N}_i} a_{ij}(x_j(k) - x_i(k)) + \varepsilon s_i(k) \tag{2.2}$$

$$s_i(k+1) = (1 - \sum_{j\in\mathcal{N}_i^o} a_{ji})s_i(k) + \sum_{j\in\mathcal{N}_i} a_{ij}s_j(k) - \Big(x_i(k+1) - x_i(k)\Big). \tag{2.3}$$

The parameter ε in (2.2) is a positive real number, i.e. $\varepsilon > 0$.

Remark 2.1 *In SAA, (2.2) is the state update equation where the first two terms on the right constitute the averaging scheme in Example 2.2, and the last term specifies a certain amount of surplus used to influence the state update. On the other hand, (2.3) is the surplus update equation*

where the first two terms on the right represent sending (resp. receiving) surplus to out-neighbors (resp. from neighbors), and the third term records the change of state. Summing up (2.3) from $i = 1$ to n on both sides, we derive

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

Hence SAA ensures constant sum of states and surpluses for all times; namely (2.1) holds.

Remark 2.2 *In SAA, the weights a_{ij} are required to satisfy two conditions: $\sum_{j \in \mathcal{N}_i^o} a_{ji} < 1$ and $\sum_{j \in \mathcal{N}_i} a_{ij} < 1$. In Example 2.2 the weights are chosen to be $a_{ij} = \frac{1}{|\mathcal{N}_i|+1}$ for every $j \in \mathcal{N}_i$, and for that example the two conditions are satisfied. However, in general this choice only ensures $\sum_{j \in \mathcal{N}_i} a_{ij} < 1$ but not necessarily $\sum_{j \in \mathcal{N}_i^o} a_{ji} < 1$. An example illustrating this point is a variant of the digraph in Fig. 2.1 with an additional edge $(4, 3)$: in this case $\sum_{j \in \mathcal{N}_4^o} a_{j4} = \frac{1}{2} + \frac{1}{4} + \frac{1}{3} > 1$. A simple choice that does ensure both conditions is the following:*

$$a_{ij} = \min \Big\{ \frac{1}{|\mathcal{N}_i| + 1}, \frac{1}{|\mathcal{N}_i^o| + 1} \Big\}.$$

Another simple choice that requires the knowledge of the number of agents is $a_{ij} = \frac{1}{n}$.

Remark 2.3 *Let $x := [x_1 \cdots x_n]^\top \in \mathbb{R}^n$ and $s := [s_1 \cdots s_n]^\top \in \mathbb{R}^n$ be the aggregated state and surplus, respectively, of the networked agents. Then the n equations of (2.2) become*

$$x(k+1) = (I - L)x(k) + \varepsilon s(k).$$

Since $\sum_{j \in \mathcal{N}_i} a_{ij} < 1$, $I - L$ is nonnegative. Moreover, since L has zero row sums, $I - L$ is row stochastic. *On the other hand, the n equations of (2.3) become*

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

where L^o is the out-degree Laplacian matrix (Remark 1.1 in Section 1.3). Since $\sum_{j \in \mathcal{N}_i^o} a_{ji} < 1$, $I - L^o$ is also nonnegative. Moreover, since L^o has zero column sums, $I - L^o$ is column stochastic. *Together, SAA is written compactly as follows:*

$$\begin{bmatrix} x(k+1) \\ s(k+1) \end{bmatrix} = M \begin{bmatrix} x(k) \\ s(k) \end{bmatrix}, \text{ where } M := \begin{bmatrix} I - L & \varepsilon I \\ L & I - L^o - \varepsilon I \end{bmatrix}. \tag{2.4}$$

The initial conditions are $x(0) \in \mathbb{R}^n$ *(arbitrary) and* $s(0) = 0$*. Notice that*

- *the matrix* M *has negative entries due to the presence of the Laplacian matrix* L *in the* $(2,1)$*-block;*
- *the column sums of* M *are equal to one, which implies that the quantity* $\mathbf{1}^T(x(k) + s(k))$ *is a constant for all* $k \geq 0$ *(cf. (2.1));*
- *the state evolution specified by the* $(1,1)$*-block of* M*, i.e.* $x(k+1) = (I-L)x(k)$ *is the averaging scheme in Example 2.2.*

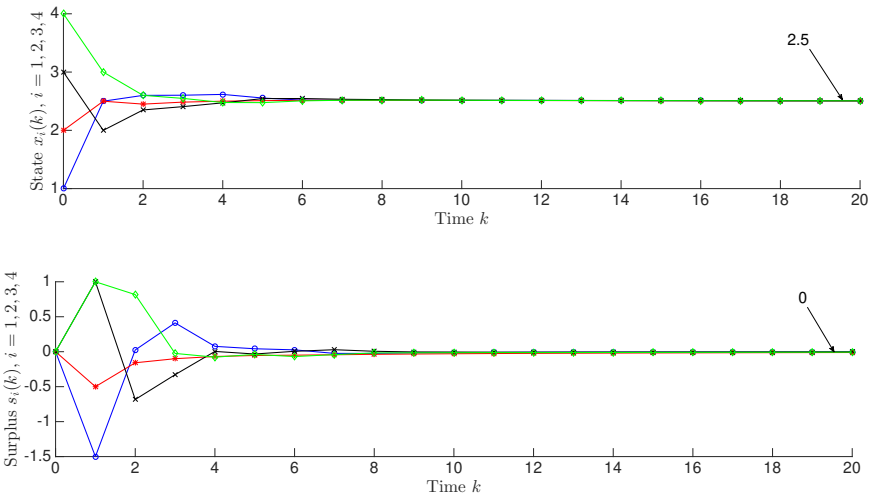

Figure 2.3: Convergence to average consensus when $\varepsilon = 0.1$

Example 2.3 *Let us revisit Example 2.2. It is checked that the weights* a_{ij} *satisfy the two conditions* $\sum_{j \in \mathcal{N}_i^o} a_{ji} < 1$ *and* $\sum_{j \in \mathcal{N}_i} a_{ij} < 1$*. We have seen the standard Laplacian matrix* L *and the row-stochastic* $I - L$*. The following are the out-degree Laplacian matrix* L^o *and the column-stochastic* $I - L^o$*:*

$$L^o = \begin{bmatrix} \frac{3}{4} & 0 & 0 & -\frac{1}{2} \\ -\frac{1}{4} & \frac{1}{3} & -\frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{2} & 0 & \frac{7}{12} & 0 \\ 0 & -\frac{1}{3} & -\frac{1}{3} & \frac{3}{4} \end{bmatrix}, \quad I - L^o = \begin{bmatrix} \frac{1}{4} & 0 & 0 & \frac{1}{2} \\ \frac{1}{4} & \frac{2}{3} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{2} & 0 & \frac{5}{12} & 0 \\ 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{4} \end{bmatrix}.$$

With these matrices, the matrix M *in (2.4) may be constructed. Fig. 2.3 displays the case*

in which convergence to the initial average 2.5 *is achieved when the parameter* $\varepsilon = 0.1$*; while Fig. 2.4 shows that when* $\varepsilon = 0.5$*, convergence does not occur. Hence the parameter* ε *needs to be carefully chosen (to be small enough) so as to achieve averaging.*

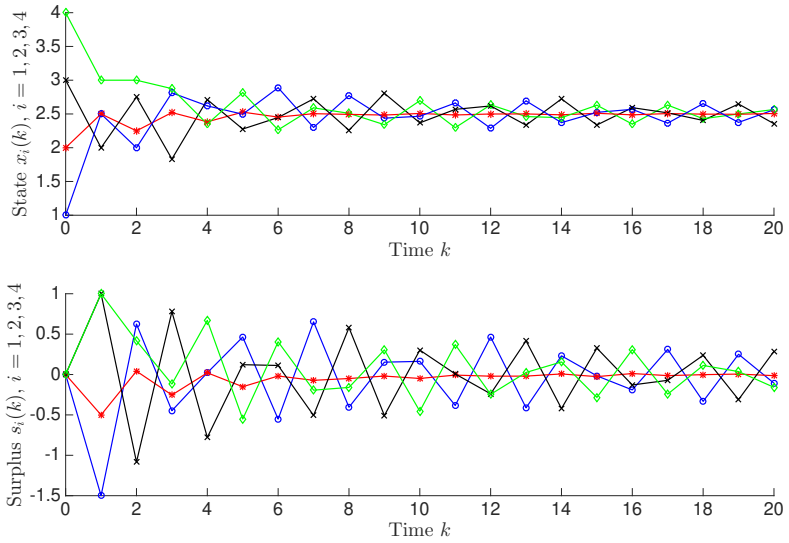

Figure 2.4: Failure to converge when $\varepsilon = 0.5$

2.3 Convergence Result

The following is the main result of this section.

Theorem 2.1 *Suppose that Assumption 2.1 holds. If the parameter* $\varepsilon > 0$ *is sufficiently small, then SAA solves the averaging problem.*

To prove Theorem 2.1, we will analyze the eigenvalues and eigenvectors of matrix M in (2.4). Write M in two parts: $M = M_0 + \varepsilon E$, where

$$M_0 := \begin{bmatrix} I - L & 0 \\ L & I - L^o \end{bmatrix}, \quad E := \begin{bmatrix} 0 & I \\ 0 & -I \end{bmatrix}.$$

The proof of Theorem 2.1 is structured in two steps. First, we analyze the eigenvalues and eigenvectors of M_0. Second, we analyze the (infinitesimal) movement of M_0's eigenvalues upon being