Partial Solutions Manual

Parallel and Distributed Computation: Numerical Methods

Dimitri P. Bertsekas and John N. Tsitsiklis

Massachusetts Institute of Technology

WWW site for book information and orders

http://world.std.com/~athenasc/

Athena Scientific, Belmont, Massachusetts

CHAPTER 1

SECTION 1.2

1.2.1:

The inequality $T_\infty \geq \log n$ in the statement of Prop. 2.1 is replaced by the inequality $T_\infty \geq \log_B n$, where $\log_B$ stands for the base B logarithm. The proof is similar with the proof of Prop. 2.1. We show by induction on k, that $t_j \geq \log_B k$ for every node j depending on k inputs, and for every schedule. The claim is clearly true if $k = 1$. Assume that the claim is true for every k smaller than some k_0 and consider a node j that depends on $k_0 + 1$ inputs. Since j has at most B predecessors, it has a predecessor ℓ that depends on at least $\lceil (k_0 + 1)/B \rceil$ inputs. Then, using the induction hypothesis,

$$t_j \geq t_\ell + 1 \geq \log_B \left\lceil \frac{k_0 + 1}{B} \right\rceil + 1 \geq \log_B \left(\frac{k_0 + 1}{B} \right) + 1 = \log_B (k_0 + 1),$$

which completes the induction.

1.2.2:

Here we need to assume that a particular algorithm has been fixed and that $T^*(n)$ is its serial execution time. We assume that a fraction f of the algorithm is inherently serial. This part requires $fT^*(n)$ time no matter how many processors we try to use. The remaining fraction, which is $1 - f$, needs at least $(1 - f)T^*(n)/p$ time, when p processors are available. Thus, $T_p(n) \geq \big(f + (1 - f)/p\big)T^*(n)$, which yields

$$S_p(n) = \frac{T^*(n)}{T_p(n)} \leq \frac{1}{f + (1 - f)/p} \leq \frac{1}{f}.$$

1.2.3:

(a) The main idea here is to divide and conquer. We start by considering the case where n is a power of 2. If $n = 1$, there is nothing to be done and the problem is solved in zero time using

zero processors. Assume that have already constructed a parallel algorithm that solves the prefix problem for some n which is a power of 2, in time $T(n)$ using $p(n)$ processors. We now construct an algorithm that solves the same problem when n is replaced by $2n$. We use the already available algorithm to compute all of the quantities $\prod_{i=1}^{k} a_i$, $k = 1, 2, \ldots, n$, and $\prod_{i=n+1}^{k} a_i$, $k = n+1, \ldots, 2n$. This amounts to solving two prefix problems, each one involving n numbers. This can be done in parallel, in time $T(n)$ using $2p(n)$ processors. We then multiply each one of the numbers $\prod_{i=n+1}^{k} a_i$, $k = n+1, \ldots, 2n$, by $\prod_{i=1}^{n} a_i$, and this completes the desired computation. This last stage can be performed in a single stage, using n processors.

The above described recursive definition provides us with a prefix algorithm for every value of n which is a power of 2. Its time and processor requirements are given by

$$T(2n) = T(n) + 1,$$

$$p(2n) = \max\Big\{2p(n), n\Big\}.$$

Using the facts $T(1) = 0$ and $p(1) = 0$, an easy inductive argument shows that $T(n) = \log n$ and $p(n) = n/2$.

The case where n is not a power of 2 cannot be any harder than the case where n is replaced by the larger number $2^{\lceil \log n \rceil}$. Since the latter number is a power of 2, we obtain

$$T(n) \leq T\Big(2^{\lceil \log n \rceil}\Big) = \log\Big(2^{\lceil \log n \rceil}\Big) = \lceil \log n \rceil,$$

and

$$p(n) \leq p\Big(2^{\lceil \log n \rceil}\Big) = 2^{\lceil \log n \rceil - 1} < 2^{logn} = n.$$

(b) The algorithm is identical with part (a) except that each scalar multiplication is replaced by a multiplication of two $m \times m$ matrices. Each such multiplication can be performed in time $O(\log m)$ using $O(m^3)$ processors, as opposed to unit time with one processor. Thus, the time of the algorithm is $O(\log m \cdot \log n)$, using $O(nm^3)$ processors.

(c) The solution of the difference equation $x(t+1) = A(t)x(t) + u(t)$ is given by

$$x(n) = \Big[\prod_{j=0}^{n-1} A(j)\Big]x(0) + \sum_{i=0}^{n-1}\Big[\prod_{j=i+1}^{n-1} A(j)\Big]u(i).$$

We use the algorithm of part (b) to compute the matrix products $\prod_{j=i}^{n-1} A(j)$, for $i = 0, 1, \ldots, n-1$. This takes $O(\log m \cdot \log n)$ time. We then multiply each such product with the corresponding vector $u(i)$ or $x(0)$ [this can be done in parallel, in $O(\log m)$ time], and then add the results [this can be done in parallel in $O(\log n)$ time]. We conclude that the overall time is $O(\log m \cdot \log n) + O(\log m) + O(\log n) = O(\log m \cdot \log n)$.

1.2.4:

We represent k in the form

$$k = \sum_{i=0}^{\lfloor \log k \rfloor} b_i 2^i,$$

where each b_i belongs to the set $\{0, 1\}$. (In particular, the coefficients b_i are the entries in the binary representation of k.) Then,

$$A^k = \prod_{i=0}^{\lfloor \log k \rfloor} A^{b_i 2^i}. \tag{1}$$

We compute the matrices $A^2, A^4, \ldots, A^{2^{\lfloor \log k \rfloor}}$ by successive squaring, and then carry out the matrix multiplications in Eq. (1). It is seen that this algorithm consists of at most $2\lfloor \log k \rfloor$ successive matrix multiplications and the total parallel time is $O(\log n \cdot \log k)$, using $O(n^3)$ processors.

1.2.5:

(a) Notice that

$$\begin{bmatrix} x(t+1) \\ x(t) \end{bmatrix} = \begin{bmatrix} a(t) & b(t) \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ x(t-1) \end{bmatrix}.$$

We define

$$A(t) = \begin{bmatrix} a(t) & b(t) \\ 1 & 0 \end{bmatrix},$$

to obtain

$$\begin{bmatrix} x(n) \\ x(n-1) \end{bmatrix} = A(n-1)A(n-2)\cdots A(1)A(0) \begin{bmatrix} x(0) \\ x(-1) \end{bmatrix}. \tag{1}$$

This reduces the problem to the evaluation of the product of n matrices of dimensions 2×2, which can be done in $O(\log n)$ time, using $O(n)$ processors [cf. Exercise 1.2.3(b)].

(b) Similarly with Eq. (1), we have

$$\begin{bmatrix} x(n) \\ x(n-1) \end{bmatrix} = D \begin{bmatrix} x(1) \\ x(0) \end{bmatrix}, \tag{2}$$

where $D = A(n-1)A(n-2)\cdots A(1)$. This is a linear system of two equations in the four variables $x(0)$, $x(1)$, $x(n-1)$, $x(n)$. Furthermore, the coefficients of these equations can be computed in $O(\log n)$ time as in part (a). We fix the values of $x(0)$ and $x(n-1)$ as prescribed, and we solve for the remaining two variables using Cramer's rule (which takes only a constant number of arithmetic operations). This would be the end of the solution, except for the possibility that the system of equations being solved is singular (in which case Cramer's rule breaks down), and we must ensure that this is not the case. If the system is singular, then either there exists no solution for $x(1)$ and $x(n)$, or there exist several solutions. The first case is excluded because we have assumed

the existence of a sequence $x(0), \ldots, x(n)$ compatible with the boundary conditions on $x(0)$ and $x(n-1)$, and the values of $x(1)$, $x(n)$ corresponding to that sequence must also satisfy Eq. (2). Suppose now that Eq. (2) has two distinct solutions $\big(x^1(1), x^1(n)\big)$ and $\big(x^2(1), x^2(n)\big)$. Consider the original difference equation, with the two different initial conditions $\big(x(0), x^1(1)\big)$ and $\big(x(0), x^2(1)\big)$. By solving the difference equation we obtain two different sequences, both of which satisfy Eq. (2) and both of which have the prescribed values of $x(0)$ and $x(n-1)$. This contradicts the uniqueness assumption in the statement of the problem and concludes the proof.

1.2.6:

We first compute $x^2, x^3, \ldots, x^{n-1}, x^n$, and then form the inner product of the vectors $(1, x, x^2, \ldots, x^n)$ and $(a_0, a_1, \ldots, a_n)$. The first stage is no harder than the prefix problem of Exercise 1.2.3(a). (Using the notation of Exercise 1.2.3, we are dealing with the special case where $a_i = x$ for each i.) Thus, the first stage can be performed in $O(\log n)$ time. The inner product evaluation in the second stage can also be done in $O(\log n)$ time. (A better algorithm can be found in [MuP73]).

1.2.7:

(a) Notice that the graph in part (a) of the figure is a subgraph of the dependency graph of Fig. 1.2.12. In this subgraph, every two nodes are neighbors and, therefore, different colors have to be assigned to them. Thus, four colors are necessary.

(b) See part (b) of the figure.

(c) Assign to each processor a different "column" of the graph. *Note:* The result of part (c) would not be correct if the graph had more than four rows.

SECTION 1.3

1.3.1:

Let k_A and k_B be as in the hint. Then, based on the rules of the protocol, the pair (k_A, k_B) changes periodically as shown in the figure. (To show that this figure is correct, one must argue that at state $(1,1)$, A cannot receive a 0, at state $(1,0)$, which would move the state to $(0,1)$, B cannot

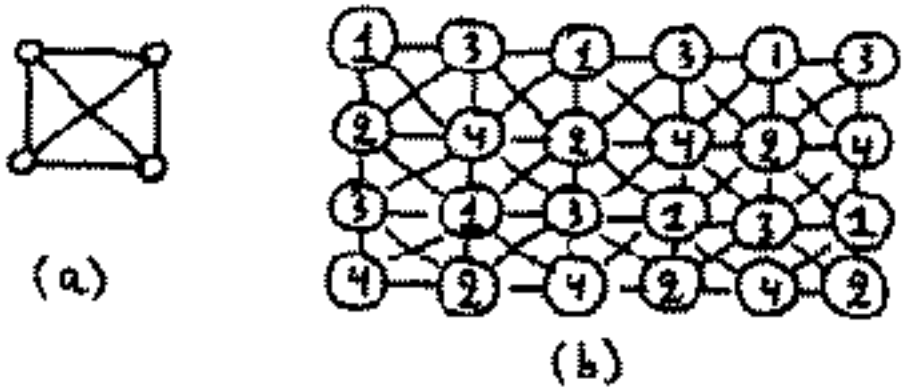

Figure For Exercise 1.2.7.

receive a 1, etc.). B stores a packet numbered 0 when the state changes from $(1,1)$ to $(1,0)$, and stores a packet numbered 1 when the state changes from $(0,0)$ to $(0,1)$. Thus, B alternates between storing a packet numbered 0 and a packet numbered 1. It follows that packets are received in order. Furthermore each packet is received only once, because upon reception of a 0 following a 1, B will discard all subsequent 0's and will only accept the first subsequent 1 (a similar argument holds also with the roles of 0 and 1 reversed). Finally, each packet will eventually be received by B, that is, the system cannot stop at some state (assuming an infinite packet supply at A). To see this, note that at state $(1,1)$, A keeps transmitting packet 0 (after a timeout of Δ) up to the time A receices a 0 from B. Therefore, B will eventually receive one of these 0's switching the state to $(1,0)$. Similarly, at state $(1,0)$, B keeps transmitting 0's in response to the received 0's, so eventually one of these 0's will be received by A, switching the state to $(0,0)$, and the process will be repeated with the roles of 0 and 1 reversed.

1.3.2:

(a) We claim that the following algorithm completes a multinode broadcast in $p-1$ time units.

At the first time unit, each node sends its packet to all its neighbors. At every time unit after the first, each processor i considers each of its incident links (i,j). If i has received a packet that it has neither sent already to j, nor it has yet received from j, then i sends such a packet on link (i,j). If i does not have such a packet, it sends nothing on (i,j).

For each link (i,j), let $T(i,j)$ be the set of nodes whose unique simple walk to i on the tree passes through j, and let $n(i,j)$ be the number of nodes in the set $T(i,j)$. We claim that in the preceding algorithm, each node i receives from each neighbor j a packet from one of the nodes of $T(i,j)$ at

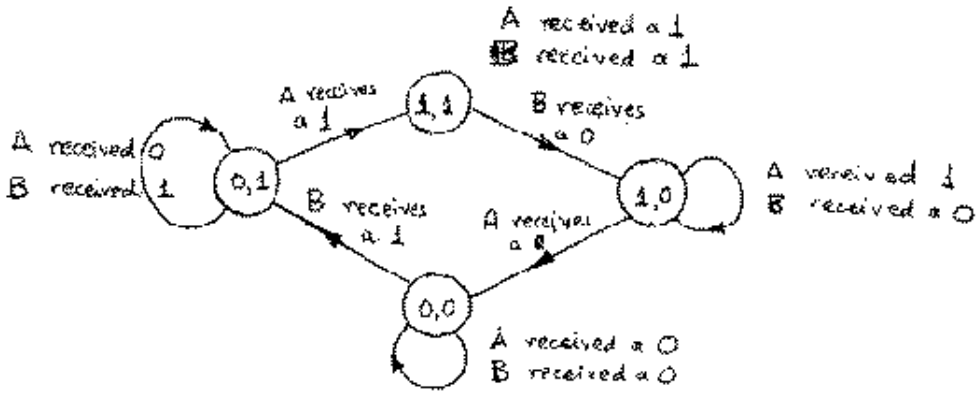

Figure For Exercise 1.3.1. State transition diagram for the stop-and-wait protocol.

each of the time units $1, 2, \ldots, n(i,j)$, and as a result, the multinode broadcast is completed in $\max_{(i,j)} n(i,j) = p - 1$ time units.

We prove our claim by induction. It is true for all links (i,j) with $n(i,j) = 1$, since all nodes receive the packets of their neighbors at the first time unit. Assuming that the claim is true for all links (i,j) with $n(i,j) = k$, we will show that the claim is true for all (i,j) with $n(i,j) = k+1$. Indeed let (i,j) be such that $n(i,j) = k+1$ and let $j_1, j_2, \ldots, j_m$ be the neighbors of j other than i. Then we have

$$T(i,j) = (i,j) \cup \left(\cup_{v=1}^{m} T(j,j_v)\right)$$

and therefore

$$n(i,j) = 1 + \sum_{v=1}^{m} n(j,j_v).$$

Node i receives from j the packet of j at the first time unit. By the induction hypothesis, j has received at least t packets by the end of t time units, where $t = 1, 2, \ldots, \sum_{v=1}^{m} n(j,j_v)$. Therefore, j has a packet to send to i from some node in $\cup_{v=1}^{m} T(j,j_v) \subset T(i,j)$ at each time unit $t = 2, 3, \ldots, 1 + \sum_{v=1}^{m} n(i,j_v)$. By the rules of the algorithm, i receives such a packet from j at each of these time units, and the induction proof is complete.

(b) Let T_1 and T_2 be the subtrees rooted at the two neighbors of the root node. In a total exchange, all of the $\Theta(p^2)$ packets originating at nodes of T_1 and destined for nodes of T_2 must be transmitted by the root node. Therefore any total exchange algorithm requires $\Omega(p^2)$ time. We can also perform a total exchange by carrying out p successive multinode broadcasts requiring $p(p-1)$ time units as per part (a). Therefore an optimal total exchange requires $\Theta(p^2)$ time units. [The alternative algorithm, based on the mapping of a unidirectional ring on the binary balanced tree (cf. Fig. 1.3.29)

is somewhat slower, but also achieves the optimal order of time for a total exchange.]

1.3.3:

Let $(00\cdots0)$ be the identity of node A and $(10\cdots0)$ be the identity of node B. The identity of an adjacent node of A, say C, has an identity with one unity bit, say the ith from the left, and all other bits zero. If $i = 1$, then $C = B$ and node A is the only node in S_B that is a neighbor of C. If $i > 1$, then the node with bits 1 and i unity is the only node in S_B that is a neighbor of C.

1.3.4:

(a) Consider a particular direction for traversing the cycle. The identity of each successive node of the cycle differs from the one of its predecessor node by a single bit, so going from one node to the next on the cycle corresponds to reversing a single bit. After traversing the cycle once, ending up at the starting node, each bit must have been reversed an even number of times. Therefore, the total number of bit reversals, which is the number of nodes in the cycle, is even.

(b) For p even, the ring can be mapped into a $2 \times 2^{d-1}$ mesh, which in turn can be mapped into a d-cube. If p is odd and a mapping of the p-node ring into the d-cube existed, we would have a cycle on the d-cube with an odd number of nodes contradicting part (a).

1.3.5:

See the figure.

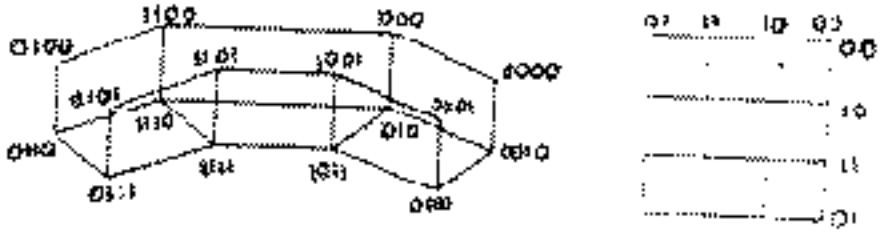

Figure Solution of Exercise 1.3.5.

1.3.6:

Follow the given hint. In the first phase, each node $(x_1, x_2, \ldots, x_d)$ sends to each node of the form $(x_1, y_2, \ldots, y_d)$ the $p^{1/d}$ packets destined for nodes $(y_1, y_2, \ldots, y_d)$, where y_1 ranges over $1, 2, \ldots, p^{1/d}$. This involves $p^{1/d}$ total exchanges in $(d-1)$-dimensional meshes with $p^{(d-1)/d}$ nodes each. By the induction hypothesis, each of these total exchanges takes time $O\left((p^{(d-1)/d})^{d/(d-1)}\right) = O(p)$, for a total of $p^{1/d}O(p) = O\left(p^{(d+1)/d}\right)$ time. At the end of phase one, each node $(x_1, x_2, \ldots, x_d)$ has $p^{(d-1)/d}p^{1/d}$ packets, which must be distributed to the $p^{1/d}$ nodes obtained by fixing $x_2, x_3, \ldots, x_d$, that is, nodes $(y_1, x_2, \ldots, x_d)$ where $y_1 = 1, 2, \ldots, p^{1/d}$. This can be done with $p^{(d-1)/d}$ total exchanges within one-dimensional arrays with $p^{1/d}$ nodes. Each total exchange takes $O\left((p^{1/d})^2\right)$ time (by the results of Section 1.3.4), for a total of $p^{(d-1)/d}O\left(p^{2/d}\right) = O\left(p^{(d+1)/d}\right)$ time.

1.3.7:

See the hint.

1.3.8:

Without loss of generality, assume that the two node identities differ in the rightmost bit. Let C_1 (or C_2) be the $(d-1)$-cubes of nodes whose identities have zero (or one, respectively) as the rightmost bit. Consider the following algorithm: at the first time unit, each node starts an optimal single node broadcast of its own packet within its own $(d-1)$-cube (either C_1 or C_2), and also sends its own packet to the other node. At the second time unit, each node starts an optimal single node broadcast of the other node's packet within its own $(d-1)$-cube (and using the same tree as for the first single node broadcast). The single node broadcasts takes $d-1$ time units each, and can be pipelined because they start one time unit apart and they use the same tree. Therefore the second single node broadcast is completed at time d, at which time the two-node broadcast is accomplished.

1.3.9:

(a) Consider the algorithm of the hint, where each node receiving a packet not destined for itself, transmits the packet at the next time unit on the next link of the path to the packet's destination. This algorithm accomplishes the single node scatter in $p-1$ time units. There is no faster algorithm for single node scatter, since s has $p-1$ packets to transmit, and can transmit at most one per time unit.

(b) Consider the algorithm of the hint, where each node receiving a packet not destined for itself, transmits the packet at the next time unit on the next link of the path to the packet's destination. Then s starts transmitting its last packet to the subtree T_i at time $N_i - 1$, and all nodes receive their packet at time N_i. (To see the latter, note that all packets destined for the nodes of T_i that are k links away from s are sent before time $N_i - k$, and each of these packets completes its journey in exactly k time units.) Therefore all packets are received at their respective destinations in $\max\{N_1, N_2, \ldots, N_r\}$ time units.

(c) We will assume without loss of generality that $s = (00\cdots0)$ in what follows. To construct a spanning tree T with the desired properties, let us consider the equivalence classes R_{kn} introduced in Section 1.3.4 in connection with the multinode broadcast problem. As in Section 1.3.4, we order the classes as

$$(00\cdots0)R_{11}R_{21}\cdots R_{2n_2}\cdots R_{k1}\cdots R_{kn_k}\cdots R_{(d-2)1}\cdots R_{(d-2)n_{d-2}}R_{(d-1)1}(11\cdots1)$$

and we consider the numbers $n(t)$ and $m(t)$ for each identity t, but for the moment, we leave the choice of the first element in each class R_{kn} unspecified. We denote by m_{kn} the number $m(t)$ of the first element t of R_{kn} and we note that this number depends only on R_{kn} and not on the choice of the first element within R_{kn}.

We say that class $R_{(k-1)n'}$ is *compatible* with class R_{kn} if $R_{(k-1)n'}$ has d elements (node identities) and there exist identities $t' \in R_{(k-1)n'}$ and $t \in R_{kn}$ such that t' is obtained from t by changing some unity bit of t to a zero. Since the elements of $R_{(k-1)n'}$ and R_{kn} are obtained by left shifting the bits of t' and t, respectively, it is seen that for every element x' of $R_{(k-1)n'}$ there is an element x of R_{kn} such that x' is obtained from x by changing one of its unity bits to a zero. The reverse is also true, namely that for every element x of R_{kn} there is an element x' of $R_{(k-1)n'}$ such that x is obtained from x' by changing one of its zero bits to unity. An important fact for the subsequent spanning tree construction is that for every class R_{kn} with $2 \leq k \leq d-1$, there exists a compatible class $R_{(k-1)n'}$. Such a class can be obtained as follows: take any identity $t \in R_{kn}$ whose rightmost bit is a one and its leftmost bit is a zero. Let σ be a string of consecutive zeros with maximal number of bits and let t' be the identity obtained from t by changing to zero the unity bit immediately to the right of σ. [For example, if $t = (0010011)$ then $t' = (0010001)$ or $t' = (0000011)$, and if $t = (0010001)$ then $t' = (0010000)$.] Then the equivalence class of t' is compatible with R_{kn} because it has d elements [$t' \neq (00\cdots0)$ and t' contains a unique substring of consecutive zeros with maximal number of bits, so it cannot be replicated by left rotation of less than d bits].

The spanning tree T with the desired properties is constructed sequentially by adding links incident to elements of the classes R_{kn} as follows (see the figure):

Initially T contains no links. We choose arbitrarily the first element of class R_{11} and we add

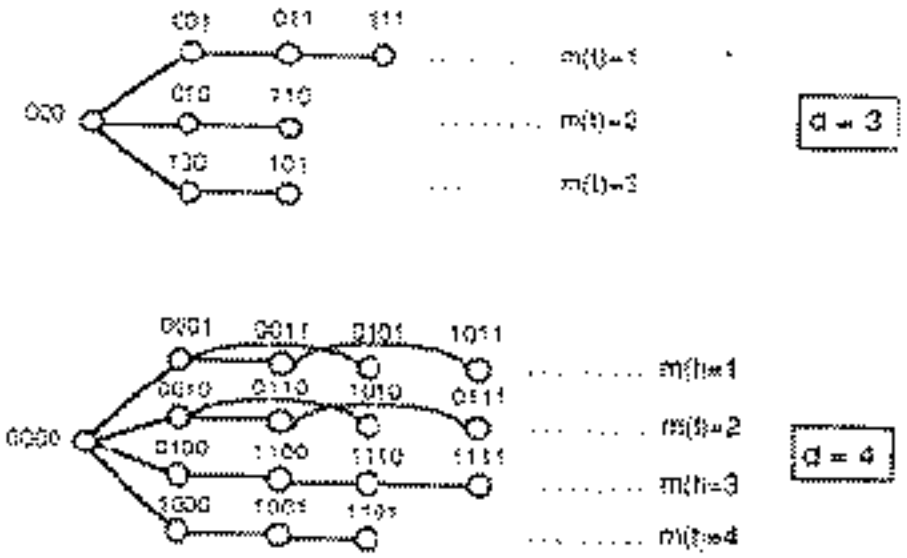

Figure For Exercise 1.3.9(c). Spanning tree construction for optimal single node scatter for $d = 3$ and $d = 4$, assuming transmission along all incident links of a node is allowed.

to T the links connecting $(00\cdots0)$ with all the elements of R_{11}. We then consider each class R_{kn} $(2 \le k \le d-1)$ one-by-one in the order indicated above, and we find a compatible class $R_{(k-1)n'}$ and the element $t' \in R_{(k-1)n'}$ such that $m(t') = m_{kn}$ (this is possible because $R_{(k-1)n'}$ has d elements). We then choose as first element of R_{kn} an element t such that t' is obtained from t by changing one of its unity bits to a zero. Since $R_{(k-1)n'}$ has d elements and R_{kn} has at most d elements, it can be seen that, for any x in R_{kn}, we have $m(x') = m(x)$, where x' is the element of $R_{(k-1)n'}$ obtained by shifting t' to the left by the same amount as needed to obtain x by shifting t to the left. Moreover x' can be obtained from x by changing some unity bit of x to a zero. We add to T the links (x', x), for all $x \in R_{kn}$ (with x' defined as above for each x). After exhausting the classes R_{kn}, $2 \le k \le d-1$, we finally add to T the link $\big(x, (11\cdots1)\big)$, where x is the element of $R_{(d-1)1}$ with $m(x) = m(11\cdots1)$.

The construction of T is such that each node $x \ne (00\cdots0)$ is in the subtree $T_{m(x)}$. Since there are at most $\lceil (2^d-1)/d \rceil$ nodes x having the same value of $m(x)$, each subtree contains at most $\lceil (2^d-1)/d \rceil$ nodes. Furthermore, the number of links on the path of T connecting any node and $(00\cdots0)$ is the corresponding Hamming distance. Hence, T is also a shortest path tree from $(00\cdots0)$, as desired.

1.3.10:

(a) See the hint.

(b) See the hint. (To obtain the equality

$$\sum_{k=1}^{d} k\binom{d}{k} = d2^{d-1}$$

write $(x+1)^d = \sum_{k=0}^{d} \binom{d}{k} x^k$, differentiate with respect to x, and set $x = 1$.)

(c) Let T_d be the optimal total exchange time for the d-cube, assuming transmission along at most one incident link for each node. We have $T_1 = 1$. Phases one and three take time T_d, while phase two takes time 2^d. By carrying out these phases sequentially we obtain $T_{d+1} = 2T_d + 2^d$, and it follows that $T_d = d2^{d-1}$.

1.3.11:

For the lower bounds see the hint. For a single node broadcast upper bound, use the tree of Fig. 1.3.16 (this is essentially the same method as the one described in the hint). For a multinode broadcast, use the imbedding of a ring into the hypercube.

1.3.12:

We prove that S_k has the characterization stated in the hint by induction. We have $S_1 = \{-1, 0, 1\}$. If S_{k-1} is the set of all integers in $[-(2^{k-1} - 1), (2^{k-1} + 1)]$, then S_k is the set $2S_{k-1} + \{-1, 0, 1\}$, which is the set of all integers in $[-(2^k - 1), (2^k + 1)]$.

Using the characterization of the hint, an integer $m \in [1, 2^d - 1]$ can be represented as

$$m = u(d-1) + 2u(d-2) + 2^2u(d-3) + \cdots + 2^{d-1}u(0),$$

where $u(k) \in \{-1, 0, 1\}$. Thus a generalized vector shift of size m can be accomplished by successive shifts on the level k subrings for all k such that $u(d-1-k) \neq 0$, where the shift is forward or backward depending on whether $u(d-1-k)$ is equal to 1 or -1, respectively.

1.3.14:

(Due to George D. Stamoulis.) We will show that the order of time taken by an optimal algorithm is $\Theta(d)$ for single node broadcast, $\Theta(d2^d)$ for multinode broadcast, and $\Theta(d^2 2^d)$ for a total exchange.

By using any shortest path spanning tree, the single node broadcast may be accomplished in D time units [where D is the diameter of the cube-connected cycles (or CCC for short) graph]. Because of symmetry, D equals the maximum over all nodes $(j, x_1 \dots x_d)$ of the (minimum) distance between nodes $(1, 0 \dots 0)$ and $(j, x_1 \dots x_d)$. If $x_1 = \cdots = x_d = 0$, then the two nodes are in the same ring, which implies that their (minimum) distance is at most $\lceil \frac{d-1}{2} \rceil$. Furthermore, we consider a node $(j, x_1 \dots x_d)$ such that $x_{i_1} = \cdots = x_{i_n} = 1$ (where $1 \leq n \leq d$ and $i_1 < \cdots < i_n$) and the remaining x_i's equal 0. This node is accessible from node $(1, 0 \dots 0)$ through the following path:

$$(1, 0 \dots 0) \rightarrow \cdots \rightarrow (i_1, 0 \dots 0) \overset{\text{1-st bit-flip}}{\rightarrow} (i_1, 0 \dots 010 \dots 0) \rightarrow \cdots \rightarrow$$

$$\rightarrow (i_2, 0 \dots 010 \dots 0) \overset{\text{2-nd bit-flip}}{\rightarrow} (i_2, 0 \dots 010 \dots 010 \dots 0) \rightarrow \cdots \rightarrow (i_n, x_1 \dots x_d) \rightarrow \cdots \rightarrow (j, x_1 \dots x_d) \,. \quad (1)$$

All parts of this path that consist of links of the same ring are assumed to have the minimum possible length. The path presented in (1) consists of L links, where

$$L = \sum_{k=1}^{n} \big(\min\{ i_k - i_{k-1}, d - i_k + i_{k-1} \} + 1 \big) + \min\{ |i_n - j|, d - |i_n - j| \} \,, \quad (2)$$

with $i_0 \overset{\text{def}}{=} 1$ (recall the formula for the (minimum) distance between two nodes in a d-ring). It follows from (2) that

$$L \leq \sum_{k=1}^{n} \big(i_k - i_{k-1} + 1 \big) + \min\{ |i_n - j|, d - |i_n - j| \} = i_n + n - 1 + \min\{ |i_n - j|, d - |i_n - j| \} \,. \quad (3)$$

Using the inequalities $i_n \leq d$, $n \leq d$ and $\min\{|i_n - j|, d - |i_n - j|\} \leq \lceil \frac{d-1}{2} \rceil$, we obtain $L \leq 2d - 1 + \lceil \frac{d-1}{2} \rceil$. Combining this with the fact that any node in the same ring as $(1, 0 \dots 0)$ is at a distance of at most $\lceil \frac{d-1}{2} \rceil$ from it, we have $D \leq 2d - 1 + \lceil \frac{d-1}{2} \rceil$ (note that, for establishing this inequality, it was not necessary to deal exclusively with shortest paths; depending on the destination node, the path presented in (1) may or may not be a shortest one). Furthermore, in the case where the destination node is $(\lceil \frac{d-1}{2} \rceil + 1, 1 \dots 1)$, it is straightforward that the path in (1) attains the minimum possible length. Since in this case the path has length $2d - 1 + \lceil \frac{d-1}{2} \rceil$, we conclude that $D = 2d - 1 + \lceil \frac{d-1}{2} \rceil$, which is $\Theta(d)$.

A spanning tree (possibly, not of the shortest path type) that may be used for the optimal execution of a single node broadcast by node $(1, 0 \dots 0)$ (without loss of generality) may be constructed as follows: First, consider the subgraph consisting of the CCC links that correspond to the unbalanced spanning tree of the d-cube, the one rooted at node $(0 \dots 0)$ (*). Moreover, for each of the 2^d rings, append to this subgraph $d - 1$ of the ring's links. The link to be excluded is chosen in such a way

(*) This is the spanning tree in which bit-flips are performed in an increasing bit-index order (see also Fig. 1.3.16 of [BeT89]).

that each node of the ring is at a distance of at most $\lceil \frac{d-1}{2} \rceil$ from this node (of the ring) which is the end node of the "hypercube-tree" link incoming to the ring (i.e., this "hypercube-tree" link that is pointing from the root to the ring). It is straightforward to check that this construction leads to a spanning tree, with each node being accessible from the root through the path in (1).

We now consider the multinode broadcast case. During the multinode broadcast, each node receives $d2^d - 1$ packets. Since the number of incident links to any node is 3, we have

$$\frac{d2^d - 1}{3} \leq T_{MNB} \, ,$$

where T_{MNB} is the optimal time for the multinode broadacst. Therefore, T_{MNB} is $\Omega(d2^d)$. In what follows, we present a multinode broadcast algorithm which requires $\Theta(d2^d)$ time.

First, we observe that, for any $k \leq d$, k groups of d packets that are stored in different nodes of a d-ring may be broadcasted among the ring's nodes in at most $d\lceil \frac{d-1}{2} \rceil$ time units (recall that the optimal time for the multinode broadcast in a d-ring is $\lceil \frac{d-1}{2} \rceil$).

The algorithm is as follows:

First, 2^d multinode broadcasts take place (in parallel) within each of the rings. This takes $\lceil \frac{d-1}{2} \rceil$ time units. Now we introduce the term "super-node" to denote each of the 2^d rings. After the first phase of the algorithm, the situation may alternatively be visualized as follows: we have 2^d "super-nodes" (connected in a hypercube), with each of them broadcasting d packets to the others. This may be accomplished under the following rules:

A) Every "super-node" uses the same paths as in the optimal multinode broadcast in the d-cube, and transmits packets in groups of d.

B) Following d successive time units of transmissions in the "hypercube" links, the groups of packets just received in a "super-node" are broadcasted among its d nodes. This takes $d\lceil \frac{d-1}{2} \rceil$ time units. During this time interval no transmissions take place in the "hypercube" links; such transmissions resume immediately after this interval.

The algorithm presented above requires time $T = \lceil \frac{d-1}{2} \rceil + (d + d\lceil \frac{d-1}{2} \rceil)\lceil \frac{2^d-1}{d} \rceil$ (actually, this algorithm may be further parallelized by simultaneously performing "hypercube" transmissions and transmissions within the rings). Thus, T is $\Theta(d2^d)$. Since $T_{MNB} \leq T$, we conclude that T_{MNB} is $O(d2^d)$. This, together with the fact that T_{MNB} is $\Omega(d2^d)$, implies that T_{MNB} is $\Theta(d2^d)$.

We now consider the case of a total exchange. Let S_0 (S_1) be the set of nodes (m, j) such that the first bit of j is 0 (1, respectively). We have $|S_0| = |S_1| = d2^{d-1}$. Moreover, there are $N = 2^{d-1}$ links connecting nodes of S_0 with nodes of S_1. Thus, we obtain

$$T_{EX} \geq \frac{|S_0||S_1|}{2^{d-1}} = d^2 2^{d-1} \, ,$$

where T_{EX} is the time required by the optimal total exchange algorithm. Therefore, T_{EX} is $\Omega(d^2 2^d)$. In what follows, we present an algorithm which requires time $\Theta(d^2 2^d)$.

First, we briefly present a total exchange algorithm for the d-cube [SaS85]. We denote as kth dimension the set of links of type k (a type k link is a link between two nodes the identities of which differ in the kth bit). A total exchange may be accomplished in d successive phases, as follows: during the ith phase each packet that must cross the ith dimension (due to the fact that the identity of its destination node differs from that of its origin node in the ith bit) takes this step. It may be proved by induction that just before the ith phase each node has 2^d packets in its buffer, for $i = 1, \ldots, d$; these packets are originating from 2^{i-1} different nodes (including the node considered), with each of these nodes contributing 2^{d-i+1} packets (*). Each phase lasts for 2^{d-1} time units; this follows from the fact that exactly half of the packets that are stored in a node just before the ith phase have to flip the ith bit (in order to reach their destinations). Therefore, under this algorithm, the total exchange is performed in time $d2^{d-1}$. In the case where each node transmits to each of the other nodes exactly d packets (instead of one, which is the case usually considered) a modified version of the previous algorithm may be used. Indeed, d instances of the above total exchange algorithm may be executed in parallel. Each node labels its packets arbitrarily, with the permissible label values being $1, \ldots, d$; any two packets originating from the same node are assigned different labels. Packets labelled 1 follow the same paths as in the above total exchange algorithm. Packets labelled 2 take part in another total exchange, which is performed similarly as in the above algorithm; the only difference is that these packets cross dimensions in the order $2, 3, \ldots, d, 1$ (that is, during the ith phase these packets may only cross the $(i \bmod d+1)$st dimension). Similarly, during the ith phase, packets labelled m may only cross the $((i+m-2) \bmod d+1)$st dimension. It follows that, during each of the d phases, packets of different labels cross links of different dimensions. Therefore, no conflicts occur, which implies that the total exchange involving d packets per ordered pair of nodes may be accomplished in $d2^{d-1}$ time units under the previous algorithm (in fact, this is the minimum time for this task). This algorithm may be modified so that it may be used for a total exchange in the CCC, with the time required being $\Theta(d^2 2^d)$.

Each "super-node" sends d^2 packets to each of the other "super-nodes". All packets originating from nodes (m, j) are labelled m, for $m = 1, \ldots, d$. First, we have d successive phases. Each packet that is destined for some node in the same ring as its origin is not transmitted during these phases. In particular, during the ith phase, packets labelled m may only cross the $((i+m-2) \bmod d+1)$th dimension of the "hypercube" links; following the necessary "hypercube" transmissions, each packet takes exactly one clockwise step in the ring where it resides (that is, it changes its current ring-index

(*) For convenience, we assume that each node stores a null packet that is destined for itself.

from m^* to (m^* mod $d+1$)), in order to be ready to cross the corresponding "hypercube" dimension of the next phase, if necessary (note that these steps are taken in the dth phase, even though it is the last one). Each of these d phases may be accomplished in $d2^{d-1} + d(2^d - 1)$ time units. By the end of the dth phase, node (m, j) has received all packets originating from all nodes (m, j') with $j \neq j'$ and destined for all nodes (m', j), with $m' = 1, \ldots, d$ (i.e., destined for all nodes in the same ring as node (m, j)). Recalling that nodes within the same ring also send packets to each other, we see that it remains to perform a total exchange within each ring, involving 2^d packets per ordered pair of nodes. Since the optimal total exchange in a d-ring takes time $\frac{1}{2}\lfloor\frac{d}{2}\rfloor(\lfloor\frac{d}{2}\rfloor + 1)$, the total exchanges within the rings may be accomplished (in parallel) in time $2^{d-1}\lfloor\frac{d}{2}\rfloor(\lfloor\frac{d}{2}\rfloor + 1)$. Therefore, the above algorithm for a total exchange in the CCC requires time $T = 3d^2 2^{d-1} - d^2 + 2^{d-1}\lfloor\frac{d}{2}\rfloor(\lfloor\frac{d}{2}\rfloor + 1)$, which is $\Theta(d^2 2^{d-1})$ (in fact, this algorithm may be further parallelized by simultaneously performing "hypercube" transmissions and transmissions within the rings). Since $T_{EX} \leq T$, it follows that T_{EX} is $O(d^2 2^d)$. Recalling that T_{EX} is $\Omega(d^2 2^d)$, we conclude that T_{EX} is $\Theta(d^2 2^d)$.

1.3.16:

Consider the following algorithm for transposing B, that is, move each b_{ij} from processor (i, j) to processor (j, i): for all $j = 1, 2, \ldots, n$, do in parallel a single node gather along the column (linear array) of n processors $(1, j)$, $(2, j)$, ..., (n, j) to collect b_{ij}, $i = 1, \ldots, n$, at processor (j, j). This is done in $n - 1$ time units by the linear array results. Then for all $j = 1, \ldots, n$, do in parallel a single node scatter along the row (linear array) of n processors $(j, 1)$, $(j, 2)$, ..., (j, n) to deliver b_{ij}, $i = 1, \ldots, n$, at processor (j, i). This is done again in $n - 1$ time units by the linear array results. Thus the matrix transposition can be accomplished in $2(n-1)$ time units. Now to form the product AB', we can transpose B in $2(n - 1)$ time units as just described, and we can then use the matrix multiplication algorithm of Fig. 1.3.27, which requires $O(n)$ time. The total time is $O(n)$ as required.

1.3.17:

Follow the hint. Note that each of the transfers indicated in Fig. 1.3.34(b) takes 2 time units, so the total time for the transposition is $4 \log n$.

1.3.18:

(a) For each k, the processors (i, j, k), $i, j = 1, 2, \ldots, n$, form a hypercube of n^2 processors, so the algorithm of Exercise 3.17 can be used to transpose A within each of these hypercubes in parallel in $4 \log n$ time units.

(b) For all (i,j), the processors (i,j,j) hold initially a_{ij} and can broadcast it in parallel on the hypercube of n nodes (i,k,j), $k=1,\ldots,n$ in $\log n$ time units.

1.3.19:

Using a spanning tree of diameter r rooted at the node, the transmission of the mth packet starts at time $(m-1)(w+1/m)$ and its broadcast is completed after time equal to r link transmissions. Therefore the required time is

$$T(m)=(m-1+r)(w+1/m).$$

We have that $T(m)$ is convex for $m>0$ and its first derivative is

$$\frac{dT(m)}{dm}=w+\frac{1}{m}-\frac{m-1+r}{m^2}=w-\frac{r-1}{m^2}.$$

It follows that $dT(m)/dm=0$ for $m=\sqrt{(r-1)/w}$. If $w>r-1$, then $m=1$ is optimal. Otherwise, one of the two values

$$\left\lceil\sqrt{\frac{r-1}{w}}\right\rceil, \qquad \left\lfloor\sqrt{\frac{r-1}{w}}\right\rfloor$$

is optimal.

1.3.20:

(a) Let c_i be the ith column of C. An iteration can be divided in four phases: in the first phase processor i forms the product c_ix_i, which takes m time units. In the second phase, the sum $\sum_{i=1}^n c_ix_i$ is accumulated at a special processor. If pipelining is not used (cf. Problem 3.19), this takes $(d+1)m\log n$ time in a hypercube and $(d+1)m(n-1)$ time in a linear array. If pipelining is used and overhead is negligible, this takes $(d+1)(m+\log n)$ time in a hypercube and $(d+1)(m+n-1)$ time in a linear array. In the third phase, the sum $\sum_{i=1}^n c_ix_i$ is broadcast from the special processor to all other processors. If pipelining is not used, this takes $dm\log n$ time in a hypercube and $dm(n-1)$ time in a linear array. If pipelining is used and overhead is negligible, this takes $d(m+\log n)$ time in a hypercube and $d(m+n-1)$ time in a linear array. Finally in the fourth phase, each processor i has to form the inner product $c_i'\left(\sum_{j=1}^n c_jx_j\right)$ and add b_i to form the ith coordinate of $C'Cx+b$. this takes $m+1$ time units. The total time is

$$2m+1+(2d+1)m\log n \qquad \text{in a hypercube with no pipelining}$$

$$2m+1+(2d+1)m(n-1) \qquad \text{in a linear array with no pipelining}$$

$$2m + 1 + (2d + 1)(m + \log n) \qquad \text{in a hypercube with pipelining}$$

$$2m + 1 + (2d + 1)(m + n - 1) \qquad \text{in a linear array with pipelining.}$$

(b) Let p_i be the ith row of $C'C$. processor i must form the inner product $p_i'x$ (n time units), add b_i, and broadcast the result to all other processors. The total time is

$$n + 1 + d\left\lceil \frac{n-1}{\log n} \right\rceil \qquad \text{in a hypercube}$$

$$n + 1 + d(n - 1) \qquad \text{in a linear array.}$$

(c) If $m << n$ and a hypercube is used, the implementation of part (a) is superior. Otherwise the implementation of part (b) is superior.

1.3.21:

Each processor i computes the ith coordinate $[Ax]_i$ by forming the inner product of the ith row of A and x using $O(r)$ arithmetic operations. Then, each processor i broadcasts $[Ax]_i$ to all other processors. This is a multinide broadcast requiring $O(n/\log n)$ time units, so the total time is $O\big(\max(n/\log n, r)\big)$.

If processor i stores instead the ith column of A and x_i, it can compute the products of x_i with the nonzero entries of the ith column in no more than r time units. The processors i can then accumulate the n terms of $[Ax]_i$ by a multinode accumulation in $n/\log n$ time units.

1.3.22:

For any two node identities t and z, we denote by $t \oplus z$ the node identity obtained by performing modulo 2 addition of the jth bit of t and z for $j = 1, 2, \ldots, \log p$. We also denote by e_k the node identity with all bits equal to zero except for the kth bit from the left, which is a one. Let $s_i(k)$ be the value held at processor i at the end of the kth stage and let $s_i(0) = a_i$. The algorithm can be expressed as

$$s_i(k) = s_i(k-1) + s_{i\oplus k}(k-1), \qquad \forall\ k = 1, 2, \ldots, \log p. \tag{$*$}$$

Let N_k be the set of node identities whose $\log p - k$ rightmost bits are zero, e.g.,

$$N_1 = \{(00\cdots00), (00\cdots01)\}, \quad N_2 = \{(00\cdots000), (00\cdots001), (00\cdots010), (00\cdots011)\},$$

etc. Note that $N_{\log p}$ is the set of all node identities. Then we can prove by induction that

$$s_i(k) = \sum_{n\in N_k} a_{i\oplus n}, \qquad \forall\ i,\ k. \tag{$**$}$$

Indeed, using Eq. $(*)$ for $k = 1$ and the fact $s_i(0) = a_i$, we see that Eq. $(**)$ holds for $k = 1$. Assume that Eq. $(**)$ holds up to some k. We have, using Eqs. $(*)$ and $(**)$,

$$s_i(k+1) = \sum_{n \in N_k} a_{in} + \sum_{n \in N_k} a_{(i \oplus e_{k+1}) \oplus n} = \sum_{n \in N_{k+1}} a_{i \oplus n},$$

so Eq. $(**)$ holds with k replaced by $k+1$. Applying Eq. $(**)$ with $k = \log p$, we obtain the desired result.

1.3.23:

(a) The jth coordinate of $C'Cx$ is

$$\sum_{i=1}^{n} c_{ij} r_i,$$

where r_i is the ith coordinate of Cx,

$$r_i = \sum_{j=1}^{n} c_{ij} x_j.$$

Consider the following algorithm: the ith row processors (i, j), $j = 1, \ldots, n$, all obtain r_i in $\log n$ time using the algorithm of Exercise 3.22. Then the jth column processors (i, j), $i = 1, \ldots, n$, calculate $c_{ij} r_i$ and obtain the sum $\sum_{i=1}^{n} c_{ij} r_i$ in $\log n$ time units using the algorithm of Exercise 3.22. In the end this algorithm yields the jth coordinate of $C'Cx$ at the jth column processors (i, j), $i = 1, \ldots, n$.

(b) The algorithm of part (a) calculates a product of the form $C'CC'C \cdots C'Cx$ in $2m \log p$ time units, where m is the number of terms $C'C$ involved in the product, and stores the jth coordinate of the result at the jth column processors (i, j), $i = 1, \ldots, n$. Also, the algorithm of part (a) calculates a product of the form $CC'CC'C \cdots C'Cx$ in $(1+2m) \log p$ time units, where m is the number of terms $C'C$ involved in the product, and stores the jth coordinate of the result at the ith row processors (i, j), $j = 1, \ldots, n$. Combining these facts we see that if C is symmetric, $C^k x$ is calculated in $k \log p$ time units, with the ith coordinate of the product stored in the ith column processors or the ith row processors depending on whether k is even or odd.

1.3.24:

From the definition of the single node accumulation problem, we see that the packets of all nodes can be collected at a root node as a composite packet by combining them along a single node accumulation tree. The composite packet can then be broadcast from the root to all nodes, which is a single node broadcast.

SECTION 1.4

1.4.1:

We first consider the case of global synchronization. The time needed for each phase is equal to the maximum delay of the messages transmitted during that phase. Each processor is assumed to transmit d messages at each phase, for a total of nd messages. Thus, the expected time of each phase is equal to the expectation of the maximum of nd independent, exponentially distributed, random variables with mean 1. According to Prop. D.1 of Appendix D, the latter expectation is approximately equal to $\ln(nd)$ which leads to the estimate $G(k) = \Theta\big(k \log(nd)\big)$.

We now consider local synchronization. As in Subsection 1.4.1, we form the directed acyclic graph $G = (N, A)$ (cf. Fig. 1.4.3) with nodes $N = \{(t, i) \mid t = 1, \ldots, k+1;\ i = 1, \ldots, n\}$ and arcs of the form $\big((t, i), (t+1, j)\big)$ for each pair (i, j) of processors such that processor i sends a message to processor j (i.e., $j \in P_i$). We associate with each such arc in G a "length" which is equal to the delay of the message sent by processor i to processor j at time t. For any positive path p in this graph, we let M_p be its length, and we let $M = M_p$, where the maximum is taken over all paths. As discussed in Subsection 1.4.1, we have $L(k) = M$.

We now construct a particular path p that will lead to a lower bound on $E[L(k)]$. We first choose some i, j, such that the length of the arc $\big((1, i), (2, j)\big)$ is largest among all pairs (i, j) with $j \in P_i$. We take this to be our first arc. Its length is the maximum of nd independent exponential random variables and its expected length is $\Theta\big(\log(nd)\big) = \Omega(\log n)$. We then proceed as follows. Given a current node, we choose an outgoing arc whose length is largest, until we reach a node with no outgoing arcs. The length of the arc chosen at each stage is the maximum of d independent exponential random variables and, therefore, its expected length is $\Theta(\log d)$. There are $k-1$ arcs that are chosen in this way (since G has depth k). Thus, the expected length of the path we have constructed is $\Omega(\log n) + \Theta(k \log d)$.

We now derive an upper bound on M. Let us fix a positive path p. Its length p is equal to the sum of k independent exponential random variables with mean 1, and Prop. D.2 in Appendix D applies. In particular, we see that there exist positive constants α and C such that

$$\Pr(M_p \geq kc) \leq e^{-\alpha kc} = 2^{-\beta kc}, \qquad \forall k \geq 1,\ \forall c \geq C,$$

where $\beta > 0$ is chosen so that $e^{-\alpha} = 2^{-\beta}$. The total number of paths is nd^k. (We have a choice of the initial node and at each subsequent step we can choose one out of d outgoing arcs.) Thus, the

probability that *some* path p has length larger than kc is bounded by

$$\Pr(M \geq kc) \leq nd^k 2^{-\beta kc} = 2^{\log n + k\log d - \beta kc}, \qquad \forall k \geq 1,\ \forall c \geq C.$$

Let

$$D = \max\left\{\lceil C\rceil, \left\lceil \frac{\log n + k\log d}{k\beta}\right\rceil\right\}.$$

We then have,

$$\begin{aligned}
E[L(k)] = E[M] &\leq Dk + \sum_{c=D}^{\infty} \Pr\big(M \in [ck,(c+1)k]\big)\cdot(c+1)k \\
&\leq Dk + \sum_{c=D}^{\infty} \Pr\big(M \geq ck\big)\cdot(c+1)k \\
&\leq Dk + \sum_{c=D}^{\infty} 2^{\log n + k\log d - \beta Dk - \beta(c-D)k}(c+1)k \\
&\leq Dk + k\sum_{c=D}^{\infty} 2^{-\beta(c-D)}(c+1) \\
&= Dk + k\sum_{c=0}^{\infty} 2^{-\beta c}(c+D+1) \\
&= Dk + \frac{k(D+1)}{1-2^{-\beta}} + k\sum_{c=0}^{\infty} 2^{-\beta c}c \\
&= O(kD) = O(\log n + k\log d).
\end{aligned}$$

1.4.2:

The synchronous algorithm has the form

$$x_1[(k+1)(1+D)] = ax_1[k(1+D)] + bx_2[k(1+D)], \qquad k = 0,1,\ldots,$$

$$x_2[(k+1)(1+D)] = bx_1[k(1+D)] + ax_2[k(1+D)], \qquad k = 0,1,\ldots,$$

and we have

$$|x_i[k(1+D)]| \leq C(|a|+|b|)^k, \qquad i = 1,2,\ k = 0,1,\ldots$$

Therefore

$$|x_i(t)| \leq C(|a|+|b|)^{t/(1+D)} = C\rho_S^t,$$

where

$$\rho_S = (|a|+|b|)^{1/(1+D)}. \qquad (*)$$

For the asynchronous algorithm (since $D < 1$) we have

$$x_1(t+1) = ax_1(t) + bx_2(t-1),$$

$$x_2(t+1) = bx_1(t-1) + ax_2(t),$$

so by the results of Example 4.1,

$$|x_i(t)| \leq C\rho_A^t,$$

where ρ_A is the unique positive solution of

$$|a| + \frac{|b|}{\rho} = \rho.$$

It can be seen (using the fact $b \neq 0$) that $\rho_A > |a| + |b|$, while from $(*)$ it is seen that by making D sufficiently small, ρ_S can be made arbitrarily close to $|a| + |b|$.

1.4.3:

Let

$$C = \max\{|x_1(0)|, |x_2(0)|\}.$$

For $i = 1, 2$, we will prove the stronger relation

$$|x_i(t-k)| \leq C\rho_S^t, \qquad \forall\ t = n(D+1), n = 1, 2, \ldots, k = 0, 1, \ldots, D,$$

or equivalently

$$|x_i\big(n(D+1) - k\big)| \leq C(|a| + |b|)^n, \qquad \forall\ k = 0, 1, \ldots, D. \tag{$*$}$$

We use induction on n. For $n = 1$, this relation has the form

$$|x_i(D+1-k)| \leq C(|a| + |b|), \qquad \forall\ k = 0, 1, \ldots, D, \tag{$**$}$$

and can be proved by backward induction on k. Indeed for $k = D$ we have, (since it is assumed that $x_i(t) = x_i(0)$ for $t \leq 0$),

$$|x_1(1)| = |ax_1(0) + bx_2(-D)| \leq |a||x_1(0)| + |b||x_2(-D)| \leq C(|a| + |b|),$$

and similarly

$$|x_2(1)| \leq C(|a| + |b|).$$

Assuming that for $m \leq D - 1$ we have

$$|x_i(m)| \leq C(|a| + |b|),$$

we obtain using the fact $|a| + |b| < 1$,

$$\begin{aligned} |x_1(m+1)| &= |ax_1(m) + bx_2(-D)| \\ &\leq |a||x_1(m)| + |b||x_2(-D)| \leq |a|C(|a| + |b|) + |b|C \leq C(|a| + |b|), \end{aligned}$$

and similarly

$$|x_2(m+1)| \le C(|a|+|b|).$$

Thus, the induction proof of $(**)$ is complete.

Assume now that $(*)$ holds for some n. We will show that

$$|x_i\big((n+1)(D+1)-k\big)| \le C(|a|+|b|)^{n+1}, \qquad \forall\ k=0,1,\dots,D. \tag{$***$}$$

Again we use backward induction on k. We have for $k=D$, using $(*)$,

$$\begin{aligned}|x_1\big(n(D+1)+1\big)| &= |ax_1\big(n(D+1)\big)+bx_2\big(n(D+1)-D\big)| \\ &\le |a|C(|a|+|b|)^n + |b|(|a|+|b|)^n \le C(|a|+|b|)^{n+1},\end{aligned}$$

and similarly

$$|x_2\big(n(D+1)+1\big)| \le C(|a|+|b|)^{n+1}.$$

Assuming that for $m \le D-1$ we have

$$|x_i\big(n(D+1)+m\big)| \le C(|a|+|b|),$$

we obtain using the fact $|a|+|b|<1$,

$$\begin{aligned}|x_1\big(n(D+1)+m+1\big)| &= |ax_1\big(n(D+1)+m\big)+bx_2\big(n(D+1)-D\big)| \\ &\le |a||x_1\big(n(D+1)+m\big)| + |b||x_2\big(n(D+1)-D\big)| \\ &\le |a|C(|a|+|b|)^{n+1} + |b|C(|a|+|b|)^n \le C(|a|+|b|)^{n+1},\end{aligned}$$

and similarly

$$|x_2\big(n(D+1)+m+1\big)| \le C(|a|+|b|)^{n+1}.$$

Thus, the induction proof of $(***)$ and also of $(*)$ is complete.

CHAPTER 2

SECTION 2.1

2.1.1:

Let us define $k = n/p$. Consider the following scheme. Each processor i receives the values of $x_1, \ldots, x_{(i-1)k}$ from processor $(i-1)$ and forwards them in the same order to processor $i+1$. As soon as $x_{(i-1)k}$ is received, processor i is able to compute the values of $x_{(i-1)k+1}, \ldots, x_{ik}$ which are subsequently transmitted to processor $i+1$. Let t_i denote the time that processor i starts transmitting to processor $i+1$. We have $t_{i+1} = t_i + 1$. This is because if x_1 is received by processor i at time t_i, it is received one time unit later by $i+1$. Assuming $t_1 = 0$, we obtain $t_{p-1} = p-2$. Processor p receives $n-k$ messages. Thus, all information is received by processor p (and the algorithm terminates) at time $t_{p-1} + n - k$ and therefore $T(n,p) = p - 2 + n - n/p$. [Notice that $T(n,2) = n/2$, as expected.]

2.1.2:

In our original version of odd–even reduction, at the completion of the first stage of the algorithm, processors $2, 4, \ldots, \lfloor n/2 \rfloor$ have the coefficients of a reduced system of equations, satisfied by the variables $x_2, x_4, \ldots, x_{\lceil n/2 \rceil}$. However, we could at the same time eliminate the even variables and have processors $1, 3, \ldots$ obtain a reduced system for the odd–indexed variables $x_1, x_3, \ldots$. From then on, we are dealing with two independent tridiagonal systems with roughly half as many variables. Proceeding similarly, after $O(\log n)$ stages we have n independent equations, each one involving a different variable and the algorithm can terminate as soon as these variables are evaluated. See the figure for the case $n = 8$ and compare it with Fig. 2.1.3 in the text.

2.1.3:

Each processor is assigned $\Theta(\log n)$ variables and does the work of $\Theta(\log n)$ processors in the original

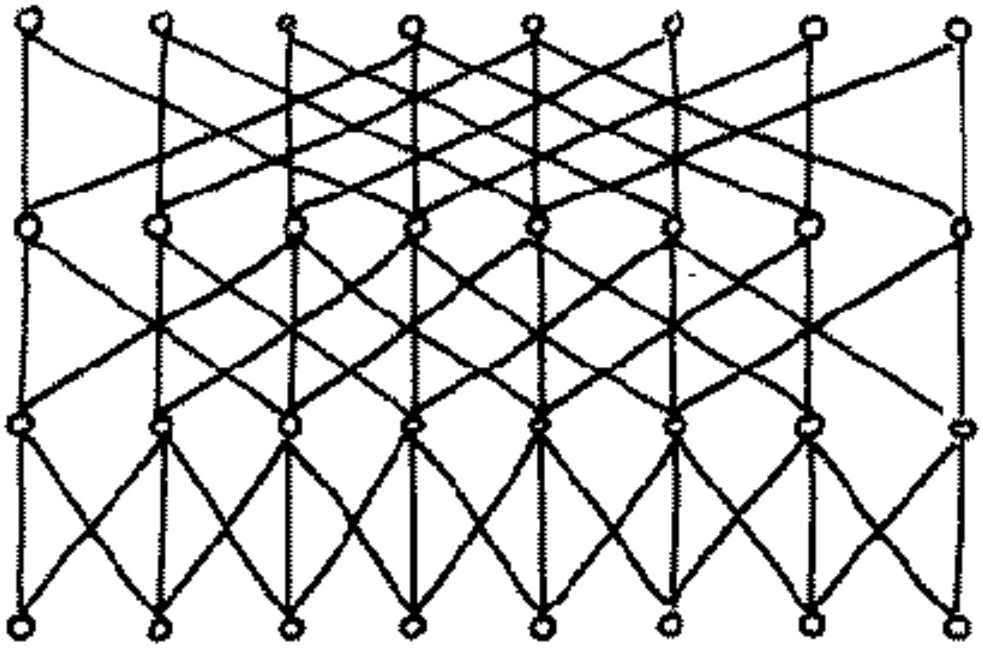

Figure For Exercise 2.1.2.

algorithm. In fact, this is exactly the case covered by Prop. 2.4 of Section 1.2. We have $p = O(n/\log n) = O(T_1/T_\infty)$, which implies that $T_p = O(T_\infty) = O(\log n)$.

SECTION 2.2

2.2.1:

If the maximum is zero, then all entries $C_{ji}^{(i-1)}$ with $j \geq i$ are zero. Thus, the lower left submatrix D of $C^{(i-1)}$, consisting of rows $i, i+1, \ldots, n$ and columns $i, i+1, \ldots, n$ has a zero column. It follows that D is singular and its determinant is zero. The determinant of $C^{(i-1)}$ is easily seen to be equal to $C_{11}^{(i-1)} \cdots C_{i-1,i-1}^{(i-1)}\mathrm{det(D)}$ and is also zero. Thus, $C^{(i-1)}$ is singular. It is easily seen that the matrices $M^{(j)}$ used for eliminating variables, as well as the permutation matrices P^{ij} are nonsingular. It follows that the original matrix A must have been singular.

2.2.2:

Each phase of the algorithm proceeds as in the figure. We now specify the timing of each message transmission so that the total execution time is $O(n)$. We assume that the time needed for a message transmission together with the computations performed by a processor at any given stage is no more than one unit. We refer to the communications and computations needed for computing $C^{(i)}$ from $C^{(i-1)}$ (illustrated in the figure) as the ith phase. Notice that during the ith phase, each processor (j,k) with $j \geq i$ and $k \geq i$ sends exactly one message to its neighbors $(j+1,k)$ (if $j < n$) and $(j,k+1)$ (if $k < n$). We let processor (j,k) send both of these messages at time $i+j+k$ (see the figure).

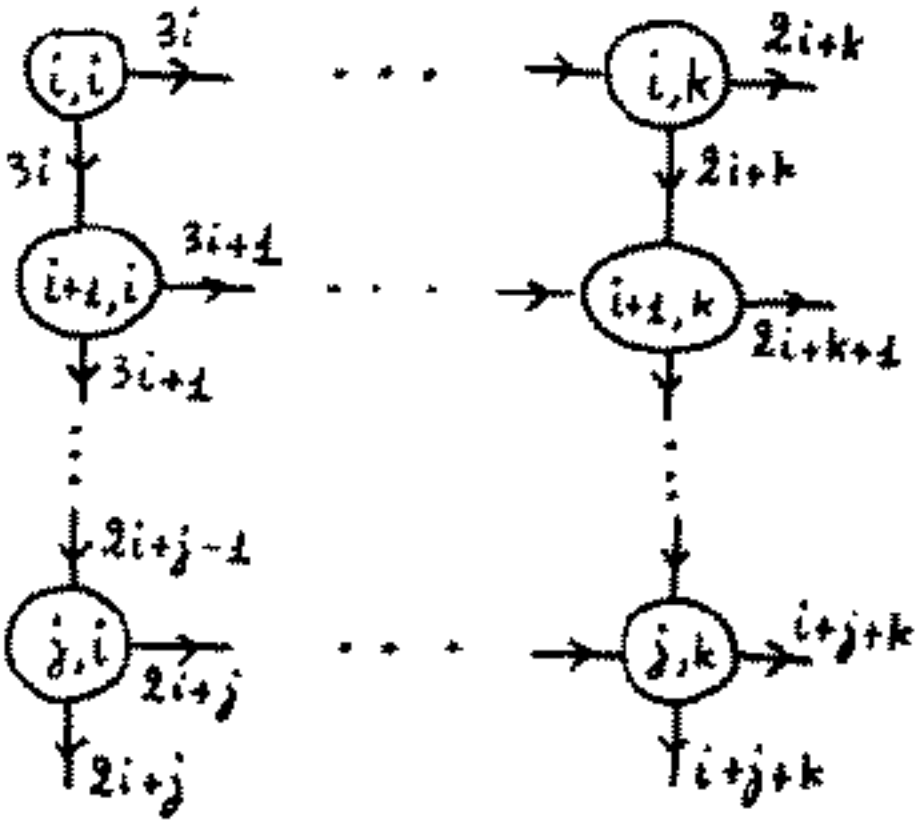

Figure For Exercise 2.2.2. The times at which the messages of the ith stage are transmitted.

Consider stages i and i', with $i \neq i'$. Processor (j,k) sends the messages corresponding to these two different stages at times $i+j+k$ and $i'+j+k$, respectively. These times are different since $i \neq i'$. Therefore, there is no conflict between the different stages, as far as link availability is concerned. Furthermore, within the ith stage, the messages transmitted to processor (j,k) are sent at time

$i+j+k-1$ and processor (j,k) is able to transmit the required messages at time $i+j+k$, as specified. (This should be clear from the figure.)

We finally need to verify that the values $C_{jk}^{(i-1)}$ computed during the $(i-1)$st stage are available at processor (j,k) at the needed time for the ith stage. In particular, we must check that $C_{jk}^{(i-1)}$ is available at processor (j,k) at time $i+j+k$. We argue by induction on i. Assuming that the first $i-1$ stages progress correctly, the messages of stage $(i-1)$ are received by processor (i,j) at time $i+j+k-1 < i+j+k$. This shows that the data needed for stage i are available at the right time, and the induction is complete.

The timing of this implementation is equal to the largest possible value of $i+j+k$, which is $3n = O(n)$.

2.2.3:

(a) Suppose that some processor in the mesh obtains the value of the maximum within $n^{1/3}$ time units. This means that the value obtained by that processor can only depend on the computations of processors within $n^{1/3}$ time distance. There are only $O(n^{2/3})$ such processors. Since the maximum of n numbers depends on all of the n numbers, $\Omega(n)$ elementary computations are needed. Since these are performed by $O(n^{2/3})$ processors, some processor must have spent $\Omega(n^{1/3})$ time units.

(b) Each one of the first $\lceil n/2 \rceil$ stages of Gaussian elimination with row pivoting involves the computation of the maximum of $\Omega(n)$ numbers. Each such computation takes time $\Omega(n^{1/3})$, according to part (a), and furthermore, these computations have to be performed consecutively, for a total of $\Omega(n^{4/3})$ time.

2.2.4:

This can be done in several ways. One method, not necessarily the most economical, does not involve any interleaving of successive elimination stages. We imbed an $n \times n$ mesh into an $O(n^2)$–node hypercube, using a reflected Gray code (see Subsection 1.3.4). At the beginning of the ith stage, the (j,k)th processor knows the value of $C_{jk}^{(i-1)}$. The algorithm proceeds as follows.

1. Processors $(i,i),\ldots,(n,i)$ perform a single node accumulation to obtain the value i^* of i for which $|C_{ji}^{(i-1)}|$, $j \geq i$ is maximized. The value of i^* together with the maximal value $C_{i^*i}^{(i-1)}$ is then broadcast back to these processors. [This takes $O(\log n)$ time].

2. Upon determination of i^*, the processors in rows i and i^* exchange their data. Since the exchange of different data involves distinct "columns" of the hypercube, these exchanges can be done simultaneously. Since the diameter of the hypercube is $O(\log n)$, the exchange also takes

$O(\log n)$ time.

Let us denote by D the matrix $C^{(i-1)}$ after it is subjected to the row interchange. Notice that $D_{ii} = C^{(i-1)}_{i^* i}$ and the value of D_{ii} has already been made available to the processors in the ith column.

3. Processor (i, k), for each $k > i$, broadcasts D_{ik} to all processors in the kth column [$O(\log n)$ time].

4. Each processor (j, i), with $j > i$, computes the ratio D_{ji}/D_{ii} and broadcasts it along the jth row [$O(\log n)$ time].

5. Each processor (j, k), with $j > i$, $k \geq i$, computes $D_{ik}D_{ji}/D_{ii}$ to obtain the value of $C^{(i)}_{jk}$.

2.2.5:

It is not hard to see that it is sufficient to verify that

$$\begin{bmatrix} c & s \\ s & -c \end{bmatrix}' \begin{bmatrix} c & s \\ s & -c \end{bmatrix} = I.$$

By construction, $c^2 + s^2 = 1$ and the result follows.

2.2.6:

Let us replace the schedule of Fig. 2.2.2 by the following one that needs only 12 parallel stages:

$$\begin{bmatrix} * & & & & & & & \\ 3 & * & & & & & & \\ 2 & 5 & * & & & & & \\ 2 & 4 & 7 & * & & & & \\ 1 & 3 & 6 & 8 & * & & & \\ 1 & 3 & 5 & 7 & 9 & * & & \\ 1 & 2 & 4 & 6 & 8 & 11 & * & \\ 1 & 2 & 3 & 5 & 7 & 10 & 12 & * \end{bmatrix}$$

As in Fig. 2.2.2, the (i, j)th entry in this diagram is the stage $T(i, j)$ at which the corresponding entry is annihilated. Recall that the (i, j)th entry is annihilated by a Givens rotation operating on rows i and $S(i, j)$. The following diagram indicates one possible choice of the rows $S(i, j)$ corresponding

to each entry (i, j).

$$\begin{bmatrix} * & & & & & & & \\ 1 & * & & & & & & \\ 1 & 2 & * & & & & & \\ 2 & 3 & 3 & * & & & & \\ 1 & 3 & 4 & 4 & * & & & \\ 2 & 4 & 5 & 5 & 5 & * & & \\ 3 & 5 & 6 & 6 & 6 & 6 & * & \\ 4 & 6 & 7 & 7 & 7 & 7 & 7 & * \end{bmatrix}$$

Notice that any two entries (i, j) and (k, ℓ) that are annihilated at the same stage satisfy $S(i, j) \neq S(k, \ell)$, as required.

SECTION 2.3

2.3.1:

We partition A by letting

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{12} & A_{22} \end{bmatrix},$$

where A_{11} has dimensions $\lceil n/2 \rceil \times \lceil n/2 \rceil$. Consider the equation

$$\begin{bmatrix} I & X \\ 0 & I \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ A'_{12} & A_{22} \end{bmatrix} \begin{bmatrix} I & 0 \\ X' & I \end{bmatrix} = \begin{bmatrix} B_1 & 0 \\ 0 & B_2 \end{bmatrix}.$$

Carrying out the matrix multiplications on the left, we obtain

$$\begin{bmatrix} A_{11} + XA'_{12} + A_{12}X' + XA_{22}X' & A_{12} + XA_{22} \\ A'_{12} + A_{22}X' & A_{22} \end{bmatrix} = \begin{bmatrix} B_1 & 0 \\ 0 & B_2 \end{bmatrix}. \tag{1}$$

We choose X so that $A_{12} + XA_{22} = 0$. Because of the symmetry of A_{22}, we also have $A'_{12} + A_{22}X' = 0$. Then, Eq. (1) is satisfied with $B_1 = XA_{22}X' + A_{12}X' + XA'_{12} + A_{11}$ and $B_2 = A_{22}$. Notice that B_1 and B_2 are also symmetric positive definite and the same procedure can be repeated on each one of them.

After $O(\log n)$ such stages we have obtained matrices $Y_1, Y_2, \ldots, Y_k$, with $k = O(\log n)$, for which

$$Y_k \cdots Y_2 Y_1 A Y'_1 Y'_2 \cdots Y'_k = D, \tag{2}$$

where D is a diagonal matrix. Let $L = Y_1' \cdots Y_k'$. Each Y_i is upper triangular, so L is lower triangular. Notice that each Y_i is invertible because its diagonal entries are equal to 1. Then, L^{-1} exists and is also lower triangular. Thus, Eq. (2) can be rewritten as $A = (L')^{-1} D L^{-1}$, which is of the desired form. Since A is assumed positive definite (and therefore nonsingular), it is also seen that D is nonsingular.

The algorithm involves a matrix inversion at each step [solving the system $XA_{22} + A_{12} = 0$, which takes $O(\log^2 n)$ time] and a few matrix multiplications. At the end of the algorithm, the matrices $Y_1, \ldots, Y_k$ must be multiplied and inverted [$O(\log^3 n)$ time]. Thus the total timing of the algorithm is $O(\log^3 n)$.

Finally, to verify that the algorithm is well–defined, we need to check that the equation $XA_{22} + A_{12} = 0$ has a solution. It is sufficient to show that the matrix A_{22} is invertible. To see that this is the case, suppose the contrary. Then, ther would exist a nonzero vector y of dimension $n - \lceil n/2 \rceil$ such that $y' A_{22} y = 0$. We could then extend y to an n–dimensional vector x by appending $\lceil n/2 \rceil$ zeroes. Then, $x' A x = 0$, contradicting the positive definiteness of A.

SECTION 2.6

2.6.1:

Without loss of generality we assume that the vector b in Eq. (6.5) is zero. If $I - M$ is singular then there exists a fixed point $x \neq 0$ of the iterative algorithm (6.5). For any one of the algorithms of Section 2.4, this implies that $Ax = 0$ which contradicts the invertibility of A.

2.6.2:

Let $\alpha \in (0, 1/3)$ and

$$A = \begin{bmatrix} 1-\alpha & -\alpha & -\alpha \\ -\alpha & 1-\alpha & -\alpha \\ -\alpha & -\alpha & 1-\alpha \end{bmatrix}.$$

Let $M = (1-\epsilon)A$, where ϵ is a positive scalar such that $(1+\alpha)(1-\epsilon) > 1$. Notice that $|M|e = (1-\epsilon)(1+\alpha)e$, where e is the vector $(1,1,1)$. Therefore, $\rho(|M|) \geq (1-\epsilon)(1+\alpha) > 1$. This shows that $\|M\|_\infty^w > 1$ for any positive vector w.

We now show that $\rho(M) < 1$. We represent A in the form $A = I - N$, where N is a matrix with all entries equal to α. The eigenvalues of N are easily seen to be 0, 0, and 3α. Thus, the eigenvalues of A are 1, 1, and $1 - 3\alpha$. It follows that the eigenvalues of M are $1 - \epsilon$, $1 - \epsilon$ and $(1 - \epsilon)(1 - 3\alpha)$, all of them smaller than 1 in magnitude. This shows that $\rho(M) < 1$.

2.6.3:

(a) Let e be the vector with all entries equal to one. Since M is irreducible, each one of its rows has a nonzero entry. Thus $Me > 0$. We have $r(e) = \sup\{\rho \mid [Me]_i \geq \rho,\ \forall i\} = \min_i [Me]_i > 0$, and $\lambda \geq r(e) > 0$.

(b) For any positive scalar c, we have $\{\rho \mid Mx \geq \rho x\} = \{\rho \mid cMx \geq c\rho x\}$, which implies that $r(x) = r(cx)$. It follows that

$$\sup\{r(x) \mid x \in X\} = \sup\left\{ r\left(\frac{x}{\sum_{i=1}^n x_i}\right) \;\middle|\; x \in X \right\} = \sup\{r(x) \mid x \in S\}.$$

(c) Since M is irreducible, we have $(I + M)^{n-1} > 0$ (Prop. 6.3). If $x \in S$ then $x \geq 0$ and $x \neq 0$, from which it follows that $(I + M)^{n-1}x > 0$.

(d) By definition, $\sup\{r(x) \mid x \in Q\} \leq \sup\{r(x) \mid x \in X\} = \lambda$. For the reverse inequality, let $x \in S$. The definition of $r(x)$ yields $Mx \geq r(x)x$. We multiply both sides by $(I + M)^{n-1}$ to obtain $M(I + M)^{n-1}x \geq r(x)(I + M)^{n-1}x$. The definition of $r\big((I + M)^{n-1}x\big)$ implies that $r\big((I + M)^{n-1}x\big) \geq r(x)$. Taking the supremum over all $x \in S$, we obtain

$$\sup\{r(x) \mid x \in Q\} = \sup\big\{r\big((I + M)^{n-1}x\big) \mid x \in S\big\} \geq \sup\{r(x) \mid x \in S\} = \lambda,$$

where the last step uses the result of part (b).

(e) We have $r(x) = \sup\{\rho \mid \rho \leq [Mx]_i/x_i, \forall i \text{ such that } x_i \neq 0\}$. Thus, $r(x) = \min_i\big\{[Mx]_i/x_i \mid x_i \neq 0\big\}$. For $x \in Q$ and for all i, we have $x_i > 0$, and it follows that on the set Q, $r(x)$ is given by $r(x) = \min_i [Mx]_i/x_i$, which is a continuous function.

(f) The function $r\big((I + M)^{n-1}x\big)$, is continuous on the set S. This is because, for $x \in S$, we have $(I + M)^{n-1}x \in Q$ and $r(x)$ is continuous on Q. The set S is closed and bounded and (by Weierstrass' theorem) there exists some $y \in S$ such that

$$r\big((I + M)^{n-1}y\big) = \sup_{x \in S} r\big((I + M)^{n-1}x\big).$$

Let $w = (I + M)^{n-1}y$. Then,

$$r(w) = \sup_{x \in S} r\big((I + M)^{n-1}x\big) = \sup_{x \in Q} r(x) = \lambda.$$

(g) Let $z = Mw - \lambda w$. Since $r(w) = \lambda$, we have $Mw \geq \lambda w$ and $z \geq 0$. If $z \neq 0$, then $(I+M)^{n-1}z > 0$ which shows that $M(I+M)^{n-1}w > \lambda(I+M)^{n-1}w$. This implies that $r\big((I+M)^{n-1}w\big) > \lambda$, which contradicts the definition of λ.

2.6.4:

(a) See Prop. 2.2 in Section 3.2.

(b) Assume, without loss of generality, that $b = 0$. In particular, $x^* = 0$. Consider an update of the ith coordinate of x. The update formula for the SOR algorithm [cf. Eq. (4.8)] can be written as

$$x_i := x_i - \frac{\gamma}{a_{ii}}(a_i'x),$$

where a_i' is the ith row of x. Then, the value of $F(x) = \frac{1}{2}x'Ax$ after the update is given by

$$\frac{1}{2}x'Ax - x'a_i\frac{\gamma}{a_{ii}}(a_i'x) + \frac{1}{2}\frac{\gamma^2}{a_{ii}^2}a_{ii}(a_i'x)^2 = \frac{1}{2}x'Ax - \frac{(a_i'x)^2}{a_{ii}}\big(\gamma - \frac{1}{2}\gamma^2\big).$$

If $\gamma < 0$ or if $\gamma > 2$, we see that the value of F does not decrease. Thus, $F\big(x(t)\big) \geq F\big(x(0)\big)$, for all t. If we start with some $x(0) \neq 0$, then $F\big(x(t)\big) \geq F\big(x(0)\big) > 0$ and $x(t)$ does not converge to zero.

2.6.5:

See Prop. 2.1 in Section 3.2.

SECTION 2.7

2.7.1:

Let

$$P(\lambda) = \frac{2}{(a+b)\lambda_1 \cdots \lambda_k}\Big(\frac{a+b}{2} - \lambda\Big)(\lambda_1 - \lambda)\cdots(\lambda_k - \lambda)$$

Note that P is a polynomial of degree $k+1$ and its zeroth order term is equal to 1 or -1. This polynomial vanishes at the eigenvalues $\lambda_1, \ldots, \lambda_k$ of A. Thus, using Eq. (7.11),

$$F\big(x(k+1)\big) \leq \max_{k+1 \leq i \leq n} \big(P(\lambda_i)\big)^2 F\big(x(0)\big). \tag{1}$$

For $k+1 \leq i \leq n$, we have $\lambda_i \in [a,b]$. Thus, $\left|(a+b)/2 - \lambda_i\right| \leq (b-a)/2$. Furthermore, for every $\lambda \in [a,b]$, we have

$$\left|\frac{(\lambda_1 - \lambda)\cdots(\lambda_k - \lambda)}{\lambda_1 \cdots \lambda_k}\right| \leq 1,$$

because $\lambda_1, \ldots, \lambda_k > b \geq \lambda$. Thus, for $k+1 \leq i \leq n$, we have $|P(\lambda_i)| \leq (b-a)/(a+b)$ which, in conjunction with Eq. (1) yields the desired result.

2.7.2:

According to the discussion in Subsection 2.7.3, the bounds of Eqs. (7.11) and (7.12) are applicable, provided that we consider the eigenvalues of $H^{1/2}AH^{1/2}$, where H is the preconditioning matrix. In our case,

$$H^{1/2}AH^{1/2} = I + M^{-1/2}\Big(\sum_{i=1}^{k} v_i v_i'\Big)M^{-1/2}$$

The rank of the matrix $\sum_{i=1}^{k} v_i v_i'$ is at most k, and therefore $n-k$ of its eigenvalues are zero. The remaining k of its eigenvalues are nonnegative. Thus, $n-k$ of the eigenvalues of $H^{1/2}AH^{1/2}$ are equal to 1, and the remaining are no smaller than one. Thus, its eigenvalues take at most $k+1$ distinct values and, according to the discussion in the end of Subsection 2.7.2, the conjugate gradient method terminates after at most $k+1$ steps.

2.7.3:

The computation per processor at any given iteration is $\Theta(N/p)$. The communication needed for the inner product evaluation is proportional to the diameter of the network, that is $\Theta(p^{1/2})$. We thus wish to minimize $\Theta(N/p) + \Theta(p^{1/2})$ with respect to p, which yields $p = N^{2/3}$.

2.7.4:

Suppose that the algorithm has not terminated after k stages, that is, $x(k) \neq 0$. Since A is nonsingular, we obtain $g(k) \neq 0$. We use Eq. (7.4) to obtain

$$s(k)'g(k) = -g(k)'g(k) + \sum_{i=0}^{k-1} c_i s(i)'g(k) = -g(k)'g(k) < 0,$$

where the second equality follows from Prop. 7.1(b). This implies that

$$\frac{\partial}{\partial\gamma}F\big(x(k)+\gamma s(k)\big)\big|_{\gamma=0} = s(k)'g(k) < 0$$

and shows that when γ is positive and very small, we have $F\big(x(k)+\gamma s(k)\big) < F\big(x(k)\big)$. Since $x(k+1)$ minimizes $F\big(x(k)+\gamma s(k)\big)$ over all $\gamma > 0$, we conclude that $F\big(x(k+1)\big) < F\big(x(k)\big)$.

SECTION 2.8

2.8.1:

(a) We first express the algorithm in a more convenient form. Let $d_t = \|Ax(t)\|$. Then, $x(t+1) = Ax(t)/d_t$ which shows that $x(t) = A^t x(0)/(d_0 \cdots d_{t-1})$. For any $t > 0$, we have $\|x(t)\| = 1$ which implies that $d_0 \cdots d_{t-1} = \|A^t x(0)\|$. We conclude that

$$x(t) = \frac{A^t x(0)}{\|A^t x(0)\|}. \tag{1}$$

The eigenvectors $x^1, \ldots, x^n$ are linearly independent, they span $\Re^n$, and there exist scalars $c_1, \ldots, c_n$ such that $x(0) = \sum_{i=1}^n c_i x^i$. Furthermore, since $x(0)$ does not belong to the span of $x^2, \ldots, x^n$, we must have $c_1 \neq 0$. Notice that $A^t x(0) = \sum_{i=1}^n c_i \lambda_i^t x^i$. Equivalently, $A^t x(0)/\lambda_1^t = \sum_{i=1}^n c_i(\lambda_i^t/\lambda_1^t)x^i$ and since $|\lambda_i| < |\lambda_1|$ (for $i \neq 1$), we obtain $\lim_{t\to\infty} A^t x(0)/\lambda_1^t = c_1 x^1$. We then see that $\lim_{t\to\infty} \|A^t x(0)\|/\lambda_1^t = \|c_1 x^1\| \neq 0$. We finally use Eq. (1) to obtain

$$\lim_{t\to\infty} x(t) = \frac{c_1 x^1}{\|c_1 x^1\|}.$$

This vector is a scalar multiple of the eigenvector x^1 and therefore satisfies $Ax = \lambda_1 x$.

(b) We use the norm defined by $\|\pi\| = \sum_{i=1}^n |\pi_i|$. Then, iteration (8.3) can be written as

$$\pi(t+1)' = P'\pi(t)' = \frac{P'\pi(t)'}{\|P'\pi(t)'\|}.$$

The last equality follows because if $\pi(0) \geq 0$ and $\sum_{i=1}^n \pi_i(0) = 1$, then $\|\pi(t)\| = \sum_{i=1}^n \pi_i(t) = 1$ for all $t \geq 0$.

2.8.2:

(a) Since P is irreducible, $\alpha P/(1-\alpha)$ is also irreducible. Thus $\big(I + \frac{\alpha}{1-\alpha}P\big)^{n-1} > 0$. Equivalently, $Q^{n-1} > 0$ which shows that Q is primitive.

(b) We notice that a vector π satisfies $\pi P = \pi$ if and only if $\pi Q = \pi$. By Prop. 8.3, there exists a unique positive vector (up to multiplication by a positive scalar) such that $\pi^* Q = \pi^*$. It follows that there is a unique vector (up to multiplication by a scalar) such that $\pi^* P = \pi^*$. Such a vector π^* can be computed by fixing some $\alpha \in (0,1)$ and using the iteration $\pi := \alpha\pi + \alpha\pi/(1-\alpha)\pi P = \alpha Q$.

2.8.3:

Since C is irreducible, there exists a unique positive row vector $\tilde{\pi} \in \Re^{n-n_1}$ whose entries sum to one and such that $\tilde{\pi}C = \tilde{\pi}$. Consider the vector $\pi^* = [0,\ \tilde{\pi}] \in \Re^n$. Then $\pi^* P = \pi^*$ which establishes an existence result. We now prove uniqueness. Consider a row vector $\pi = [\hat{\pi},\ \bar{\pi}] \geq 0$ in $\Re^n$ with $\hat{\pi} \in \Re^{n_1}$, $\bar{\pi} \in \Re^{n-n_1}$, such that $\pi P = \pi$. Then $\hat{\pi}A = \hat{\pi}$. We proceed as in the proof of Prop. 8.4, to see that there exists some $T \geq 1$ such that $\sum_{j=1}^{n_1} [A^T]_{ij} < 1$, for each $i \leq n_1$. Thus, $\rho(A^T) \leq \|A^T\|_\infty < 1$ and we conclude that $\rho(A) < 1$. This implies that 1 is not an eigenvalue of A and, therefore, $\hat{\pi} = 0$. Thus, $\bar{\pi}C = \bar{\pi}$ and if the entries of $\bar{\pi}$ are normalized to sum to 1, we must have $\bar{\pi} = \tilde{\pi}$ which proves the uniqueness of π^*.

2.8.4

Since P is irreducible, for every j and k, there exists some t_{jk} such that $[P^{t_{jk}}]_{jk} > 0$,

Let $T = 2\max_{j,k} t_{jk}$. For any ℓ and m, we have

$$[P^T]_{\ell m} \geq [P^{t_{\ell i}}]_{\ell i} [P^{T - t_{\ell i} - t_{im}}]_{ii} [P^{t_{im}}]_{im} > 0,$$

which proves that P is primitive.

SECTION 2.9

2.9.1:

(a) If D is a matrix of sufficiently small norm, then

$$\begin{aligned} f(X+D) &= A - (X+D)^{-1} \\ &= A - \big(X(I + X^{-1}D)\big)^{-1} \\ &= A - (I + X^{-1}D)^{-1}X^{-1} \\ &= A - \big(I - X^{-1}D + X^{-1}DX^{-1}D - \cdots\big)X^{-1} \\ &= A - X^{-1} + X^{-1}DX^{-1} + h(X,D). \end{aligned}$$

(b) We want $A - X^{-1} + X^{-1}DX^{-1} = 0$. Equivalently, $XAX - X + D = 0$, or $D = X - XAX$.

(c) We have $X := X + D = 2X - XAX$.

2.9.2:

Let $\lambda_1 < \cdots < \lambda_n$ be the eigenvalues of $A'A$. The inequalities $\|A\|_2^2 \leq \|A\|_\infty \cdot \|A\|_1 \leq n\|A\|_2^2$ [Props. A.25(e) and A.13(f) in Appendix A] yield $\lambda_n \leq \|A\|_\infty \cdot \|A\|_1 \leq n\lambda_n$. Thus, the eigenvalues of $I - A'A/(\|A\|_\infty \cdot \|A\|_1)$ are bounded below by $1 - (\lambda_n/\lambda_n) = 0$ and above by $1 - \lambda_1/(n\lambda_n) = 1 - 1/\big(n\kappa^2(A)\big)$.

2.9.3:

Since A is symmetric, we have $\|A\|_\infty = \|A\|_1$. Thus, $\lambda_n \leq \|A\|_\infty^2 \leq n\lambda_n$. Proceeding as in Exercise 9.2, the eigenvalues of $I - A/||A||_\infty$ are bounded below by zero and above by $1 - \big(\lambda_1/(n\lambda_n)\big)^{1/2} = 1 - 1/\big(n^{1/2}\kappa(A)\big)$.

2.9.4:

We have $[I - B_0A]_{ij} = -a_{ij}/a_{ii}$ for $j \neq i$, and $[I - B_0A]_{ii} = 0$. Thus, $\|I - B_0A\|_\infty = \sum_{j\neq i} |a_{ij}/a_{ii}| \leq 1 - 1/n^c$.

CHAPTER 3

SECTION 3.1

3.1.1:

Let $X = \Re - \{0\}$ and $T(x) = x/2$.

3.1.2:

(a) Let $X = \{(x_1, x_2) \mid x_1 \geq 0, x_2 \geq 0\}$ and

$$T(x_1, x_2) = \begin{bmatrix} \min\{x_1, x_2/2\} \\ x_2/2 \end{bmatrix}.$$

Here $x^* = (0, 0)$ is the unique fixed point of T. Also,

$$\|T(x_1, x_2)\|_\infty \leq \frac{x_2}{2} \leq \frac{1}{2}\|x\|_\infty.$$

Furthermore, T is continuous and, therefore, it satisfies the assumptions of Prop. 1.8. Now, by definition, $R_1(x) = \{y_1 \geq 0 \mid y_1 = \min\{y_1, x_2/2\}\}$ and therefore $R_1(x) = \{y_1 \mid 0 \leq y \leq x_2/2\}$, which contains infinitely many elements.

(b) Let $X = \{0, 1, 2\} \times \{0, 3\}$, which is closed, nonempty but not convex. Let $T_2(x) = 3$, for all x and $T_1(x_1, 3) = 2$, $T_1(0, 0) = 1$, $T_1(1, 0) = 0$, $T_1(2, 0) = 1$. Here, $x^* = (2, 3)$ is the unique fixed point of T. We have $T(x) - x^* = 0$, if $x_2 = 3$. If $x_2 = 0$, then $\|x - x^*\|_\infty = 3$ and

$$\|T(0, 0) - x^*\|_\infty = \left\| \begin{bmatrix} 1 \\ 3 \end{bmatrix} - \begin{bmatrix} 2 \\ 3 \end{bmatrix} \right\|_\infty = 1 < \frac{3}{4} \left\| \begin{bmatrix} 0 \\ 0 \end{bmatrix} - x^* \right\|_\infty,$$

$$\|T(1, 0) - x^*\|_\infty = \left\| \begin{bmatrix} 0 \\ 3 \end{bmatrix} - \begin{bmatrix} 2 \\ 3 \end{bmatrix} \right\|_\infty = 2 < \frac{3}{4} \left\| \begin{bmatrix} 1 \\ 0 \end{bmatrix} - x^* \right\|_\infty,$$

$$\|T(2,0)-x^*\|_\infty = \left\| \begin{bmatrix} 1 \\ 3 \end{bmatrix} - \begin{bmatrix} 2 \\ 3 \end{bmatrix} \right\|_\infty = 1 < \frac{3}{4} \left\| \begin{bmatrix} 2 \\ 0 \end{bmatrix} - x^* \right\|_\infty .$$

Thus, T has the property $\|T(x)-x^*\|_\infty < \frac{3}{4}\|x-x^*\|$ for all x. Furthermore, T is continuous. Nevertheless, if $x_2 = 0$, there is no solution to the equation $x_1 = T_1(x_1, 0)$ and the set $R_1(x_1, 0)$ is empty for every $x_1 \in \{0, 1, 2\}$.

(c) Let $X = \Re^2$. For $x \in \{0,1,2\} \times \{0,3\}$, let T be the same as in part (b). For any other x, let $T(x) = x^* = (2,3)$. Clearly, T is a pseudocontraction but it is not continuous. If $x_2 = 0$, then the following hold. If $x_1 \notin \{0,1,2\}$, then $T_1(x_1, 0) = 2 \neq x_1$. Also, $T_1(0,0) = 1$, $T(1,0) = 0$, $T_1(2,0) = 1$, and there is no x_1 satisfying $x_1 = T_1(x_1, 0)$, which shows that the set $R_1(x_1, 0)$ is empty for every x_1.

3.1.3:

We will apply Prop. 1.10, with $G_i = 1$ for each i, and with $\|\cdot\|$ being the weighted maximum norm $\|\cdot\|_\infty^w$. We thus have $\|x_i\|_i = |x_i|/w_i$. Notice that, for any $a \in \Re$,

$$\|a\|_{ij} = \max_{x \neq 0} \frac{\|ax\|_i}{\|x\|j} = \frac{|ax|/w_i}{|x|/w_j} = |a|\frac{w_j}{w_i}.$$

Let γ satisfy $0 < \gamma < 1/K$. Then,

$$\big|1-\gamma\big(\nabla_i f_i(x)\big)\big| + \gamma \sum_{j \neq i} |\nabla_j f_i(x)| \cdot \frac{w_j}{w_i} = 1 - \gamma\Big(\nabla_i f_i(x) - \sum_{j \neq i} |\nabla_j f_i(x)| \frac{w_j}{w_i}\Big) \leq 1 - \frac{\gamma\beta}{w_i}.$$

Therefore, condition (1.14) of Prop. 1.10 is satisfied with $\alpha = \max_i\big(1 - \gamma\beta/w_i\big) < 1$.

3.1.4:

(a) Let $T^k : \Re^n \mapsto \Re^n$ be defined by $T^1(x) = T(x)$ and $T^k(x) = T\big(T^{k-1}(x)\big)$, $k > 1$. Since $T(y^*) \geq y^*$, an easy inductive argument shows that $T^k(y^*) \geq T^{k-1}(y^*)$ for all $k > 1$. In particular, the sequence $\{T^k(y^*)\}$ is nondecreasing. Similarly, the sequence $\{T^k(z^*)\}$ is nonincreasing. Using the monotonicity of T we have

$$y^* \leq T^k(y^*) \leq T^k(z^*) \leq z^*, \quad \forall k.$$

This shows that the sequence $\{T^k(y^*)\}$ is bounded above and, therefore, it converges to some $\hat{x} \in H$. Since T is continuous,

$$T(\hat{x}) = T\Big(\lim_{k\to\infty} T^k(y^*)\Big) = \lim_{k\to\infty} T^{k+1}(y^*) = \hat{x},$$

and, since x^* is the unique fixed point of T, we conclude that $\hat{x} = x^*$. In particular, $x^* \in H$. The proof that $T^k(z^*)$ converges to x^* is identical.

(b) We have $y^* \leq x(0) \leq z^*$ and, using the monotonicity of T, we obtain $T^k(y^*) \leq T^k\big(x(0)\big) \leq T(z^*)$, for all k. Thus, the sequence $\big\{T^k\big(x(0)\big)\big\}$ lies between two sequences converging to x^* and must converge to x^* as well.

(c) Let $\hat{T}_1 : \Re^n \to \Re^n$ be defined as in Eq. (1.22). The mapping $S : \Re^n \to \Re^n$ corresponding to one iteration of the Gauss-Seidel algorithm based on T is equal to the composition of $\hat{T}_1, \hat{T}_2, \ldots, \hat{T}_n$. Since T is monotone, each $\hat{T}_1$ is also monotone and the same conclusion obtains for S. Furthermore, each $\hat{T}_1$ maps H into H and the same must be true for S. In particular, $S(y^*) \geq y^*$ and a similar argument yields $S(z^*) \leq z^*$. The mapping S is clearly continuous and has x^* as its unique fixed point. Convergence of the Gauss-Seidel algorithm follows by applying the result of part (b) to the mapping S.

(d) Since the mapping $\hat{T}_i$ is monotone, the sequence $\{x_i(t)\}$ is either nonincreasing or nondecreasing, depending on whether $x_i(1) \leq x_i(0)$ or $x_i(1) \geq x_i(0)$, respectively. Furthermore, $y^* \leq \hat{T}_i(x) \leq z^*$, for every x in H, and this shows that $x_i(t)$ is bounded between y_i^* and z_i^*. Thus, the sequence $\{x_i(t)\}$ is monotone and bounded and must converge.

(e) We define $(\hat{T}_i)^k$ as the composition of k copies of $\hat{T}_i$. If $y \leq z$, then $(\hat{T}_i)^k(y) \leq (\hat{T}_i)^k(z)$ for all k, because $\hat{T}_i$ is monotone and, taking the limit, we obtain $Q_i(y) \leq Q_i(z)$. Thus Q is monotone. For an example where Q is discontinuous, let

$$y^* = (0,0), \qquad z^* = (1,1)$$

and

$$T_1(x_1, x_2) = \frac{x_1}{2}(1 + x_2), \qquad T_2(x_1, x_2) = 0.$$

The mapping T is clearly monotone. It is also continuous and has a unique fixed point $x^* = (0,0)$. Notice that $\hat{T}_1(x_1, 1) = x_1$ for every $x_1 \in [0,1]$, and this shows that $Q_1(x_1, 1) = x_1$ for every $x_1 \in [0,1]$. On the other hand, for every $x_2 \in [0,1)$ we have $(\hat{T}_i)^k(x_1, x_2) = x_1\big((1 + x_2)/2\big)^k$, which converges to zero. Thus, $Q_1(x_1, x_2) = 0$, if $x_2 \in [0,1)$, and the mapping Q_1 is discontinuous at $(x_1, 1)$, for every $x_1 \neq 0$.

(f) It can be seen that $Q(y^*) \geq y^*$ and $Q(z^*) \leq z^*$. However, the result does not follow from parts (b) and (c) of this exercise because Q is not necessarily continuous. We shall show that Q has the following property: if $x \leq T(x)$ then $T(x) \leq Q(x)$. Indeed, if $x \leq T(x)$ then $x_i \leq T_i(x) = [\hat{T}_i(x)]_i$ and by the monotonicity of $\hat{T}_i$, we have $x_i \leq [(\hat{T}_i)^k(x)]_i$ for all k. Taking the limit, as $k \to \infty$, we obtain $x_i \leq Q_i(x)$. We now use induction to show $T^k(y^*) \leq Q^k(y^*)$. For $k = 1$ we have $y^* \leq T(y^*)$ which implies that $T(y^*) \leq Q(y^*)$. Assume that $T^{k-1}(y^*) \leq Q^{k-1}(y^*)$. Since $T^{k-1}(y^*) \leq T\big(T^{k-1}(y^*)\big)$

we obtain $T^k(y^*) = T\big(T^{k-1}(y^*)\big) \leq Q\big(T^{k-1}(y^*)\big) \leq Q\big(Q^{k-1}(y^*)\big) = Q^k(y^*)$, which completes the induction. An identical argument proves that $Q^k(z^*) \leq T^k(z^*)$, for all k. Thus the sequence $\{Q^k(z^*)\}$ lies between two sequences converging to x^* and must also converge to x^*. The same result obtains for any $x(0) \in H$ because the monotonicity of Q implies that $Q^k(y^*) \leq Q^k\big(x(0)\big) \leq Q^k(z^*)$.

Let $P : \Re^n \mapsto \Re^n$ be the mapping corresponding to the Gauss-Seidel algorithm based on Q. By repeating the argument in part (c) of this exercise, we can show that P is monotone and that if $x \leq Q(x)$ then $Q(x) \leq P(x)$. We then repeat the argument in the preceding paragraph to see that $Q^k(y^*) \leq P^k(y^*) \leq P^k\big(x(0)\big) \leq P^k(z^*) \leq Q^k(z^*)$, from which convergence of $P^k\big(x(0)\big)$ to x^* follows.

SECTION 3.2

3.2.1:

(a) Given a bounded set A, let $r = \sup\{\|x\|_2 \mid x \in A\}$ and $B = \{x \mid \|x\|_2 \leq r\}$. Let $K = \max\{\|\nabla^2 F(x)\|_2 \mid x \in B\}$, which is finite because a continuous function on a compact set is bounded. For any $x, y \in A$ we have

$$\nabla F(x) - \nabla F(y) = \int_0^1 \Big(\nabla^2 F(tx + (1-t)y)\Big)(x-y)dt.$$

Notice that $tx + (1-t)y \in B$, for all $t \in [0,1]$. It follows that

$$\|\nabla F(x) - F(y)\|_2 \leq K\|x-y\|_2,$$

as desired.

(b) The key idea is to show that $x(t)$ stays in a bounded set and to use a step size γ determined by the constant K corresponding to this bounded set. Given the initial vector $x(0)$, let $A = \big\{x \mid F(x) \leq F\big(x(0)\big)\big\}$ and $R = \max\{\|x\|_2 \mid x \in A\}$. Let $a = \max\{\|\nabla F(x)\|_2 \mid x \in A\}$ and $B = \{x \mid \|x\|_2 \leq R+2a\}$. Using condition (i), there exists some constant K such that $\|\nabla F(x) - \nabla F(y)\|_2 \leq K\|x-y\|_2$, for all $x, y \in B$. Let us choose a step size γ satisfying $0 < \gamma < 1$ and $\gamma < 2K_2 \min\{1, 1/K\}$. Let $\beta = \gamma(K_2 - K\gamma/2)$ which is positive by our choice of γ. We will, show by induction on t, that, with such a choice of step size, we have $x(t) \in A$ and

$$F\big(x(t+1)\big) \leq F\big(x(t)\big) - \beta\|s(t)\|_2^2, \tag{1}$$

for all $t \geq 0$.

To start the induction, we notice that $x(0) \in A$, by the definition of A. Suppose that $x(t) \in A$. Inequality (2.12) in the text yields

$$K_2\|s(t)\|_2^2 \le \big|s(t)'\nabla F\big(x(t)\big)\big| \le \|s(t)\|_2 \cdot \big\|\nabla F\big(x(t)\big)\big\|_2.$$

Thus, $\|s(t)\|_2 \le \big\|\nabla F\big(x(t)\big)\big\|_2/K_2 \le a/K_2$. Hence, $\|x(t)+\gamma s(t)\|_2 \le \|x(t)\|_2 + \gamma a/K_2 \le R+2a$, which shows that $x(t)+\gamma s(t) \in B$. In order to prove Eq. (1), we now proceed as in the proof of Prop. 2.1. A difficulty arises because Prop. A.32 is used there, which assumes that the inequality $\|\nabla F(x)-\nabla F(y)\|_2 \le K\|x-y\|_2$ holds for all x, y, whereas in this exercise this inequality holds only for $x, y \in B$. We thus essentially repeat the proof of Prop. A.32, to obtain

$$\begin{aligned} F\big(x(t+1)\big) &= F\big(x(t)+\gamma s(t)\big) \\ &= \int_0^1 \gamma s(t)'\nabla F\big(x(t)+\tau\gamma s(t)\big)\,d\tau \\ &\le \gamma s(t)'\nabla F\big(x(t)\big) + \left|\int_0^1 \gamma s(t)'\Big(\nabla F\big(x(t)+\gamma\tau s(t)\big) - \nabla F\big(x(t)\big)\Big)\,d\tau\right| \\ &\le \gamma s(t)'\nabla F\big(x(t)\big) + \gamma^2\|s(t)\|_2^2\int_0^1 K\tau\,d\tau \\ &= \gamma s(t)'\nabla F\big(x(t)\big) + \frac{K\gamma^2}{2}\|s(t)\|_2^2. \end{aligned} \tag{2}$$

We have used here the inequality

$$\big\|\nabla F\big(x(t)+\gamma\tau s(t)\big) - \nabla F\big(x(t)\big)\big\|_2 \le \gamma K\tau\|s(t)\|_2,$$

which holds because of our definition of K and because $x(t) \in A \subset B$, $x(t)+\gamma s(t) \in B$ and (because of the convexity of B) $x(t)+\gamma\tau s(t) \in B$, for $\tau \in [0,1]$.

Inequality (1) now follows from Eq. (2) as in the proof of Prop. 2.1. In particular $F\big(x(t+1)\big) \le F\big(x(t)\big) \le F\big(x(0)\big)$ and $x(t+1) \in A$. This completes the induction. The remainder of the proof is the same as in Prop. 2.1.

3.2.2:

Let $F : \Re^2 \to \Re$ be the function defined by

$$F(x) = \max\Big\{(x_1-1)^2+(x_2+1)^2, (x_1+1)^2+(x_2-1)^2\Big\}.$$

Such an F is the maximum of two strictly convex functions and is therefore itself strictly convex. The function F is minimized at $(x_1, x_2) = (0,0)$. To see this, notice that $F(x_1, x_2) = F(-x_1, -x_2)$

and, $F(0,0) \leq \big(F(x_1,x_2) + F(-x_1,-x_2)\big)/2 = F(x_1,x_2)$. On the other hand, the point $x^* = (1,1)$ is a fixed point of the nonlinear Jacobi algorithm. To see this, notice that

$$F(1,1) = 4 \leq \max\{(x_1-1)^2+4, (x_1+1)^2\} = F(x_1,1), \qquad \forall x_1,$$

$$F(1,1) = 4 \leq \max\{(1+x_2)^2, 4+(x_2-1)^2\} = F(1,x_2). \qquad \forall x_2.$$

In particular, the nonlinear Jacobi or the nonlinear Gauss-Seidel algorithm, initialized at $(1,1)$ do not converge to the minimizing point $(0,0)$.

3.2.3:

We have $\nabla F(x) - \nabla F(y) = (Ax-b) - (Ay-b) = A(x-y)$. Thus, $\|\nabla F(x) - \nabla F(y)\|_2 \leq \|A\|_2 \cdot \|x-y\|_2$. Since A is symmetric positive definite, $\|A\|_2$ is equal to the largest eigenvalue of A (Prop. A.24 in Appendix A), which yields the desired result.

As far as the convergence of the scaled gradient iteration is concerned, we notice that the iteration can be written in the form $x(t+1) = (I - \gamma M^{-1}A)x(t) + \gamma M^{-1}b$. If the method converges, then it converges to some x^* satisfying $x^* = (I-\gamma M^{-1}A)x^* + \gamma M^{-1}b$. Equivalently, $M^{-1}Ax^* = M^{-1}b$, or $x^* = A^{-1}b$. To show that the method converges, it is sufficient to establish that $\rho\big(I - \gamma M^{-1}A\big) < 1$, which we do next.

Let C, D be two square matrices of the same dimensions. If λ is an eigenvalue of CD, then there exists a nonzero vector x such that $CDx = \lambda x$. This implies that $DC(Dx) = \lambda(Dx)$. Thus, Dx is an eigenvector of DC with eigenvalue λ. We conclude that CD and DC have the same eigenvalues, and $\rho(DC) = \rho(CD)$. We apply this result to $M^{-1}A$ to obtain $\rho\big(M^{-1}A\big) = \rho\big(M^{-1/2}AM^{-1/2}\big)$. This shows that $\rho\big(I-\gamma M^{-1}A\big) = \rho\big(I-\gamma M^{-1/2}AM^{-1/2}\big)$. Notice that $M^{-1/2}AM^{1/2}$ is symmetric positive definite and therefore its eigenvalues lie in the interval $(0,\bar{K}]$, where $\bar{K}$ is the largest eigenvalue. Thus the eigenvalues of $I - \gamma M^{-1/2}AM^{-1/2}$ lie between $1-\gamma\bar{K}$ and 1, the value 1 itself being excluded. If $\gamma \in (0, 2/\bar{K})$, then $|1-\gamma\bar{K}| < 1$, which shows that the eigenvalues of $I - \gamma M^{-1/2}AM^{-1/2}$ lie in the interval $(-1,1)$ and proves the desired result.

SECTION 3.3

3.3.1:

(a) If $y \in X$ and $(y-x)'z = 0$ for every $z \in X$, then $(y-x)'y = 0$. Therefore, $(y-x)'(y-z) = 0$ for all $z \in X$, and Prop. 3.2(b) shows that $y = [x]^+ = f(x)$. Conversely, if $y = [x]^+$ then $(w-y)'(x-y) \leq 0$ for all $w \in X$. Given any $z \in X$, let $w = y + z$, which belongs to X because $y \in X$ and X is a subspace. Then, $z'(x-y) \leq 0$. Similarly, by letting $w = y - z$, we obtain $-z'(x-y) \leq 0$. These two inequalities together imply that $z'(y-x) = 0$.

(b) In view of part (a), it is sufficient to show that

$$(af(x) + bf(y) - ax - by)'z = 0, \qquad \forall z \in X. \tag{1}$$

Using part (a), we have $(f(x)-x)'z = 0$ and $(f(y)-y)'z = 0$, for all $z \in X$, and by combining these two equalities we obtain Eq. (1).

(c)

(i) Let $x \in X$. Since $(x-x)'z = 0$ for all $z \in X$, part (a) shows that $Px = x$.

(ii) For any $x \in \Re^n$, we have $Px \in X$ and, by part (i), $P(Px) = Px$. Since this is true for all $x \in \Re^n$, the equality $P^2 = P$ follows.

(iii) Using part (a) and the property $Px \in X$, we see that the vectors Px and $x-Px$ are orthogonal. The result then follows from the Pythagorean theorem.

(iv) Let $x, y \in \Re^n$. We have

$$y'Px = (Py)'Px + (y-Py)'Px = (Py)'Px.$$

[The second equality follows from part (a) and the fact $Px \in X$.] Similarly,

$$y'P'x = x'Py = (Px)'(Py) = (Py)'(Px).$$

We conclude that $y'Px = y'P'x$, for all $x, y \in \Re^n$. Let $y = (Px-P'x)$ to obtain $\|Px-P'x\|_2^2 = 0$, for all $x \in \Re^n$. Therefore, $Px = P'x$ for all $x \in \Re^n$, which implies that $P = P'$.

3.3.2:

(a) It is sufficient to show that the function $(x-y)'M(t)(x-y)$ is a strictly convex function of y, when y is restricted to X. The result will then follow by the same argument that was used in the proof of Prop. 3.2(a). We have $(x-y)'M(t)(x-y) = x'M(t)x + y'M(t)y - 2x'M(t)y$. The term $x'M(t)x$ is independent of y, the term $2x'M(t)y$ is linear in y (hence convex), and it is sufficient to show that $y'M(t)y$ is strictly convex. Indeed, for any y and z belonging to X, we have

$$\frac{(y+z)'}{2}M(t)\frac{(y+z)}{2} = \frac{1}{2}y'M(t)y + \frac{1}{2}z'M(t)z - \frac{(y-z)'}{2}M(t)\frac{(y-z)}{2}$$

$$\leq \frac{1}{2}y'M(t)y + \frac{1}{2}z'M(t)z - \frac{1}{4}\alpha\|y-z\|_2^2,$$

and strict convexity follows.

(b) Let $f(z) = (x-z)'M(t)(x-z)$. Notice that $\nabla f(z) = 2M(t)(z-x)$. Using Prop. 3.1, a necessary and sufficient condition for z to minimize f over the set X is

$$2(y-z)'M(t)(z-x) = (y-z)'\nabla f(z) \geq 0$$

for all $y \in X$.

(c) Let x, y be elements of $\Re^n$. From part (b), we have

$$\big(z - [x]_t^+\big)'M(t)\big(x - [x]_t^+\big) \leq 0, \qquad \forall z \in X.$$

Since $[y]_t^+ \in X$, we obtain

$$\big([y]_t^+ - [x]_t^+\big)'M(t)\big(x - [x]_t^+\big) \leq 0.$$

Similarly,

$$\big([x]_t^+ - [y]_t^+\big)'M(t)\big(y - [y]_t^+\big) \leq 0.$$

Adding these two inequalities and rearranging, we obtain

$$\big([y]_t^+ - [x]_t^+\big)'M(t)\big([y]_t^+ - [x]_t^+\big) \leq \big([y]_t^+ - [x]_t^+\big)'M(t)(y-x). \tag{1}$$

The left hand side of Eq. (1) is bounded below by $\alpha\|[y]_t^+ - [x]_t^+\|_2^2$ [cf. Eq. (3.9) in the text]. The right hand side of Eq. (1) is bounded above by $B\|[y]_t^+ - [x]_t^+\|_2 \cdot \|y-x\|_2$, where B is a bound on $\|M(t)\|_2$. We conclude that

$$\big\|[y]_t^t - [x]_t^t\big\|_2 \leq \frac{B}{\alpha}\big\|y-x\big\|_2.$$

(d) We use the norm $\|x\|_t = \big(x'M(t)x\big)^{\frac{1}{2}}$. Then Eq. (1) and the Schwartz inequality [Prop. A.28(e) in Appendix A] yield

$$\big\|[y]_t^+ - [x]_t^+\big\|_t^2 \leq \big\|[y]_t^+ - [x]_t^+\big\|_t \cdot \big\|x-y\big\|_t,$$

from which the result follows.

(e) Let

$$f_t(x,y) = \frac{1}{2\gamma}(y-x)'M(t)(y-x) + (y-x)'\nabla F(x).$$

The gradient $\nabla_2 f_t$ of f_t with respect to y is

$$\nabla_2 f_t(x,y) = \frac{1}{\gamma}M(t)(y-x) + \nabla F(x).$$

We have $T_t(x) = x$ if and only if the minimum of f_t, with respect to y, is attained at $y = x$; that is, if and only if

$$0 \leq (z - x)'\nabla_2 f_t(x, x) = (z - x)'\nabla F(x), \qquad \forall z \in X.$$

[We have used here Prop. 3.1 and the convexity of f_t as a function of y that was proved in part (a)]. The result follows from Prop. 3.1.

(f) We proceed as in part (c). In particular, the optimality conditions for the optimization problem defining $T_t(x)$ and $T_t(y)$ yield

$$0 \leq \big(T_t(y) - T_t(x)\big)'\nabla_2 f_t\big(x, T_t(x)\big) = \big(T_t(y) - T_t(x)\big)'\Big(\frac{1}{\gamma}M(t)\big(T_t(x) - x\big) + \nabla F(x)\Big),$$

$$0 \leq \big(T_t(x) - T_t(y)\big)'\nabla_2 f_t\big(y, T_t(y)\big) = \big(T_t(x) - T_t(y)\big)'\Big(\frac{1}{\gamma}M(t)\big(T_t(y) - y\big) + \nabla F(y)\Big).$$

Adding these two inequalities and rearranging we obtain

$$\begin{aligned}
&\frac{1}{\gamma}\big(T_t(y) - T_t(x)\big)'M(t)\big(T_t(y) - T_t(x)\big) \\
&\qquad \leq \frac{1}{\gamma}\big(T_t(y) - T_t(x)\big)'M(t)(y - x) + \big(T_t(y) - T_t(x)\big)'\big(\nabla F(x) - \nabla F(y)\big) \qquad (2)\\
&\qquad \leq \big\|T_t(y) - T_t(x)\big\|_2\Big(\frac{1}{\gamma}B\|y - x\|_2 + K\|x - y\|_2\Big).
\end{aligned}$$

(We have used here Assumption 3.1.) The left–hand side of Eq. (2) is bounded below by $(\alpha/\gamma)\|T_t(y) - T_t(x)\|_2^2$. Therefore,

$$\frac{\alpha}{\gamma}\Big\|T_t(y) - T_t(x)\Big\|_2 \leq \Big(\frac{B}{\gamma} + K\Big)\big\|y - x\big\|_2.$$

(g) Let $x \in X$. Using the optimality conditions for the problem of Eq. (3.8), we have

$$\begin{aligned}
0 &\leq \Big(x - T_t(x)\Big)'\Big(\frac{1}{\gamma}M(t)(T_t(x) - x) + \nabla F(x)\Big)\\
&\leq -\frac{\alpha}{\gamma}\Big\|T_t(x) - x\Big\|_2^2 + \big(x - T_t(x)\big)'\nabla F(x).
\end{aligned}$$

[We have used here Eq. (3.9).] Thus,

$$(T_t(x) - x)'\nabla F(x) \leq -\frac{\alpha}{\gamma}\big\|T_t(x) - x\big\|_2^2.$$

We then use Prop. A.32 and proceed as in the proof of part (a) of Prop. 3.3.

(h) We have

$$0 \leq \big(y - T_t(x(t))\big)'\Big(M(t)\big(T_t\big(x(t)\big) - x(t)\big) + \gamma\nabla F\big(x(t)\big)\Big), \qquad \forall y \in X. \qquad (3)$$

Let x^* be a limit point of the sequence $x(t)$. Consider a sequence $\{t_k\}$ such that $\{x(t_k)\}$ converges to x^*. Notice that F is bounded below and part (g) implies that $\big\|T_t\big(x(t)\big) - x(t)\big\|_2$ converges to zero. In particular, $T_{t_k}\big(x(t_k)\big)$ converges to x^* as well. Since the set of matrices $\{M(t) \mid t \geq 0\}$ is bounded, the term $\big(y - T_t\big(x(t)\big)\big)M(t)\big(T_t\big(x(t)\big) - x(t)\big)$ converges to zero, for every $y \in X$. Thus, taking the limit in Eq. (3) along the sequence $\{t_k\}$, we obtain $0 \leq (y - x^*)'\nabla F(x^*)$ for all $y \in X$, which concludes the proof.

SECTION 3.4

3.4.1:

(a) We will use Prop. 3.4 once we establish the Lipschitz condition (3.4). This condition is written

$$\|\nabla F(x) - \nabla F(y)\| \leq K\|x - y\|$$

or equivalently,

$$\|P(x - y)\| \leq K\|x - y\|,$$

and is satisfied if K is the maximum eigenvalue of P (see Prop. A.24). The result now follows from Prop. 3.4.

(b) The sum of the diagonal elements of P is equal to the sum of the eigenvalues of P. Since these eigenvalues are all nonnegative, we have that the sum of the diagonal elements of P is an upper bound for K.

(c) We use the transformation of variables $y(t) = M^{1/2}x(t)$. Let

$$H(y) = F(M^{-1/2}y) = \frac{1}{2}y'\bar{P}y + r'M^{-1/2}y,$$

where $\bar{P} = M^{-1/2}PM^{-1/2}$, and note that $\bar{P}$ has ones along its main diagonal. The problem $\min_{x\geq 0} F(x)$ is equivalent to $\min_{y\geq 0} H(y)$. The gradient projection method for the latter problem is given by

$$y(t+1) = \big[y(t) - \gamma\nabla H\big(y(t)\big)\big]^+ \qquad (*)$$

or

$$M^{1/2}x(t+1) = \big[M^{1/2}x(t) - \gamma M^{-1/2}\nabla F\big(M^{-1/2}y(t)\big)\big]^+$$

or

$$x(t+1) = \big[x(t) - \gamma M^{-1}\nabla F\big(x(t)\big)\big]^+, \qquad (**)$$

which is the linearized Jacobi method for the problem $\min_{x\geq 0} F(x)$. By parts (a) and (b), the iteration $(*)$ converges for $\gamma < 2/T$, where T is the sum of diagonal elements of $\bar{P}$, which is equal to n. Therefore, iteration $(**)$ also converges for $\gamma < 2/n$.

3.4.2:

For parts (a)-(d), see the hints. To show part (e), note that we have for all $x^* \in X^*$,

$$F\big((x(t+1)\big) \leq F\big(x(t+1)\big) + \frac{1}{2c(t)}\|x(t+1) - x(t)\|_2^2 \leq F(x^*) + \frac{1}{2c(t)}\|x^* - x(t)\|_2^2.$$

By minimizing over $x^* \in X^*$, we obtain

$$F\big(x(t+1)\big) \leq F^* + \frac{1}{2c(t)}\rho\big(x(t); X^*\big)^2,$$

which is the first relation to be proved. From the hypothesis we also have for $x(t+1)$ within δ of X^*,

$$F^* + \beta\big(\rho(x(t+1); X^*)\big)^\alpha \leq F\big(x(t+1)\big).$$

The last two relations yield

$$\beta\big(\rho(x(t+1); X^*)\big)^\alpha \leq \frac{1}{2c(t)}\rho\big(x(t); X^*\big)^2$$

and

$$\frac{\rho\big(x(t+1); X^*\big)}{\rho\big(x(t); X^*\big)^{2/\alpha}} \leq \frac{1}{\big(2c(t)\beta\big)^{1/\alpha}}.$$

Since $\liminf_{t\to\infty} c(t) > 0$, the desired result follows.

3.4.3:

Follow the hint.

3.4.4:

As in Prop. 4.1(c), it is assumed here that X^* is nonempty. Let us define

$$\bar{x}(t) = \arg\min_{x\in X}\big\{F(x) + \frac{1}{2c}\|x - x(t)\|_2^2\big\}. \qquad (*)$$

Then the algorithm can be written as

$$x(t+1) = \bar{x}(t) + \big(x(t) - \bar{x}(t)\big)\left(1 - \frac{\gamma(t)}{c}\right). \qquad (**)$$

We have, using $(*)$ and $(**)$,

$$\begin{aligned}
F\big(\bar{x}(t+1)\big) &+ \frac{1}{2c}\|\bar{x}(t+1) - x(t+1)\|_2^2 \\
&\leq F\big(\bar{x}(t)\big) + \frac{1}{2c}\|\bar{x}(t) - x(t+1)\|_2^2 \\
&= F\big(\bar{x}(t)\big) + \frac{1}{2c}\left(1 - \frac{\gamma(t)}{c}\right)^2\|\bar{x}(t) - x(t)\|_2^2 \\
&= F\big(\bar{x}(t)\big) + \frac{1}{2c}\|\bar{x}(t) - x(t)\|_2^2 - \frac{1 - \big(1 - \frac{\gamma(t)}{c}\big)^2}{2c}\|\bar{x}(t) - x(t)\|_2^2.
\end{aligned}$$

By adding this relation over $t = 0, 1, \ldots, k$ we obtain

$$\begin{aligned}\min_{x\in X} F(x) &\leq F\big(\bar{x}(k+1)\big) + \frac{1}{2c}\|\bar{x}(k+1) - x(k+1)\|_2^2 \\ &\leq F\big(\bar{x}(0)\big) + \frac{1}{2c}\|\bar{x}(0) - x(0)\|_2^2 - \sum_{t=0}^{k} \frac{1 - \big(1 - \frac{\gamma(t)}{c}\big)^2}{2c}\|\bar{x}(t) - x(t)\|_2^2,\end{aligned}$$

and by taking the limit as $k \to \infty$,

$$\infty > \sum_{t=0}^{\infty} \frac{1 - \big(1 - \frac{\gamma(t)}{c}\big)^2}{2c}\|\bar{x}(t) - x(t)\|_2^2 \geq \sum_{t=0}^{k} \frac{1 - \big(1 - \frac{\delta}{c}\big)^2}{2c}\|\bar{x}(t) - x(t)\|_2^2.$$

It follows that $x(t) - \bar{x}(t) \to 0$.

Proceeding now as in the proof of Prop. 4.1(c) [cf. the argument following Eq. (4.39)], we obtain that every limit point of $\{\bar{x}(t)\}_{t\in T}$ and also of $\{x(t)\}_{t\in T}$ is an optimal solution. This proof also works when c depends on t as long as the sequence $\{c(t)\}$ is monotonically nondecreasing.

3.4.5:

Replace the inequality constraints with equality constraints as indicated, apply the method of multipliers for equality constraints, and perform the minimizations of the Augmented Lagrangian

$$F(x) + \sum_{j=1}^{r} p_j(t)(a_j'x - t_j + w_j) + \frac{c(t)}{2}\sum_{j=1}^{r}(a_j'x - t_j + w_j)^2$$

over (x, w) in two stages. In the first stage, the minimization is carried out over $w_j \geq 0$, $j = 1, \ldots, r$, holding x fixed. The minimum is attained for

$$w_j = \max\big\{0, -\big[p_j/c(t) + a_j'x - t_j\big]\big\}$$

and upon substitution of w_j in the above Augmented Lagrangian, we obtain the modified Augmented Lagrangian given in the exercise. In the second stage the minimization of the modified Augmented Lagrangian is done with respect to x, yielding the iteration of the exercise.

3.4.6:

For every $x^* \in C_1 \cup \cdots \cup C_m$, we have [since $x^* \in C_i$ and $x_i(t)$ is the projection of $x(t)$ on C_i and Prop. 3.2(b) applies]

$$\big(x_i(t) - x(t)\big)'\big(x_i(t) - x^*\big) \leq 0,$$

from which we obtain

$$\|x_i(t) - x(t)\|_2^2 + \|x_i(t) - x^*\|_2^2 \leq \|x(t) - x^*\|_2^2.$$

It follows that

$$\|x_i(t) - x^*\|_2 \leq \|x(t) - x^*\|_2.$$

Therefore

$$\|x(t+1) - x^*\|_2 = \left\| \frac{1}{m} \sum_{i=1}^{m} x_i(t) - x^* \right\|_2 \leq \frac{1}{m} \sum_{i=1}^{m} \|x_i(t) - x^*\|_2 \leq \|x(t) - x^*\|_2. \qquad (*)$$

It follows that $\{x(t)\}$ is bounded and has at least one limit point. By the analysis of Example 4.3, all of its limit points must belong to the intersection $C_1 \cup \cdots \cup C_m$. Suppose there were two limit points, say x^* and $\bar{x}^*$. Then by taking limit in $(*)$ over a subsequence tending to $\bar{x}^*$ and by using also the fact that $\{\|x(t) - x^*\|_2\}$ is monotonically nonincreasing, we obtain

$$\|\bar{x}^* - x^*\|_2 \leq \|x(t) - x^*\|_2, \qquad \forall\ t.$$

By taking limit in the above relation over a subsequence tending to x^*, we obtain $\|\bar{x}^* - x^*\|_2 \leq 0$. Therefore $\bar{x}^* = x^*$ and the limit point of $\{x(t)\}$ is unique.

3.4.7:

Consider the cost function

$$\frac{1}{2} \sum_{i=1}^{m} \lambda_i \|x_i - x\|_2^2$$

in place of the one used in Example 4.3. Then iteration (4.59) remains unchanged, while iteration (4.58) must take the form $x(t+1) = \sum_{i=1}^{m} \lambda_i x_i(t)$. The analysis of Example 4.3 and Exercise 4.6 applies with no essential changes to the modified iteration.

CHAPTER 4

SECTION 4.1

4.1.1:

Since $x_j^0 = \infty$ for $j \neq 0$, we have using Lemma 1.1,

$$x_i^n = w_{i1}^n, \qquad x_i^{n-1} = w_{i1}^{n-1}.$$

The condition $x_i^n < x_i^{n-1}$ implies that there exists a path of n arcs from i to 1 that is shorter than all paths of less than n arcs. This path must contain a negative cycle. Conversely, we have $x_i^k \leq x_i^{k-1}$ for all k and i, so if the condition $x_i^n < x_i^{n-1}$ does not hold for any i, we must have $x_i^{n'} = x_i^n$ for all i and $n' \geq n$. By Lemma 1.1, a negative cycle cannot exist.

4.1.2:

(a) The Gauss-Seidel method is

$$x_i^{k+1} = \min\left[\min_{j\in A(i),j<i}\left(a_{ij} + x_j^{k+1}\right), \min_{j\in A(i),j>i}\left(a_{ij} + x_j^k\right)\right], \qquad i = 2, 3, \ldots, n.$$

Let $J(x)$ and $G(x)$ be the Jacobi and Gauss-Seidel relaxation mappings, and note that J and G are monotone, that is, $J(x') \leq J(x)$ and $G(x') \leq G(x)$ if $x' \leq x$. Furthermore we have

$$G(x) \leq J(x), \qquad \text{if } J(x) \leq x.$$

To see this, note that if $J(x) \leq x$, we have

$$G_2(x) = J_2(x) \leq x_2$$

$$\begin{aligned} G_3(x) &= \min\left[\left(a_{32} + G_2(x)\right), \min_{j\in A(i),j>3}(a_{ij} + x_j)\right] \\ &\leq \min\left[\left(a_{32} + x_2\right), \min_{j\in A(i),j>3}(a_{ij} + x_j)\right] = J_3(x), \end{aligned}$$

and a similar argument shows that $G_i(x) \leq J_i(x)$ for all i. An analogous argument also shows that

$$J(x) \leq G(x), \qquad \text{if } x \leq J(x).$$

Let u be the vector with all coordinates equal to 1 except for $u_1 = 0$. For any $\gamma > 0$ it is seen that we have for all $i \neq 0$,

$$J_i(x^* + \gamma u) = \min \left[a_{i1}, \min_{j \in A(i), j \neq 1} (a_{ij} + x_j^* + \gamma) \right] \leq \min_{j \in A(i)} (a_{ij} + x_j^*) + \gamma = J_i(x^*) + \gamma = x_i^* + \gamma,$$

and similarly, $J_i(x^* - \gamma u) \geq x_i^* - \gamma$. Therefore,

$$x^* - \gamma u \leq J(x^* - \gamma u) \leq x^* \leq J(x^* + \gamma u) \leq x^* + \gamma u,$$

and from this relation it follows that

$$J^k(x^* - \gamma u) \leq G^k(x^* - \gamma u) \leq x^* \leq G^k(x^* - \gamma u) \leq J^k(x^* + \gamma u), \qquad \forall k,$$

where J^k (or G^k) denote the composition of J (or G, respectively) with itself k times. It follows that

$$G^k(x^* - \gamma u) \to x^*, \qquad G^k(x^* + \gamma u) \to x^*.$$

For any initial condition x^0 with $x_1^0 = 0$, we can take γ sufficiently large so that

$$x^* - \gamma u \leq x^0 \leq x^* + \gamma u,$$

and therefore

$$G^k(x^* - \gamma u) \leq G^k(x^0) \leq G^k(x^* + \gamma u),$$

from which we obtain $G^k(x^0) \to x^*$.

(b) The preceding argument shows also that the bounds $G^k(x^* - \gamma u)$ and $G^k(x^* + \gamma u)$ for the Gauss-Seidel method are tighter than the bounds $J^k(x^* - \gamma u)$ and $J^k(x^* + \gamma u)$ for the Jacobi method.

(c) Define

$$S_1 = \{i \mid 1 \in A(i)\}$$

$$S_{k+1} = \{i \mid \text{there exists } j \in S_k \text{ such that } j \in A(i)\}.$$

Consider the node relaxation order $S_1, S_2, \ldots$ for the Gauss-Seidel method, with the node order within each set S_k unspecified. It can be seen by induction on k that at the first iteration of the Gauss-Seidel method we obtain the shortest distances.

4.1.3:

(a) Every node of a given cycle is the start node of one arc and the end node of another arc. Therefore, as the sum of a'_{ij} along a cycle is formed, the quantity p_m is added and subtracted once for every node m of the cycle. Thus the sum of a'_{ij} along a cycle is equal to the sum of a_{ij}.

(b) Each path from i to 1 has length with respect to a_{ij} equal to its length with respect to a'_{ij} plus $p_i - p_1$. This is verified by adding a'_{ij} along the arcs (i,j) of the path and proves the desired result.

4.1.4:

(a) We have for all $i \neq 1$

$$\tilde{x}_i = \min_j (\tilde{a}_{ij} + \tilde{x}_j) = \min_j (a_{ij} + \tilde{x}_j),$$

so the Bellman-Ford algorithm terminates in a single iteration.

(b) For any $i \notin \cup_k N_k$, let p be any shortest path with respect to arc lengths $\tilde{a}_{ij}$. Let d_p be the length of p with respect to arc lengths a_{ij}. We have

$$x_i^* \leq d_p,$$

where x_i^* is the shortest distance of i with respect to arc lengths a_{ij}, while we also have

$$d_p \leq \tilde{x}_i,$$

since the length of any arc of the path p is not increased when changing from arc lengths $\tilde{a}_{ij}$ to a_{ij}. Thus, we obtain $x_i^* \leq \tilde{x}_i = x_i^0$ for all $i \notin \cup_k N_k$. Since we have $x_i^0 = \infty$ for all $i \in \cup_k N_k$, it follows that $x^* \leq x^0$. The result follows from Prop. 1.1(c).

4.1.5:

Each node has at most four incident arcs, so the computation time per iteration and node is $O(1)$. Since there are k^2 nodes assigned to each processor, the (parallel) computation time per iteration is $O(k^2)$. At the end of an iteration each processor must communicate to each neighbor processor the shortest distance estimates of the k nodes that are connected to a node assigned to the neighbor processor. Thus the communication time per iteration is $O(k)$. For k large enough the communication time is negligible relative to the computation time, particularly if packets contain the values of shortest distance estimates of many nodes to reduce the overhead penalty. Thus, for sufficiently large k, the attainable speedup is essentially equal to the number m^2 of processors, regardless of the value of m.

4.1.6:

Each processor (i,j) needs to know $x^k_{i(k+1)}$ and $x^k_{(k+1)j}$ to compute x^{k+1}_{ij}in $O(1)$ time. Therefore, the processors $(k+1,m)$ of the $(k+1)$st row must broadcast to their column hypercubes [processors

(i,m), $i=1,\ldots,n]$ the value $x^k_{(k+1)m}$. This requires n single node broadcasts, which can be done in parallel in $\log n$ time. Also the processors $(m,k+1)$ in the $(k+1)$st column must broadcast to their row hypercubes [processors (m,j), $j=1,\ldots,n]$ the value $x^k_{m(k+1)}$. This requires $\log n$ time also.

4.1.7:

Hint: Let t^k_{ij} be the time that the calculation of x^k_{ij} is completed (with $t^0_{ij}=0$). Show that for $k\geq 0$ and $i\neq j$,

$$t^{k+1}_{ij}=1+\begin{cases}\max\{t^k_{ij},t^k_{i(k+1)}+|j-k-1|,t^k_{(k+1)j}+|i-k-1|\}, & \text{if } i\neq k+1,\ j\neq k+1,\\ \max\{t^k_{ij},t^k_{i(k+1)}\}, & \text{if } j=k+1,\\ \max\{t^k_{ij},t^k_{(k+1)j}\}, & \text{if } i=k+1.\end{cases}$$

Use this equation to verify that for $i\neq j$, we have

$$t^1_{ij}=\begin{cases}1+\max\{|i-1|,|j-1|\}, & \text{if } i\neq 1,\ j\neq 1\\ 1, & \text{otherwise,}\end{cases}$$

and that for $k\geq 1$

$$t^{k+1}_{ij}=3k-1+\begin{cases}\max\{|i-k|+|j-k-1|,|j-k|+|i-k-1|\}, & \text{if } i\neq k+1,\ j\neq k+1,\\ |i-k|, & \text{if } j=k+1,\\ |j-k|, & \text{if } i=k+1.\end{cases}$$

4.1.8:

(a) Suppose every i to j walk contains an arc with weight greater or equal to a_{ij}. Consider an MST and the (unique) walk from i to j on the MST. If this walk does not consist of just arc (i,j), then replace an arc of this walk with weight greater or equal to a_{ij} with arc (i,j), thereby obtaining an MST.

Conversely suppose to obtain a contradiction, that (i,j) belongs to an MST and that there exists a walk W from i to j with all arcs having weight smaller than a_{ij}. We remove (i,j) from the MST obtaining two subtrees T_i and T_j containing i and j, respectively. The walk W must contain an arc that connects a node of T_i to a node of T_j. Adding that arc to the two subtrees T_i and T_j creates a spanning tree with smaller weight than that of the original, which is a contradiction.

(b) Walks from i to j that use nodes 1 through $k+1$ are of two types: 1) walks that use only nodes 1 through k or 2) walks that go from i to $k+1$ using nodes 1 through k and then from $k+1$ to j using nodes 1 through k. The minimum critical weight of walks of type 1) is x^k_{ij}, while the critical weight over walks of type 2) is $\max\{x^k_{i(k+1)},x^k_{(k+1)j}\}$. The characterization of x^k_{ij} given in the exercise follows.

(c) See the hint.

4.1.9:

(a) An easy induction shows that the ijth element of B^k is a one if and only if there is a directed path from i to j with exactly k arcs. We now note that $(I+B)^k = I + B + B^2 + \cdots + B^k$, which in view of the preceding observation implies that the ijth element of $(I+B)^k$ is a one if and only if there is a directed path from i to j with k arcs or less.

(b) In view of the characterization of (a), we have that $(I+B)^k$ stops changing when k becomes larger than $\max_{i,j} d_{ij}$, where d_{ij} is the shortest distance from i to j when all arc lengths are equal to one. The ijth element of the final matrix B^* is a one if and only if either $i = j$ or else there exists a directed path from i to j.

(c) For $i \neq j$, let

$$x_{ij}^1 = \begin{cases} 1, & \text{if } (i,j) \text{ is an arc,} \\ 0, & \text{otherwise,} \end{cases}$$

and for $k \geq 1$,

$$x_{ij}^{k+1} = \big(x_{ij}^k\big)\text{OR}\Big(\big(x_{i(k+1)}\big)\text{AND}\big(x_{(k+1)j}\big)\Big).$$

Then the ijth element of B^* is a one if $i = j$ and is x_{ij}^n if $i \neq j$.

4.1.10:

(a) By construction of the algorithm, for all i, d_i is at all times either ∞ or the length of some path from s to i. Because the arc lengths are all nonnegative, the number of distinct lengths of paths from s to i is finite, and since d_i decreases monotonically, it follows that d_i converges finitely. This means that the set $\tilde{M}$ in Step 1 will be empty after some iteration, implying that the algorithm terminates finitely.

To show that upon termination d_1 equals the shortest distance d^* from s to 1, let

$$(s, i_1), (i_1, i_2), \ldots, (i_m, 1)$$

be a shortest path. Then each path $(s, i_1), (i_1, i_2), \ldots, (i_{k-1}, i_k)$ is a shortest path from s to i_k for all $k = 1, 2, \ldots, m$. If at termination d_1 is larger than d^*, the same will be true throughout the algorithm and therefore $d_1 - h_{i_k}$ will be larger than the length of the path

$$(s, i_1), (i_1, i_2), \ldots, (i_{k-1}, i_k)$$

for all $k = 1, 2, \ldots, m$ throughout the algorithm. It follows that node i_m will never enter the set L with d_{i_m} equal to the shortest distance from s to i_k, since in this case d_1 would be set to d^* at the next iteration. Similarly, this means that node i_{m-1} will never enter the set L with $d_{j_{m-1}}$ equal to the shortest distance from s to i_{m-1}. Proceeding backwards, we conclude that i_1 never enters the set L with d_{i_1} equal to the shortest distance from s to i_1 $(= a_{si_1})$, which is a contradiction because d_{i_1} is set to a_{si_1} at the first iteration.

(b) Similar to (a).

(c) To obtain the shortest path upon termination, give to node j the label "i" each time d_j is decreased in Step 1. A shortest path is then obtained by tracing labels backwards starting from the destination.

SECTION 4.2

4.2.1:

Define

$$\beta = \min_i c_i, \qquad \bar{\beta} = \max_i c_i.$$

The cost vector x^* associated with P, c, and α can be written as

$$x^* = c + \sum_{t=1}^{\infty} a^t P^t c,$$

from which

$$c + \frac{\alpha\beta}{1-\alpha} e \leq x^* \leq c + \frac{\alpha\bar{\beta}}{1-\alpha} e.$$

Using the definition of β and $\bar{\beta}$, we can strengthen this relation as follows

$$\frac{\beta}{1-\alpha} e \leq c + \frac{\alpha\beta}{1-\alpha} e \leq x^* \leq c + \frac{\alpha\bar{\beta}}{1-\alpha} \leq \frac{\bar{\beta}}{1-\alpha}. \qquad (*)$$

We now consider the equation $x^* = c + \alpha P x^*$ and we subtract from it the given equation $\bar{x} = c + \alpha P x$. We obtain

$$x^* - x = (\bar{x} - x) + \alpha P (x^* - x).$$

It follows that $(x^* - x)$ is the unique solution of the equation $y = (\bar{x} - x) + \alpha P y$, i.e., it is the cost vector associated with P, $(\bar{x} - x)$, and α. Therefore, Eq. $(*)$ applies with c replaced by $(\bar{x} - x)$ and x^* replaced by $(x^* - x)$. We thus obtain

$$\frac{\gamma}{1-\alpha} e \leq (\bar{x} - x) + \frac{\alpha\gamma}{1-\alpha} e \leq x^* - x \leq (\bar{x} - x) + \frac{\alpha\bar{\gamma}}{1-\alpha} \leq \frac{\bar{\gamma}}{1-\alpha},$$

where

$$\gamma = \min_i(\bar{x}_i - x_i), \qquad \bar{\gamma} = \max_i(\bar{x}_i - x_i).$$

4.2.2:

Let $c_i = 1$ if $i \neq 1$, $c_1 = 0$, and $\alpha = 1$. With these identifications, t_i is the cost x_i^* associated with the state costs c_i, while the given system of equations coincides with the equation $x = c + \alpha P x$. Therefore the times t_i are the unique solution of this system.

SECTION 4.3

4.3.1:

(a) By applying the mapping T_μ on both sides of the given equation

$$T_\mu(x^*) \leq T(x^*) + \epsilon e,$$

by using the monotonicity property of T_μ (cf. Prop. 3.1), and by using also the fact $T(x^*) = x^*$ we obtain

$$T_\mu^2(x^*) \leq T_\mu\big(T(x^*)\big) + \alpha\epsilon e = T_\mu(x^*) + \alpha\epsilon e \leq T(x^*) + \epsilon e + \alpha\epsilon e = x^* + \epsilon e + \alpha\epsilon e.$$

Applying the same process to this equation, we obtain

$$T_\mu^3(x^*) \leq T_\mu(x^*) + \alpha(e + \alpha\epsilon e) \leq T(x^*) + \epsilon e + \alpha\epsilon e + \alpha^2\epsilon e = x^* + \epsilon e + \alpha\epsilon e + \alpha^2\epsilon e,$$

and by proceeding similarly,

$$T_\mu^t(x^*) \leq x^* + \epsilon\left(\sum_{k=0}^{t-1}\alpha^i\right)e, \qquad \forall\ t \geq 1.$$

Taking limit as $t \to \infty$, we obtain

$$x(\mu) \leq x^* + \frac{\epsilon}{1-\alpha}e.$$

(b) Using the relations $|x_i - x_i^*| \leq \epsilon$ for all i, and $T_\mu(x) = T(x)$, and applying the monotone mappings T and T_μ, we have

$$T_\mu(x^*) \leq T_\mu(x) + \alpha\epsilon e,$$

$$T_\mu(x) = T(x) \leq T(x^*) + \alpha\epsilon e.$$

Adding these two inequalities, we obtain

$$T_\mu(x^*) \leq T(x^*) + 2\alpha\epsilon e,$$

and the result follows by using part (a).

4.3.2:

Since $\{\mu, \mu, \ldots\}$ is improper, there exists a state i such that $[P^t(\mu)]_{i1} = 0$, for all t. Thus with probability one, after a finite number of transitions, the states generated by the system starting from i form a sequence of cycles. Since every cycle has positive cost, the total cost is infinite for almost every sample path of the system starting from state i.

4.3.3:

See the abbreviated proof given in the text.

4.3.4:

See the hint.

4.3.5:

Applying repeatedly the monotone mapping T to the inequality $x \leq T(x)$ we obtain

$$x \leq T(x) \leq T^2(x) \leq \cdots \leq T^t(x) \leq \cdots$$

and by taking the limit as $t \to \infty$ we obtain $x \leq x^*$. The constraint of the linear program can be written as $T(x) \geq x$ and is satisfied by x^*. Furthermore, every feasible solution x of the linear program satisfies $x^* \geq x$ (by what has been proved already), so $\beta x^* \geq \beta x$. It follows that x^* solves the linear program. A similar argument applies for the case $\alpha = 1$ under the Undiscounted Cost Assumption 3.2.

4.3.6:

The constraints of the linear program can be written as

$$C_k^{i1}x_1 + \cdots + C_k^{in}x_n \leq d_k^i, \qquad i = 1, \ldots n,\ k = 1, \ldots, m,$$

and can be put into the form of the constraints of the linear program of Exercise 3.5, which have the form

$$\big(1-\alpha p_{ii}(u)\big)x_i-\sum_{j\neq i}\alpha p_{ij}(u)x_j\leq c_i(u).$$

In particular, identifying u with k, and $c_i(u)$ with d_k^i, we are led to define $p_{ij}(k)$, $i,j=1,\ldots n$, $k=1,\ldots,m$, by the equations

$$1-\alpha p_{ii}(k)=\frac{1-\alpha}{\sum_{t=1}^n C_k^{it}}C_k^{ii}, \qquad (*)$$

$$-\alpha p_{ij}(k)=\frac{1-\alpha}{\sum_{t=1}^n C_k^{it}}C_k^{ij}, \qquad \forall\ j\neq i, \qquad (**)$$

where α is a scalar to be determined. It is seen that, with these definitions, the linear problem is converted into a dynamic programming problem (cf. Exercise 3.5), provided that $\alpha\in[0,1)$, and $p_{ij}(k)$ are legitimate probabilities, that is, they are nonnegative and they add to one [$\sum_{j=1}^n p_{ij}(k)=1$ for all i and k]. The latter property is clearly satisfied from the definition of $p_{ij}(k)$ [add the relations $(*)$ and $(**)$]. Furthermore, we have $p_{ij}(k)\geq 0$ for $j\neq i$, since the off-diagonal elements of C_k are nonpositive, and $\sum_{t=1}^n C_k^{it}>0$ by the diagonal dominance property of C_k. Finally, to ensure that $p_{ii}(k)\geq 0$, we must have

$$0\leq p_{ii}(k)=\frac{1}{\alpha}\left[1-\frac{(1-\alpha)C_k^{ii}}{\sum_{t=1}^n C_k^{it}}\right]$$

or equivalently,

$$\frac{(1-\alpha)C_k^{ii}}{\sum_{t=1}^n C_k^{it}}\leq 1,$$

or equivalently,

$$-\sum_{t\neq i}^{n} C_k^{it}\leq \alpha C_k^{ii}, \qquad \forall\ i,k.$$

This relation is satisfied if

$$\alpha=\max_{i,k}\left[-\frac{\sum_{t\neq i}^n C_k^{it}}{C_k^{ii}}\right].$$

By the diagonal dominance property of C_k, we have $\alpha\in[0,1)$.

4.3.7:

Since T is monotone, continuous, and has a unique fixed point, the conclusions of Exercise 1.4 in Section 3.1 apply.

4.3.8:

(a) Clearly all stationary policies are proper. The cost of a policy $\{\mu,\mu,\ldots\}$ with $\mu(x_2)=u$ satisfies $x_2(\mu)=-u+(1-u^\beta)x_2(\mu)$, from which $x_2(\mu)=-u^{1-\beta}$. Therefore μ yields finite cost.

(b) If $\beta > 1$, then by choosing a policy $\{\mu, \mu, \ldots\}$ with $\mu(x_2)$ sufficiently close to zero, we can attain arbitrarily small cost [cf. part (a)]. Therefore we have $x_2^* = -\infty$, but no stationary policy can be optimal, since all stationary policies have finite cost. Assume now that $\beta = 2$. One can prove by induction that

$$\prod_{k=1}^{\infty}\left(1-\frac{\gamma^2}{k^2}\right) \geq 1-\gamma^2\sum_{k=1}^{\infty}\frac{1}{k^2} = 1-\frac{3\gamma^2}{2} > 0.$$

Using this relation, the cost of the nonstationary policy with $u_k = \gamma/(k+1)$ can be estimated as

$$\begin{aligned}-\Big[(1-\gamma^2)\gamma + (1-\gamma^2)\left(1-\frac{\gamma^2}{4}\right)\frac{\gamma}{2} + (1-\gamma^2)\left(1-\frac{\gamma^2}{4}\right)\left(1-\frac{\gamma^2}{9}\right)\frac{\gamma}{3} + \cdots\Big] \\ \leq -\prod_{k=1}^{\infty}\left(1-\frac{\gamma^2}{k^2}\right)\left(\gamma + \frac{\gamma}{2} + \frac{\gamma}{3} + \cdots\right) = -\infty,\end{aligned}$$

so this policy is optimal.

(c) We first show that $x_2^* = -1$. Since the cost of the stationary policy that applies $u = -1$ at state 2 is -1, it follows that $x_2^* \leq -1$. Assume to obtain a contradiction, that $x_2^* < -1$. When $\beta < 1$, the cost of any policy is at least as large as the cost of the policy when $\beta = 1$ (greater probability of moving from state 2 to the absorbing state for the same cost). Therefore we must have $x_2^* < -1$ when $\beta = 1$. For any policy that applies $u_k \in (0, 1]$ at time k when at state 2, the cost starting at state 2 is

$$-\big[u_0 + u_1(1-u_0) + u_2(1-u_1)(1-u_0) + \cdots\big].$$

Since $x_2^* < -1$ and $u_k \in (0, 1]$, there must exist a policy and a positive integer k such that

$$u_0 + u_1(1-u_0) + u_2(1-u_1)(1-u_0) + \cdots + u_k(1-u_{k-1})(1-u_{k-2})\cdots(1-u_0) > 1. \qquad (*)$$

On the other hand we have, since $u_k \in (0, 1]$, we have

$$\begin{aligned}&u_0 + u_1(1-u_0) + u_2(1-u_1)(1-u_0) + \cdots + u_k(1-u_{k-1})(1-u_{k-2})\cdots(1-u_0) \\ &\quad \leq u_0 + u_1(1-u_0) + u_2(1-u_1)(1-u_0) + \cdots + (1-u_{k-1})(1-u_{k-2})\cdots(1-u_0) \\ &\quad = u_0 + u_1(1-u_0) + u_2(1-u_1)(1-u_0) + \cdots + (1-u_{k-2})(1-u_{k-3})\cdots(1-u_0) \\ &\quad = \ldots = u_0 + u_1(1-u_0) + (1-u_1)(1-u_0) = 1,\end{aligned}$$

contradicting Eq. $(*)$. Therefore $x_2^* = -1$.

Suppose now that $x_1 = 0$ and $x_2 > 0$. Then we have

$$\big[T(x)\big]_2 = x_2 + \inf_{u\in(0,1]}\big[-u - u^\beta x_2\big].$$

The infimum is attained for $u = 1$ and we have $\big[T(x)\big]_2 = -1$. If $x_2 \in [-1, 0]$, then the function $-u - u^\beta x_2$ is concave as a function of u and the infimum over $u \in (0, 1]$ either is attained by $u = 1$ or is approached by $u = 0$. Therefore we have

$$\big[T(x)\big]_2 = x_2 + \min\big[0, -(1+x_2)\big] = x_2 - (1+x_2) = -1.$$

Thus, if $x_2 \geq -1$, the infimum is attained for $u = 1$ and we have $T(x) = x^*$. On the other hand, if $x_2 < -1$, then the infimum in the preceding calculation is approached by $u = 0$ and we get $x = T(x)$.

CHAPTER 5

SECTION 5.1

5.1.1:

Using the conservation of flow constraint, the cost of the transformed problem, for every feasible flow vector f, is written as

$$\begin{aligned}\sum_{(i,j)\in A}(a_{ij}+p_j-p_i)f_{ij} &= \sum_{(i,j)\in A}a_{ij}f_{ij}+\sum_{i\in N}p_i\left(\sum_{\{j|(j,i)\in A\}}f_{ji}-\sum_{\{j|(i,j)\in A\}}f_{ij}\right)\\ &= \sum_{(i,j)\in A}a_{ij}f_{ij}+\sum_{i\in N}p_i s_i.\end{aligned}$$

5.1.2:

Assume f is optimal and let Y be a directed cycle such that

$$\delta=\min\big\{\min\{c_{ij}-f_{ij}\mid(i,j)\in Y^+\},\min\{f_{ij}-b_{ij}\mid(i,j)\in Y^-\}\big\}>0.$$

Let y be the circulation defined by

$$y_{ij}=\begin{cases}\delta, & \text{if } (i,j)\in Y^+,\\ -\delta, & \text{if } (i,j)\in Y^-,\\ 0, & \text{otherwise.}\end{cases}$$

Then $f+y$ is feasible and its cost must be no less than the cost of f, that is,

$$\sum_{(i,j)\in A}a_{ij}(f_{ij}+y_{ij})\geq\sum_{(i,j)\in A}a_{ij}f_{ij},$$

or

$$\sum_{(i,j)\in Y^+}a_{ij}\delta-\sum_{(i,j)\in Y^-}a_{ij}\delta\geq 0.$$

By dividing with δ, we obtain the desired condition

$$\sum_{(i,j)\in Y^+} a_{ij} - \sum_{(i,j)\in Y^-} a_{ij} \geq 0. \qquad (*)$$

[The preceding argument amounts to using Prop. 3.1 in Section 3.3 (cf. the given hint).]

Conversely, if the desired condition $(*)$ holds, and f is not optimal, the argument of the proof of Proposition 3.1 in Section 5.3 [cf. Eq. (3.4)] leads to a contradiction.

5.1.3:

(a) The assignment problem with $B > 1$ is equivalent to the following problem with no upper bounds on the variables

$$\begin{aligned}
&\text{maximize} \quad \sum_{(i,j)\in A} -a_{ij} f_{ij} \\
&\text{subject to} \\
&\sum_{\{j|(i,j)\in A\}} f_{ij} = 1, \qquad \forall\, i = 1, \ldots, n, \\
&\sum_{\{i|(i,j)\in A\}} f_{ij} = 1, \qquad \forall\, j = 1, \ldots, n, \\
&0 \leq f_{ij}, \qquad \forall\, (i,j) \in A.
\end{aligned}$$

A dual problem is (cf. problems (AP) and (DAP) in Appendix C)

$$\begin{aligned}
&\text{minimize} \quad -\sum_{i=1}^{n} r_i + \sum_{j=1}^{n} p_j \\
&\text{subject to} \quad a_{ij} + p_j \geq r_i, \qquad \forall\, (i,j) \in A,
\end{aligned}$$

which can be expressed in the desired format

$$\begin{aligned}
&\text{maximize} \quad \sum_{i=1}^{n} r_i - \sum_{j=1}^{n} p_j \\
&\text{subject to} \quad a_{ij} + p_j \geq r_i, \qquad \forall\, (i,j) \in A.
\end{aligned}$$

(b) Consider the assignment problem with sources i and sinks i' formulated in the hint. Its dual problem is

$$\begin{aligned}
&\text{maximize} \quad \sum_{i=1}^{|N|} p_i - \sum_{i=1}^{|N|} p'_i \\
&\text{subject to} \quad a_{ij} + p'_j \geq p_i, \qquad \forall\, (i,j) \in A, \qquad p'_i \geq p_i, \qquad \forall\, i \in N,
\end{aligned}$$

where p'_i denotes the price of i'. Let S be the feasible assignment that assigns each source i with sink i'. Assume that each positive cycle in problem (LNF) has nonnegative arc cost sum. Then

for the assignment S, all cycles of the assignment problem satisfy the condition of Exercise 1.2 (cf. the figure). Therefore from Exercise 1.2, S is optimal, which shows that there exists a dual optimal solution $\{(p_i, p_i') \mid i = 1, \ldots, n\}$ for which $p_i = p_i'$. Therefore from the first constraint of the dual problem, we obtain

$$a_{ij} + p_j \geq p_i, \qquad \forall\ (i,j) \in A,$$

which implies that all arcs $(i,j) \in A$ are inactive or balanced.

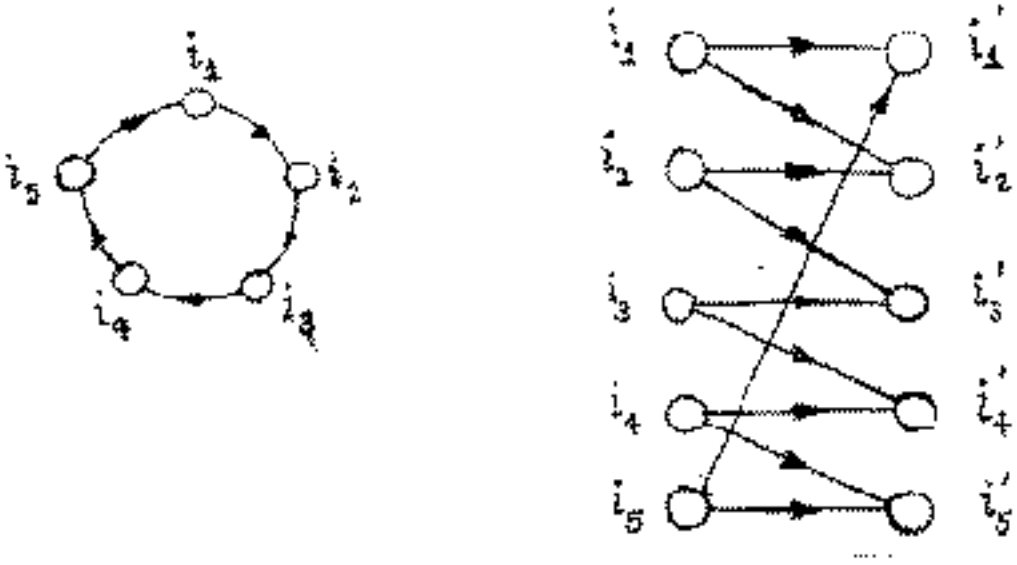

Figure For Exercise 5.1.3. Cycles of problem (LNF) and corresponding cycles of the assignment problem formulated in part (b).

5.1.4:

It is seen that an optimal flow vector f of the enlarged network problem yields an optimal flow vector of the original problem (LNF), if and only if it satisfies $f_{wv} = (1/2)\sum_{i \in N} |s_i|$. We will show that under the given condition on a_{wv}, an optimal flow vector f of the enlarged network problem satisfies $f_{wv} = (1/2)\sum_{i \in N} |s_i|$. Indeed, suppose to arrive at a contradiction, that f is an optimal flow vector of the enlarged network problem and $f_{wv} < \frac{1}{2}\sum_{i \in N} |s_i|$. Then, there must exist an augmenting path P starting at v and ending at w, since problem (LNF) is feasible (see Section 5.2 for the definition of an augmenting path). Since f is optimal for the enlarged network problem, there is a set of prices p_i, $i \in N$, and p_v, p_w, satisfying complementary slackness together with f. By adding the

complementary slackness condition along the cycle formed by P and arc (w, v), we obtain

$$0 \leq a_{wv} + \sum_{(i,j)\in P^+} a_{ij} - \sum_{(i,j)\in P^-} a_{ij} < -L + \sum_{(i,j)\in P^+} a_{ij} - \sum_{(i,j)\in P^-} a_{ij}.$$

This contradicts the definition $L = \max_p L_p$.

5.1.5:

Replace the kth arc with start node i and end node j with a node (ijk) and two arcs $\big(i,(ijk)\big)$ and $\big((ijk),j\big)$ with arc flow bounds and arc cost coefficient equal to the ones of the original arc.

5.1.6:

See the hint.

5.1.7:

Assign Lagrange multipliers $r_i(t)$ and $p_j(t)$ to the conservation of flow equations corresponding to the ith source and jth sink, respectively, and consider the Augmented Lagrangian

$$\sum_{(i,j)}[a_{ij} - r_i(t) - p_j(t)] \sum_{i=1}^{m} r_i(t) + \sum_{j=1}^{n} p_j(t) + \frac{c(t)}{2}\left[\sum_{i=1}^{m}\big(\alpha_i - \sum_j f_{ij}\big)^2 + \sum_{j=1}^{n}\big(\beta_j - \sum_i f_{ij}\big)^2\right].$$

The first method of multipliers of Example 4.5 in Subsection 3.4.4 consists of the Gauss-Seidel method for minimizing the Augmented Lagrangian. Minimization with respect to a single arc flow variable f_{ij}, while keeping all all other arc flows fixed, is carried out by setting the first derivative of the Augmented Lagrangian with respect to f_{ij} to zero, that is,

$$a_{ij} - r_i(t) - p_j(t) - c(t)\big(\alpha_i - \sum_j f_{ij}\big) - c(t)\big(\beta_j - \sum_i f_{ij}\big) = 0,$$

solving with respect to f_{ij}, which after a little calculation yields

$$f_{ij} := f_{ij} + \frac{1}{2c(t)}\big(r_i(t) + p_j(t) - a_{ij} + c(t)(y_i + w_j)\big),$$

and finally projecting the result on the flow range $[0, c_{ij}]$. This is precisely the iteration given in the exercise.

5.1.8:

The Augmented Lagrangian is the same as in Exercise 1.7. Application of iteration (4.75) of Subsection 3.4.4 verifies the desired equivalence.

SECTION 5.2

5.2.1:

Let p_n^t and p_n^0 be the prices of a node n at time t and at time 0. Consider a node i with positive surplus at time t. We claim that there exists a path $(i, i_1, i_2, \ldots, i_k, j)$ from i to some negative surplus node j consisting of forward arcs that are balanced or inactive, and backward arcs that are balanced or active, i.e.

$$\begin{aligned} p_i^t &\le p_{i_1}^t + a_{ii_1} \\ p_{i_1}^t &\le p_{i_2}^t + a_{i_1 i_2} \\ &\cdots \\ p_{i_k}^t &\le p_j^t + a_{i_k j}. \end{aligned} \qquad (*)$$

Such a path can be constructed by using the following algorithm in which initially we have $L = \{i\}$.

Step 1: If there exists an arc (k, j) [or arc (j, k)] such that $k \in L$, $j \notin L$, and (k, j) is balanced or inactive [or (j, k) is balanced and active, respectively], set $L := \{j\} \cup L$ and give the label "k" to j. Go to Step 2.

Step 2: If $g_j < 0$, construct the desired path by tracing labels backwards from j to i, and terminate the algorithm. Otherwise go to Step 1.

The algorithm will terminate, that is, it will always find a node j in Step 1, because otherwise there would exist a set L with $\sum_{k \notin L} g_k > 0$ and all outgoing (or incoming) of L are active (or inactive, respectively). This would imply that

$$0 < \sum_{k \notin L} g_k = \sum_{k \notin L} s_k - \sum_{\{(k,j) | k \in L, j \notin L\}} c_{kj} + \sum_{\{(j,k) | k \in L, j \notin L\}} b_{jk},$$

which contradicts the feasibility of the problem.

By adding the relations (*) and by taking into account the fact $p_j^t = p_j^0$ (since j has negative surplus at time t it cannot have experienced any price rise up to time t according to the rules of the algorithm), we obtain

$$p_i^t \le p_j^0 + \text{Length of path } (i, i_1, \ldots, i_k, j).$$

It follows that

$$p_i^t - p_i^0 \le p_j^0 - p_i^0 + \text{Length of path } (i, i_1, \ldots, i_k, j) \le \max_{j \in S} p_j - \min_{i \in N} p_i + L.$$

SECTION 5.3

5.3.10:

We group each set of variables $\{f_{ij} \mid j \in O(i)\}$ as a subvector that is constrained in the set

$$P_i = \{f_{ij} \mid f_{ij} \geq 0, \forall\ j \in O(i), \sum_{j \in O(i)} f_{ij} = \alpha_i\}$$

and we write the transportation problem as

$$\begin{aligned} \min \quad & \sum_{(i,j) \in A} a_{ij} f_{ij} \\ & f_{ij} = z_{ij}, \qquad \forall\ (i,j) \in A \\ & \sum_{i \in I(j)} z_{ij} = \beta_j, \qquad \forall\ j = 1, \ldots, n \\ & \{f_{ij} \mid j \in O(i)\} \in P_i, \qquad \forall\ i = 1, \ldots, m. \end{aligned}$$

We now apply the procedure for separable problems given in Example 4.4 of Subsection 3.4.4 and its alternating direction method counterpart.

SECTION 5.4

5.4.5:

(a) Even though $\epsilon = 1$ is the value used in the scaling algorithm, we will initially argue in terms of a general ϵ, specializing to the case $\epsilon = 1$ only when needed. Let f^0 be the complete assignment resulting from the previous subproblem, and let f be the current assignment. Consider the flow vector $w = f^0 - f$. The elements of w take the values $+1$, -1, or 0. For each person node i that is unassigned under f, there is a path $P(i)$ of alternating $+1$ (forward) and -1 (backward) arcs in w that terminates at an object node $j(i)$ that is also unassigned under f. We claim that for any pair i_1, i_2 of unassigned persons, the paths $P(i_1)$ and $P(i_2)$ are node-disjoint. To see this, note that the forward arcs of these paths correspond to assignment under f^0 and lack of assignment under f, while

the backward arcs correspond to lack of assignment under f^0 and assignment under f. Therefore, if k were the first node shared by $P(i_1)$ and $P(i_2)$ that would mean that either k was assigned twice under f^0, if k were an object node, or that k is assigned twice under f, if k is a person node.

Using the definition of π_i, it is seen that for the forward arcs (i,j) in $P(i)$ we have $\pi_i + p_j \geq a_{ij}$, while for the backward arcs we have, by ϵ-CS, $\pi_i + p_j \leq a_{ij} + \epsilon$. Alternately subtracting and adding these conditions along $P(i)$, we obtain

$$-\pi_i - p_{j(i)} \leq -\sum_{(k,j)\in P(i)} a_{kj} w_{kj} + \epsilon B(i),$$

where $B(i)$ is the number of backward arcs in $P(i)$. Since after scaling by a factor of two the arc costs and prices from the previous subproblem, f^0 and p^0 satisfy 3ϵ-CS, we similarly obtain

$$\pi_i^0 + p_{j(i)}^0 \leq \sum_{(k,j)\in P(i)} a_{kj} w_{kj} + 3\epsilon F(i),$$

where $F(i)$ is the number of forward arcs in $P(i)$. Since $p_{j(i)} = p_{j(i)}^0$, by combining these two equations, it follows that

$$\pi_i^0 - \pi_i \leq 4\epsilon |P(i)|,$$

where $|P(i)|$ is the number of nodes in $P(i)$. Summing over the set of all unassigned persons I,

$$\sum_{i\in I} \big(\pi_i^0 - \pi_i\big) \leq 3\epsilon \sum_{i\in I} |P(i)|.$$

Since the paths $P(i)$ are all node-disjoint and the total number of nodes is $2n$, we obtain the desired relation

$$\sum_{i\in I} \big(\pi_i^0 - \pi_i\big) \leq 6\epsilon n.$$

(b) The implementation is similar to the $O(n|A|)$ implementation of the auction algorithm, except that each time a person node performs an iteration, it saves the current value of $\pi_i^0 - \pi_i$ in a special variable, say R_i. The algorithm also maintains a set S of person nodes i that are unassigned and satisfy $R_i \leq (6n)^{1/2}\epsilon$. Every time an assignment (i,j) is broken, we examine R_i and insert i into S if $R_i \leq (6n)^{1/2}\epsilon$. A person node is removed from S each time it bids. At the start of the subproblem, S contains all unassigned persons. Since there is no danger of inserting a node i into S that is already contained in S, the set insertion/deletion operations can be implemented in $O(1)$ time by any number of simple strategies. Thus the work of locating suitable nodes for which to bid is $O(1)$ per bid. By using a strategy similar to that of the $O(n|A|)$ implementation of the auction algorithm, one can insure that the work performed at person i between successive drops in π_i is only $O\big(\deg(i)\big)$ and that each drop is by at least ϵ, where $\deg(i)$ is the degree of node i. Therefore, at most $(6n)^{1/2}+1$

iterations are performed at each person node i, and the total work is $O(n^{1/2}|A|)$. (The "+1" comes in because R_i is an underestimate of $\pi_i^0 - \pi_i$, so one iteration at i might begin when $\pi_i^0 - \pi_i > (6n)^{1/2}\epsilon$. In such iterations, no bid need be placed once $\pi_i^0 - \pi_i$ has been computed and discovered to be too high.) At termination, we have

$$\sum_{i \in I} \big(\pi_i^0 - \pi_i\big) \geq (6n)^{1/2}\epsilon |I|,$$

where I is the set of unassigned persons, and $|I|$ is the number of nodes in I. In view of part (a), it follows that $|I| \leq (6n)^{1/2}$.

(c) Within each scaling subproblem, we first run the modified auction algorithm of part (b), which takes $O\big(n^{1/2}|A|\big)$ time. We then repeatedly apply algorithm X until the assignment is complete. Since no more than $(6n)^{1/2}$ unassigned persons can remain after the modified auction ends, algorithm X is used $O\big(n^{1/2}\big)$ times, for a total of $O\big(n^{1/2}|A|\big)$ work. Thus, each subproblem takes $O\big(n^{1/2}|A|\big)$ time for a total time of $O\big(n^{1/2}|A|\log(nC)\big)$.

SECTION 5.5

5.5.1:

(a) The Lagrangian function is

$$L(x,p) = \sum_{j=1}^{n} f_j(x_j) + \sum_{i=1}^{m} p_i\big(s_i - \sum_{j=1}^{n} a_{ij}x_j\big)$$

and the dual functional is

$$q(p) = \sum_{j=1}^{n} \min_{b_j \leq x_j \leq c_j} \{f_j(x_j) - (\sum_{i=1}^{m} a_{ij}p_i)x_j\} + \sum_{i=1}^{m} p_i s_i.$$

(b) The partial derivatives of the dual function are

$$\frac{\partial q(p)}{\partial p_i} = s_i - \sum_{j=1}^{n} a_{ij}x_j(p),$$

where $x_j(p)$ are the unique scalars attaining the minimum in the above definition of the dual functional. The relaxation method is highly parallelizable if the dependency graph defined by the matrix A is sparse.

(c) Here the Lagrangian function is

$$L(x,p) = f(x) + \sum_{i=1}^{m} p_i\big(s_i - \sum_{j=1}^{n} a_{ij}x_j\big)$$

and the dual functional is

$$q(p) = \min_{b_j \le x_j \le c_j} \{f(x) - \sum_{j=1}^{n}(\sum_{i=1}^{m} a_{ij}p_i)x_j\} + \sum_{i=1}^{m} p_i s_i.$$

Again the relaxation method is highly parallelizable if the dependency graph defined by the matrix A is sparse.

5.5.2:

Follow the hint and the proof of Prop. 3.1 in Section 5.3.

5.5.4:

Consider the optimal network flow problem of the hint. Clearly the primal optimal solution is the zero flow, and from the complementary slackness conditions, we obtain that an optimal price vector satisfies $p_i - p_j = 0$ for all arcs (i,j). Since the graph is connected, it follows that all optimal price vectors have equal coordinates. Therefore it will be sufficient to show that the iteration of the exercise can be viewed as a relaxation method for which Prop. 5.2(d) applies. Let $p(t)$ be the price vector after t iterations. The corresponding surplus at node i [cf. Eqs. (5.15) and (5.16)] is given by

$$g_i\big(p(t)\big) = \sum_{j\in A(i)} a_{ji}\big[p_j(t) - p_i(t)\big] - \sum_{j\in A(i)} a_{ij}\big[p_i(t) - p_j(t)\big], \qquad (*)$$

where $A(i)$ is the set of neighbor nodes of i. Using the fact $a_{ij} = a_{ji}$ and $\sum_{j\in A(i)} a_{ij} = 1$, we obtain

$$g_i\big(p(t)\big) = -2p_i(t) + 2\sum_{j\in A(i)} a_{ij}p_j(t).$$

A relaxation method for which Prop. 5.2(d) applies, selects at iteration $t+1$ node i and leaves p_j unchanged for $j \neq i$, and sets p_i to a value $p_i(t+1)$ so that $g_i\big(p(t+1)\big) = \delta_i g_i\big(p(t)\big)$, where $\delta_i \in [0,1)$. Using Eq. $(*)$, this condition is written as

$$-2p_i(t+1) + 2\sum_{j\in A(i)} a_{ij}p_j(t) = \delta_i\big(-2p_i(t) + 2\sum_{j\in A(i)} a_{ij}p_j(t)\big),$$

or equivalently

$$p_i(t+1) = \delta_i p_i(t) + (1-\delta_i)\sum_{j\in A(i)} a_{ij}p_j(t),$$

which is the iteration given in the exercise.

5.5.5:

The condition $a_{ij}^- \leq p_i - p_j \leq a_{ij}^+$ is the complementary slackness condition for $f = 0$ and p in connection with the problem

$$\begin{aligned} \text{minimize} \quad & \sum_{(i,j)\in A} \max\{a_{ij}^- f_{ij}, a_{ij}^+ f_{ij}\} \\ \text{subject to} \quad & \sum_{\{j|(i,j)\in A\}} f_{ij} = \sum_{\{j|(j,i)\in A\}} f_{ji}, \qquad \forall\, i \in N, \\ & -1 \leq f_{ij} \leq 1, \qquad \forall\, (i,j) \in A. \end{aligned}$$

We have that $f = 0$ is a primal optimal solution if and only if the condition of Exercise 5.5.2 holds, and when this condition is applied to the preceding problem, it is equivalent to the condition (5.33) of the exercise.

5.5.6:

See the hint.

5.5.7:

(a) Clearly $\gamma_{ij} > 0$, since to increase f_{ij} (decrease f_{ji}), we must increase p_i. If $\gamma_{ij} = \infty$ for all (i,j) with $f_{ij} < c_{ij}$ and $\gamma_{ji} = \infty$ for all (j,i) with $f_{ji} > b_{ji}$, we must have for all $\gamma \geq 0$

$$f_{ij}(\gamma) - f_{ij} < \frac{\mu}{n_i} g_i(p), \qquad \forall\, (i,j) \text{ with } f_{ij} < c_{ij},$$

$$f_{ji} - f_{ji}(\gamma) < \frac{\mu}{n_i} g_i(p), \qquad \forall\, (j,i) \text{ with } f_{ji} > b_{ji}.$$

Adding and taking into account the fact $f_{ij}(\gamma) = f_{ij}$ if $f_{ij} = c_{ij}$ and $f_{ji}(\gamma) = f_{ji}$ if $f_{ji} = b_{ji}$, we obtain

$$-g_i(p + \gamma e_i) + g_i(p) < \mu g_i(p), \qquad \forall\, \gamma \geq 0,$$

$$0 < (1-\mu) g_i(p) < g_i(p + \gamma e_i), \qquad \forall\, \gamma \geq 0,$$

which contradicts feasibility of the primal problem. Therefore, $\bar{\gamma} > 0$.

We have

$$0 \leq f_{ij}(\bar{\gamma}) - f_{ij} \leq \frac{1}{n_i} g_i(p), \qquad \forall\, (i,j) \text{ with } f_{ij} < c_{ij},$$

$$0 \leq f_{ji} - f_{ji}(\bar{\gamma}) \leq \frac{1}{n_i} g_i(p), \qquad \forall\, (j,i) \text{ with } f_{ji} > b_{ji},$$

and for an arc (i, j) (or (j, i)) for which $\gamma_{ij} = \bar{\gamma}$ (or $\gamma_{ji} = \bar{\gamma}$) we have

$$\frac{\mu}{n_i} g_i(p) \leq f_{ij}(\bar{\gamma}) - f_{ij}.$$

Therefore, we have

$$\frac{\mu}{n_i} g_i(p) \leq -g_i(\bar{p}) + g_i(p) \leq g_i(p).$$

Hence

$$0 \leq g_i(\bar{p}) \leq \left(1 - \frac{\mu}{n_i}\right) g_i(p).$$

CHAPTER 6

SECTION 6.2

6.2.1:

The synchronous convergence condition in Assumption 2.1 should be modified to read: "There exists a sequence $\{s_k\}$ with $s_0 = 0$ and $s_k \to \infty$, such that

$$f(x,t) \in X(k+1), \qquad \forall\ k,\ x \in X(k),\ t \geq s_k.$$

Furthermore, if $\{y^m\}$ is a sequence such that for every k, we have $y^m \in X(k)$ for all k larger than some index m_k, then every limit point of $\{y^m\}$ is a fixed point of f."

The statement of the asynchronous convergence theorem remains unchanged and its proof is very similar as in the text.

6.2.2:

Eq. (2.2) should be changed to

$$f(x^1, x^2, \ldots, x^m) \in X(k+1), \qquad \forall\ k,\ x^i \in X(k).$$

The statement of the asynchronous convergence theorem remains unchanged and its proof is very similar as in the text.

6.2.3:

Eq. (2.2) should be changed to

$$f(x) \subset X(k+1), \qquad \forall\ k,\ x \in X(k).$$

The statement of the asynchronous convergence theorem remains unchanged and its proof is very similar as in the text.

6.2.4:

For this example, $(0,0)$ is the unique fixed point, and convergence occurs because $x_1(t)$ will be zero after its first update. On the other hand, if $P_2[X(k)] \neq \{x \mid x_1 = 0\}$ and the sets $X(k)$ satisfy the box condition, we can't have $f[X(k)] \subset X(k)$.

SECTION 6.3

6.3.1:

For the function $f : \Re^2 \mapsto \Re^2$ given by $f_1(x_1, x_2) = 0$ and $f_2(x_1, x_2) = x_1 x_2$, $x^* = (0,0)$ is the unique fixed point and all the sequences generated by the asynchronous iteration (1.1) converge to x^*. We show that there exists no $w > 0$ such that for all $x \neq 0$, we have

$$\frac{1}{w_2}|x_1 x_2| < \max\left\{\frac{|x_1|}{w_1}, \frac{|x_2|}{w_2}\right\}.$$

Choose $x_1 = 2$, $x_2 = 1$. Then we obtain $(2/w_2) < \max\{2/w_1, 1/w_2\}$ and $w_1 < w_2$. Choose $x_1 = 1$, $x_2 = 2$ and we obtain $w_2 < w_1$, a contradiction.

6.3.2:

The analog of Prop. 3.2 is:

There holds $\|x(t) - x^*\|_\infty \leq \rho_A^t \|x(0) - x^*\|_\infty$, where ρ_A is the unique nonnegative solution of

$$\rho = \max_{k=1,\ldots,m}\{\alpha_k \rho^{-b_k}\}$$

if for all i

$$|f_i(x) - x_i^*| \leq \max_{k=1,\ldots,m}\left\{\alpha_k \max_{j \in F_k(i)} |x_j - x_j^*|\right\}, \qquad (*)$$

and ρ_A is the unique nonnegative solution of

$$\rho = \alpha_1 \rho^{-b_1} + \alpha_2 \rho^{-b_2} + \cdots + \alpha_m \rho^{-b_m}$$

if for all i

$$|f_i(x) - x_i^*| \leq \sum_{k=1}^{m} \alpha_k \max_{j \in F_k(i)} |x_j - x_j^*|. \qquad (**)$$

Under Eq. $(*)$ the result is proved by induction on t as follows. The result holds for $t = 0$. Assume that it holds for all $t \leq \bar{t}$. Then we have

$$x_i(\bar{t}+1) = f_i\left(x_1\big(\tau_1^i(\bar{t})\big), \ldots, x_n\big(\tau_n^i(\bar{t})\big)\right),$$

and by using the induction hypothesis, we obtain

$$\begin{aligned}
|x_i(\bar{t}+1) - x_i^*| &= |f_i\left(x_1\big(\tau_1^i(\bar{t})\big), \ldots, x_n\big(\tau_n^i(\bar{t})\big)\right) - x_i^*| \\
&\leq \max_{k=1,\ldots,m} \left\{ \alpha_k \max_{j \in F_k(i)} |x_j(\tau_j^i(\bar{t})) - x_j^*| \right\} \\
&\leq \max_{k=1,\ldots,m} \left\{ \alpha_k \rho^{\bar{t}-b_k} \right\} \|x(0) - x^*\|_\infty \\
&= \max_{k=1,\ldots,m} \left\{ \alpha_k \rho^{-b_k} \right\} \rho^{\bar{t}} \|x(0) - x^*\|_\infty = \rho^{\bar{t}+1} \|x(0) - x^*\|_\infty,
\end{aligned}$$

completing the induction proof. The proof under Eq. $(**)$ is similar.

SECTION 6.4

6.4.1:

Let

$$f_i(p) = \frac{t_i}{\prod_{j \in S_i}(1 - p_j)}.$$

If there exists a set of feasible probabilities p_j^*, we have $p^* = f(p^*)$. Furthermore, f is monotone. Therefore

$$0 \leq f(0) \leq f^2(0) \leq \cdots \leq p^*.$$

Hence $f^k(0)$ will converge to a vector of probabilities $\tilde{p}$. The asynchronous algorithm will also converge to $\tilde{p}$.

6.4.2:

See the hint.

CHAPTER 7

SECTION 7.1

7.1.1:

Let us fix some integer $B > 0$ and consider a time interval during which the index variable t advances by B units. Whenever t gets incremented, some processor has completed an update. It follows that during the time interval under consideration, some processor has completed at least B/n updates. Since each update takes at least A_3 (local) time units, the local clock of such a processor must have advanced by at least $A_3\big((B/n) - 1\big)$ units. It follows that the local clock of every other processor had advanced by at least $(A_1A_3/A_2)\big((B/n) - 1\big)$. In particular, if B is large enough so that $(A_1A_3/A_2)\big((B/n)-1\big) \geq A_4+A_5$, then every other processor has performed at least one update and Assumption 1.1(a) is satisfied.

We now turn to Assumption 1.1(b). Each processor broadcasts information at least once every A_6 local time units; that is, at least once every A_6/A_1 global time units. This information is received by every other processor at most A_7 units later. It follows that the information available to any processor can be outdated by at most A_6/A_1+A_7 global time units. Upon receipt of that information, a processor may take up to $A_4 + A_5$ additional local time units to complete the next update [that is, up to $(A_4 + A_5)/A_1$ global time units]. Thus, the information used in an update can be outdated by at most $C = (A_6/A_1) + A_7 + (A_4 + A_5)/A_1$ global time units. In the meanwhile, each processor may have performed up to $A_2C/A_3 + 1$ updates and the index variable may have advanced by at most $nA_2C/A_3 + n$.

We conclude that parts (a) and (b) of Assumption 1.1 hold provided that

$$B \geq \max\left\{\left(\frac{(A_4 + A_5)A_2}{A_1A_3} + 1\right) n,\ n\frac{A_2C}{A_3} + n\right\}.$$

7.1.2:

(a) See parts (b) and (c) of Prop. 2.1 in Section 7.2.

(b) If c is a positive scalar then $Z^* = \{cz^* \mid z^* \in Z^*\}$, because Z^* is a subspace. Thus,

$$d(cz) = \inf_{z^* \in Z^*} \|cz - z^*\| = \inf_{z^* \in Z^*} \|cz - cz^*\| = \inf_{z^* \in Z^*} c\|z - z^*\| = cd(z).$$

Fix some $z^* \in Z^*$. Since Z^* is a subspace we have $z^* + w^* \in Z^*$ if and only if $w^* \in Z^*$. Thus,

$$d(z - z^*) = \inf_{w^* \in Z^*} \|z - z^* - w^*\| = \inf_{\{z^* + w^* | w^* \in Z^*\}} \|z - z^* - w^*\| = \inf_{y^* \in Z^*} \|z - y^*\| = d(z).$$

(c) For any $z^* \in Z^*$, we define $S(z^*) = \big\{z \in Z \mid d(z) = \|z - z^*\|\big\}$. For any scalar c, we let $R(c) = \big\{z \in Z \mid d(z) = c\big\}$. The set $S(z^*)$ is closed because the functions d and $\|\cdot\|$ are continuous. Similarly, the set $R(c)$ is closed. It follows that the set $S(0) \cap R(1)$ is closed. Furthermore, if $z \in S(0) \cap R(1)$, then $\|z\| = 1$ and the set $S(0) \cap R(1)$ is bounded. We conclude that the set $S(0) \cap R(1)$ is compact.

Since condition (a) of Prop. 1.1 is assumed to hold, we have $d\big(z(t^*)\big) < d\big(z(0)\big)$ for each $z(0) \in S(0) \cap R(1)$ and for each scenario. As far as the behavior of the algorithm during the first t^* time units is concerned, there is only a finite number of different scenarios. For any $z(0) \notin Z^*$ let

$$\rho\big(z(0)\big) = \max \frac{d\big(z(t^*)\big)}{d\big(z(0)\big)},$$

where the maximum is taken over all possible scenarios. This is the maximum of finitely many numbers smaller than 1. It follows that $\rho\big(z(0)\big) < 1$ for every $z(0) \notin Z^*$.

Since the function $f(x) = Ax$ is continuous, it follows that $z(t^*)$ is a continuous function of $z(0)$, for every scenario. Since the function d is also continuous and since the maximum of a finite number of continuous functions is continuous, we conclude that ρ is a continuous function of $z(0)$, over the set of all $z(0)$ that do not belong to Z^*. Let $\rho = \sup_{z \in S(0) \cap R(1)} \rho(z)$. Here we are maximizing a continuous function over a compact set and it follows that the supremum is attained. Thus, $\rho < 1$.

Let c be some positive scalar and consider some $z(0) \in S(0) \cap R(c)$. We will show that $z(0)/c \in S(0) \cap R(1)$. By part (b), $d\big(z(0)/c\big) = d\big(z(0)\big)/c = 1$, which shows that $z(0)/c \in R(1)$. Furthermore, $\|z(0)/c\| = \|z(0)\|/c = d\big(z(0)\big)/c = d\big(z(0)/c\big)$, which shows that $z(0)/c \in S(0)$. We conclude that $z(0)/c \in S(0) \cap R(1)$.

Fix some scenario and some $z(0) \in S(0) \cap R(c)$, and let $z(t^*)$ be the vector generated by the asynchronous iteration after t^* time units. Because of the linearity of the iteration, if $z(0)$ is replaced by $z(0)/c$, then $z(t^*)$ should be replaced by $z(t^*)/c$. Thus,

$$\frac{d\big(z(t^*)\big)}{d\big(z(0)\big)} = \frac{d\big(z(t^*)/c\big)}{d\big(z(0)/c\big)} \leq \rho\big(z(0)/c\big) \leq \rho < 1,$$

where the inequalities follow from the definition of ρ and the fact $z(0)/c \in S(0) \cap R(1)$.

Consider now some arbitrary $z(0) \notin Z^*$ and let $z^* \in Z^*$ be such that $d\big(z(0)\big) = \|z(0) - z^*\|$. Notice that $d\big(z(0) - z^*\big) = d\big(z(0)\big) = \|z(0) - z^*\|$ and this implies that $z(0) - z^* \in S(0)$. Fix a scenario. If the iteration was initialized with z^* instead of $z(0)$, the value of $z(t)$ would be equal to z^* for all times. Using the linearity of the iteration, we conclude that if $z(0)$ was replaced by $z(0) - z^*$ then $z(t^*)$ would be replaced by $z(t^*) - z^*$. We then have

$$\frac{d\big(z(t^*)\big)}{d\big(z(0)\big)} = \frac{d\big(z(t^*) - z^*\big)}{d\big(z(0) - z^*\big)} \leq \rho\big(z(0) - z^*\big).$$

Since $z(0) - z^* \in S(0)$, we use the result of the previous paragraph to conclude that $\rho\big(z(t^*) - z^*\big) \leq \rho < 1$.

To summarize, we have shown that, for all $z(0) \notin Z^*$ and all scenarios, we have $d\big(z(t^*)\big) \leq \rho d\big(z(0)\big)$ and $\rho < 1$. Since the description of the asynchronous iteration does not depend explicitly on t, we can also conclude that $d\big(z(t+t^*)\big) \leq \rho d\big(z(t)\big)$ for all t. It follows that $d\big(z(t)\big) \leq \rho^{\lfloor t/t^* \rfloor} d\big(z(0)\big)$, which shows that $d\big(z(t)\big)$ converges to zero geometrically.

(d) Let us fix some x^* such that $x^* = Ax^* + b$ and let $z^* = (x^*, \ldots, x^*) \in Z^*$. Let $y(t) = x(t) - x^*$ and $w(t) = z(t) - z^* = \big(y(t), \ldots, y(t - B + 1)\big)$. Let W^* be the set of all vectors of the form $(y^*, \ldots, y^*)$, where y^* satisfies $Ay^* = y^*$. Notice that $w^* \in W^*$ if and only if $z^* + w^* \in Z^*$.

Since $x^* = Ax^* + b$ and $x(t)$ is generated by the asynchronous iteration $x := Ax + b$, it follows that $y(t) = x(t) - x^*$ is generated by the asynchronous iteration $y := Ax + b - x^* = Ax + b - Ax^* - b = A(x - x^*) = Ay$. Now, the distance of $w(t)$ from W^* is the same as the distance $d\big(z(t)\big)$ of $z(t)$ from Z^*. Suppose that the assumptions of Prop. 1.1 hold for $d\big(z(t)\big)$. It follows that the same assumptions hold concerning the distance of $w(t)$ from W^*. Using the result of part (c), we conclude that the distance of $w(t)$ from W^* converges to zero geometrically. This shows that $d\big(z(t)\big)$ also converges to zero geometrically.

7.1.3:

The extension of Prop. 1.1 is as follows. We assume that each function f_i is jointly continuous in x and θ_i and that Assumption 1.1 (partial asynchronism) holds. Suppose also that there exists some positive integer t^* and a continuous function $d : Z \mapsto [0, \infty)$ with the following properties:

(a) For every $z(0) \notin Z^*$, any scenario, and any choice of the parameters $\theta_i(t)$, $i = 1, \ldots, n$, $t = 0, 1, \ldots$, we have $d\big(z(t^*)\big) < d\big(z(0)\big)$.

(b) For every $z(0) \in Z$, any scenario, any $t \geq 0$, and any choice of the parameters $\theta_i(\tau)$, $i = 1, \ldots, n$, $\tau = 0, 1, \ldots$, we have $d\big(z(t+1)\big) \leq d\big(z(t)\big)$.

Then, we will show that $z^* \in Z^*$ for every limit point $z^* \in Z$ of the sequence $\{z(t)\}$.

The proof of this result follows the same lines as the proof of Prop. 1.1. Let θ be a vector with all the parameters $\theta_i(\tau)$, $0 \leq \tau < t^*$, $i = 1, \ldots, n$. These are all the parameters that could influence $z(t^*)$. Notice that the set of all possible vectors θ is a Cartesian product of compact sets and is therefore itself compact.

Let $T^s(z, \theta)$ be the value of $z(t^*)$ under scenario s, when the initial conditions are $z(0) = z$ and for a particular choice for the vector θ of parameters. Similarly with the proof of Prop. 1.1, we define

$$h(z) = \max_{s,\theta} d\big(T^s(z, \theta)\big) - d(z). \tag{1}$$

Notice that T^s is a continuous function of z and θ, for each scenario s, since it is a composition of the continuous functions f_i. As far as the behavior of the algorithm during the time interval $[0, t^*]$ is concerned, the number of different scenarios is finite. Furthermore, θ belongs to a compact set. We are therefore maximizing a continuous function over a compact set and this shows that the maximum in Eq. (1) is attained for any $z \in Z$.

The rest of the proof of Prop. 1.1 remains valid without any modifications, provided that we establish that the function h is continuous, which we now prove. Let $g(z, \theta) = \max_s d\big(T^s(z, \theta)\big) - d(z)$. The function g is continuous since it is the maximum of a finite collection of continuous functions. We now show that $h(z) = \max_\theta g(z, \theta)$ is continuous. Let $\{z^k\}$ be a sequence of elements of Z converging to some $z^* \in Z$. For each k, let θ^k be such that $h(z^k) = g(z^k, \theta^k)$, and let θ be such that $h(z^*) = g(z^*, \theta)$. For each k, we have $h(z^k) = g(z^k, \theta^k) \geq g(z^k, \theta)$. Taking the limit as $k \to \infty$, we obtain

$$\liminf_{k\to\infty} h(z^k) \geq g(z^*, \theta^*) = h(z^*). \tag{2}$$

Let $A = \limsup_{k\to\infty} h(z^k)$. We notice that $A < \infty$ because h is bounded above by 0. We choose a subsequence of $\{z^k\}$ such that $h(z^k)$ converges to A. Since the sequence $\{\theta^k\}$ belongs to a compact set, we can restrict to a further subsequence along which θ^k converges to some θ^*. Taking the limit along the latter subsequence, we see that $h(z^k) = g(z^k, \theta^k)$ converges to $g(z^*, \theta^*)$. Thus, $\limsup_{k\to\infty} h(z^k) = g(z^*, \theta^*)$. Now, for any θ we have $g(z^k, \theta^k) \geq (z^*, \theta)$ and taking the limit along the same subsequence we obtain $g(z^*, \theta^*) \geq (z^*, \theta)$. Since θ was arbitrary, we conclude that

$$h(z^*) = g(z^*, \theta^*) = \limsup_{k\to\infty} h(z^k). \tag{3}$$

Equations (2) and (3) show that $\lim_{k\to\infty} h(z^k) = h(z^*)$ and this establishes the continuity of h.

SECTION 7.2

7.2.1:

(a) The only place in the proof of Prop. 2.3 where Assumption 1.1(c) was used was part (a) of Lemma 2.3 and we only need to repeat the proof of that lemma. Suppose that $x_i(t+1) \neq x_i(t)$. Then $x_i(t) \neq h_i(x^i(t))$. Let A be the interval $[x_i^* - d^*, x_i^* + d^*]$. We have

$$|x_i(t) - x_i^*| \leq D\big(z(t); x^*\big) \leq d\big(z(t)\big) \leq d^*$$

and therefore $x_i(t)$ belongs to A. Furthermore, by Lemma 2.2, $\|x^i(t) - x^*\|_\infty \leq d^*$ and it follows that $\big|h_i\big(x^i(t)\big) - x_i^*\big| \leq d^*$. Thus, $h_i\big(x^i(t)\big) \in A$. We thus see that $x_i(t+1)$ is obtained as a convex combination of two distinct elements of A. Since $0 < \gamma < 1$, $x_i(t+1)$ must lie in the interior of A, which shows that Lemma 2.3(a) is still valid.

(b) We demonstrate the possibility of divergence by means of the following example. Let $n = 1$, and let $h : \Re \to \Re$ be defined by $h(x) = -x$. To simplify notation we write $\tau(t)$ and $x(t)$ instead of $\tau_1^1(t)$ and $x_1(t)$. Suppose that $T^1 = \{0, 1, \ldots\}$, $x(0) = 1$, and that

$$\tau(t) = \max\{s \mid s \leq t \quad \text{and } s \text{ is an integer multiple of } B\}$$

We then have, for $t = 0, 1, \ldots B-1$,

$$x(t+1) = x(t) + \gamma(h(1) - 1) = x(t) - 2\gamma.$$

Therefore, $x(B) = 1 - 2B\gamma$. By a similar argument, we obtain $x(2B) = (1 - 2B\gamma)^2$ and, more generally, $x(kB) = (1 - 2B\gamma)^k$. Clearly, if B is large enough so that $2B\gamma > 1$, the sequence $\{x(t)\}$ is unbounded and divergent.

(c) To prove boundedness of the sequence $\{x(t)\}$, let x^* be some element of X^* and let

$$d = \max_{-B+1 \leq t \leq 0} \|x(t) - x^*\|_\infty.$$

We will show, by induction on t, that $\|x(t) - x^*\|_\infty \leq d$, for all $t \geq -B+1$. This is true by definition for $-B+1 \leq t \leq 0$ and assume that it is true up to some $t \geq 0$. Fix some i. If $t \notin T^i$, then $|x_i(t+1) - x_i^*| = |x_i(t) - x_i^*| \leq d$. If $t \in T^i$, then notice that $\|x^i(t) - x^*\|_\infty \leq d$ and, by the nonexpansive property of h, $\big|h_i\big(x^i(t)\big) - x_i^*\big| \leq d$. Furthermore, $|x_i^i(t) - x_i^*| \leq d$. Thus,

$$|x_i(t+1) - x_i^*| \leq (1-\gamma)|x_i^i(t) - x_i^*| + \gamma\big|h_i\big(x^i(t)\big) - x_i^*\big| \leq (1-\gamma)d + \gamma d = d.$$

Consider now the following example: we have $n = 2$ and $h(x) = Ax$, where A is the matrix

$$A = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}.$$

We are then dealing with the asynchronous linear iteration $x := Cx$ where

$$C = \begin{bmatrix} 1-\gamma & \gamma \\ -\gamma & 1-\gamma \end{bmatrix}.$$

The matrix $|C|$ is stochastic and therefore $\rho(|C|) = 1$. Furthermore, the equation $x = Cx$ has zero as a unique solution. Therefore, 1 is not an eigenvalue of C and the matrix $I - C$ is invertible. We then see that Prop. 3.1 of Section 6.3 (necessary conditions for asynchronous convergence of linear iterations) applies and shows that the algorithm is not guaranteed to converge, even if $B = 2$.

7.2.2:

Let δ be the smallest of the diagonal entries of A. The asynchronous iteration $x := Ax$ can be rewritten as

$$x := x - (1-\delta)\frac{I-A}{1-\delta}x.$$

This iteration is of the form [cf. Eq. (2.6)]

$$x := x - \gamma(Dx - b),$$

where $\gamma = 1 - \delta$, $b = 0$, and $D = (I - A)/(1 - \delta)$. We have $\delta > 0$ since the diagonal entries of A are positive. Furthermore, $\delta < 1$ because otherwise A would be the identity matrix, which is not irreducible. (An exception is if $n = 1$ and $A = 1$; in this case however, the result of the exercise is trivially true.) We thus have $0 < \delta < 1$ and it follows that $0 < \gamma < 1$. We now show that Assumption 2.3 is satisfied by the matrix D and the vector b. Indeed,

$$\Big|1 - \frac{1-a_{ii}}{1-\delta}\Big| + \sum_{j\neq i}\frac{a_{ij}}{1-\delta} = \frac{1}{1-\delta}\Big(a_{ii} - \delta + \sum_{j\neq i} a_{ij}\Big) \leq \frac{1}{1-\delta}(1-\delta) = 1.$$

This verifies Assumption 2.3(a). The equation $Dx = b$ is equivalent to the equation $Ax = x$ and has the zero vector as a solution, which verifies Assumption 2.3(b). Finally, since A is irreducible, so is D because there is a one–to–one correspondence between their nonzero entries. Therefore, Prop. 2.4 applies to the iteration $x := x - \gamma(Dx - b)$ and implies convergence to a fixed point. It follows that the equivalent iteration $x := Ax$ also converges to a fixed point, that is to a vector x^* satisfying $Ax^* = x^*$.

SECTION 7.3

7.3.1:

(a) Let there be three processors ($n = 3$), let

$$A = \begin{bmatrix} 1/2 & 0 & 1/2 \\ 1/2 & 1/2 & 0 \\ 0 & 1/2 & 1/2 \end{bmatrix},$$

and suppose that information is instantaneous; that is, $\tau_j^i(t) = t$, for all $t \in T^i$. Suppose that $x(0) = (0, 0, 1)$, let ϵ_0 be a small positive number and consider the following scenario.
(i) Processor 1 executes a large number t_1 of iterations until $x_1(t_1) \geq 1 - \epsilon_0$. Then $x(t_1) \approx (1, 0, 1)$.
(ii) Processor 3 executes a large number $t_2 - t_1$ of iterations until $x_3(t_2) \leq \epsilon_0$. Then $x(t_2) \approx (1, 0, 0)$.
(iii) Processor 2 executes a large number $t_3 - t_2$ of iterations until $x_2(t_3) \geq 1 - 2\epsilon_0$. Then $x(t_3) \approx (1, 1, 0)$.
Thus, at time t_3, each processor has performed an update and $M(t_3) - m(t_3) \geq 1 - 3\epsilon_0$.

We now repeat the same sequence of events except that we use a smaller value of ϵ; namely, $\epsilon_1 = \epsilon_0/2$. When the new round terminates, at some time t_4, we have

$$M(t_4) - m(t_4) \geq (1 - 3\epsilon_1)(1 - 3\epsilon_0).$$

More generally, at the kth round we use $\epsilon_k = 2^{-k}\epsilon_0$ and, at the time t' that the kth round ends, we have

$$M(t') - m(t') \geq \prod_{i=0}^{k-1}(1 - 3 \cdot 2^{-i}\epsilon_0).$$

As k tends to infinity, t' also tends to infinity. However, the infinite product $\prod_{i=0}^{\infty}(1 - 3 \cdot 2^{-i}\epsilon_0)$ is positive and it follows that $M(t) - m(t)$ does not converge to zero. Thus, the result of Prop. 3.1 is not true. Using the same argument for the iteration $\pi := \pi A$ we see that Prop. 3.2 is not true either.

(b) Let $n = 2$,

$$A = \begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix},$$

and $x_1(0) = 0$, $x_2(0) = 1$. Let

$$x_1(1) = x_2(1) = \frac{1}{2}x_1(0) + \frac{1}{2}x_2(0) = \frac{1}{2}.$$

Then let

$$x_1(2) = x_2(2) = \frac{1}{2}x_1(0) + \frac{1}{2}x_2(1) = \frac{1}{4}.$$

From then on, let

$$x_i(t) = \frac{1}{2}x_1(t-2) + \frac{1}{2}x_2(t-2), \quad t \geq 3.$$

It is easily seen that, for $t \geq 3$,

$$x_i(t) = \begin{cases} 1/2, & \text{if } t \text{ is odd} \\ 1/4, & \text{if } t \text{ is even.} \end{cases}$$

Clearly, this is a partially asynchronous scenario with $B = 2$, for which the agreement algorithm does not converge. Since the matrix A is symmetric, the above example also applies to the iteration $\pi := \pi A$. Thus, Props. 3.1 and 3.2 fail to hold if Assumption 1.1(c) is relaxed.

7.3.2:

(a) We replace parts (a) and (b) of Assumption 1.1 by the assumption that for every $t \geq 0$ and every i, j, such that $(j, i) \in A$, at least one of the elements of the set $\{t, t+1, \ldots, t+B-1\}$ belongs to T_j^i. Part (c) of Assumption 1.1 is not relevant to the algorithm considered here and is not needed. We also replace Assumption 3.1 by the requirement that there exists some processor i such that there exists a positive path from i to every other processor j.

With the above assumption, the proof of Prop. 3.1 goes through with only minor modifications, provided that we redefine α to be equal to

$$\alpha = \min\{a_{ij}, 1 - a_{ij} \mid (j, i) \in A\}.$$

(b) Let $\bar{G}$ be the subgraph of G obtained by removing the processor that has broken down. If there is some processor i and a positive path from i to every other processor j in the new graph $\bar{G}$, then the remaining processors still satisfy the assumptions in part (a) of this exercise and will converge to agreement. The crucial point is that the value possessed by a processor who sends no messages cannot influence the computations of the remaining processors.

(c) With the original algorithm of Eqs. (3.2)–(3.3), if some processor j has broken down then, when processor i executes Eq. (3.3), it will be forced to use an increasingly outdated value of x_j and this can destroy convergence to agreement for the remaining processors. [In contrast to part (b), the value of a processor that has broken down continues to affect the progress of the algorithm.] For a concrete example suppose that

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 1/2 & 0 & 1/2 \\ 0 & 0 & 1 \end{bmatrix}$$

and that $x_1(0) = 1, x_2(0) = x_3(0) = 0$. Here, Assumption 3.1 is satisfied, with $D = \{3\}$. Suppose that communication is instantaneous and that processor 1 breaks down right after communicating its initial value $x_1(0)$. Then,

$$x_3(t) = x_3(0) = 0, \qquad \forall t \geq 0,$$

but

$$x_2(t) = \frac{1}{2}x_1(0) + \frac{1}{2}x_3(0) = \frac{1}{2}, \quad \forall t \geq 0.$$

Thus processors 2 and 3 do not converge to agreement.

The only remedy in a situation where some processor i fails, is for the remaining processors to detect this event and modify the matrix A so that $a_{ji} = 0$, for every $j \neq i$, and so that Assumption 3.1 is still satisfied. Then, the remaining processors will again be able to converge to agreement.

7.3.3:

We fix some positive integer s and some scenario. We define $v(t)$ by letting $v(t) = x(t)$ if $t \leq s$,

$$v_i(t+1) = \sum_{j=1}^{n} a_{ij} v_j(\tau_j^i(t)), \qquad \text{if } \tau \geq s \text{ and } t \in T^i,$$

and $v_i(t+1) = v_i(t)$ if $t \geq s$ and $t \notin T^i$. Let

$$q(t) = \min_i \quad \min_{t-B+1 \leq \tau \leq t} v_i(\tau),$$

$$Q(t) = \max_i \quad \max_{t-B+1 \leq \tau \leq t} v_i(\tau).$$

We notice that $v_i(t)$ is generated by the agreement algorithm and therefore, as shown in the proof of Prop. 3.1, we have [cf. Eq. (3.9)]

$$Q(s+2LB+B) - q(s+2LB+B) \leq \eta\big(Q(s) - q(s)\big),$$

where η is a positive constant smaller than 1.

Since $\epsilon_i(t)$ converges to zero geometrically, there exist some $C > 0, c \in (0,1)$ such that $|\epsilon_i(t)| \leq Cc^t$. By comparing the equations defining $x_i(t)$ and $v_i(t)$, we see that

$$|v_i(t) - x_i(t)| \leq \sum_{\tau=s}^{t-1} |\epsilon_i(\tau)| \leq \sum_{\tau=s}^{\infty} Cc^\tau = \frac{Cc^s}{1-c}, \quad \forall i.$$

It follows that

$$M(s+2LB+B) \leq Q(s+2LB+B) + \frac{Cc^s}{1-c},$$

$$m(s+2LB+B) \geq q(s+2LB+B) - \frac{Cc^s}{1-c},$$

where the functions M and m are defined by Eqs. (3.4)–(3.5). Furthermore, $M(s) = Q(s)$ and $m(s) = q(s)$. Thus,

$$M(s+2LB+B) - m(s+2LB+B) \leq \eta\big(M(s) - m(s)\big) + \delta(s),$$

where $\delta(s) = 2Cc^s/(1-c)$. It is easily seen from the latter inequality that $M(t) - m(t)$ converges geometrically to zero.

Due to the presence of the perturbations $\epsilon_i(t)$, it is not necessarily true that $m_i(t)$ is nondecreasing in t. On the other hand, it is easily shown that

$$m(s+t) \geq m(s) - \sum_{\tau=s}^{s+t-1} |\max_i \epsilon_i(\tau)| \geq m(s) + \frac{Cc^s}{1-c}.$$

Thus,

$$\liminf_{t\to\infty} m(t) = \liminf_{t\to\infty} m(s+t) \geq m(s) + \frac{Cc^s}{1-c}. \tag{1}$$

Since s is arbitrary, we can take the limit superior of both sides to obtain

$$\liminf_{t\to\infty} m(t) \geq m(s).$$

This shows that $m(t)$ is either unbounded or it converges. We show that it cannot be unbounded. Indeed, Eq. (1) shows that $m(t)$ is bounded below. Furthermore, a symmetrical argument shows that $M(t)$ is bounded above, and since $m(t) \leq M(t)$, we see that $m(t)$ is also bounded above. We conclude that $m(t)$ converges to some limit y. Since $M(t) - m(t)$ converges to zero, $M(t)$ also converges to y and this easily implies that $x_i(t)$ also converges to y for each i.

7.3.4:

We renumber the states of the Markov chain to rewrite P in the form

$$P = \begin{bmatrix} P_{11} & P_{12} \\ 0 & P_{22} \end{bmatrix}.$$

Here P_{11} is a $m \times m$ matrix, where m is the number of transient states of the Markov chain. Since P has a single ergodic class, it follows that P_{22} is irreducible.

We decompose accordingly the vector $\pi(t) = \big(\pi^{(1)}(t), \pi^2(t)\big)$. Thus, the asynchronous iteration $\pi := \pi P$ can be rewritten as

$$\pi^{(1)} := \pi^{(1)} P_{11}, \tag{1}$$

$$\pi^{(2)} := \pi^{(1)} P_{12} + \pi^{(2)} P_{22}. \tag{2}$$

It can be seen that $\rho(P_{11}) < 1$. [The proof is identical with the proof that $\rho(\tilde{P}) < 1$, in Prop. 8.4 of Section 2.8.] Thus, under the partial asynchronism assumption, iteration (1) converges geometrically to zero. Let π^* be a positive row vector such that $\pi^* P_{22} = \pi^*$ and whose components add to one. Such a vector exists because P_{22} is irreducible. Consider the change of variables

$$x_i(t) = \frac{\pi_i^{(2)}(t)}{\pi_i^*}.$$

We proceed, as in the proof of Prop. 3.2, to see that $x_i(t)$ is generated according to the perturbed agreement algorithm of Exercise 7.3.3, provided that we let

$$\epsilon_i(t) = \frac{[\pi^{(1)}(t)P_{12}]_i}{\pi_i^*},$$

which has been shown to converge geometrically to zero. We use the result of Exercise 7.3.3 and it follows, as in the proof of Prop. 3.2, that $\pi^{(2)}(t)$ converges geometrically to $c\pi^*$, for some positive constant c. Thus, $\pi(t)$ converges geometrically to a positive multiple of the vector $(0, \pi^*)$. However, $(0, \pi^*)P = (0, \pi^*)$ and $(0, \pi^*)$ is the vector of invariant probabilities corresponding to P.

SECTION 7.4

7.4.1:

(a) Let there be two processors, let $T^i = \{1, , 2, \ldots\}$ for $i = 1, 2$, let $L = 1$, suppose that any load transferred from one processor is immediately delivered to the other, and suppose that information exchanges are instantaneous. Let $x_1(0) = 1, x_2(0) = 0$. Let $\{\epsilon_k\}$ be an increasing sequence of positive numbers whose limit is smaller than $1/2$. Suppose that at some time t_k we have

$$x_1(t_k) \geq 1 - \epsilon_k \quad \text{and} \quad x_2(t_k) \leq \epsilon_k,$$

and let the processors exchange information at that time. Until the next time t_{k+1} that information is exchanged, processor 1 keeps transferring parts of its load to processor 2. In particular, if a long time elapses until information is exchanged again, we will have $x_1(t_{k+1}) \leq \epsilon_{k+1}$, and $x_2(t_{k+1}) \geq 1 - \epsilon_{k+1}$. Since ϵ_k does not converge to $1/2$, we see that the difference $x_1(t) - x_2(t)$ does not converge to zero.

(b) The example here is similar. We have initially $x_2(0) = 0$. As long as the load transferred from processor 1 has not reached its destination, the condition $x_1(t) \geq x_2(t) = 0$ holds. Thus $x_1(t + 1) \leq (1 - \alpha)x_1(t)$, where α is the constant of Assumption 4.2. Eventually, we will have $x_1(t_1 + 1) \leq \epsilon_1$. Suppose that when this happens, all of the load in transit reaches processor 2. We repeat this sequence, with the roles of of the two processors interchanged, to obtain $x_2(t_2 + 1) \leq \epsilon_2$ at some later time t_2. We continue similarly and if the sequence $\{\epsilon_k\}$ has a limit smaller than $1/2$, the difference $x_1(t) - x_2(t)$ does not converge to zero.

(c) Let there be four processors connected as in the figure.

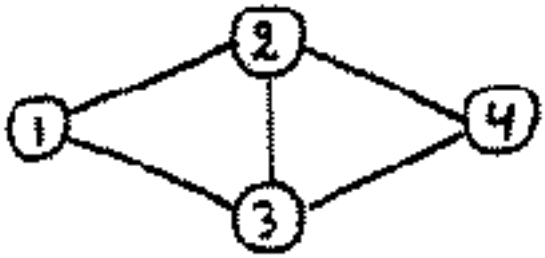

Figure For Exercise 7.4.1.

Let $L = 3$. Initially $x_1(0) = 3, x_2(0) = x_3(0) = x_4(0) = 0$. Suppose that at some time t we have

$$x_1(t) = 1 + \epsilon(t), \quad x_2(t) = x_3(t) = 1 - \frac{\epsilon(t)}{2} \qquad x_4(t) = 0,$$

where $\epsilon(t)$ is positive. Let $t \in T^1$. Processor 1 sends an amount $\big(x_1(t) - x_2(t)\big)/3 = \epsilon(t)/2$ of load to processor 2, which is received immediately. Thus, $x_2(t+1) = 1$. Let $t+1 \in T^2$. Processor 2 instead of sending load to the lightest loaded neighbor, sends an amount $\big(x_2(t+1) - x_3(t+1)\big)/2 = \epsilon(t)/4$ to processor 3 [thus violating Assumption 4.2(a)]. We then have

$$x_1(t+2) = 1 + \frac{\epsilon(t)}{2}, \quad x_2(t+2) = x_3(t+2) = 1 - \frac{\epsilon(t)}{4}, \quad x_4(t) = 0.$$

We now interchange the roles of processors 2 and 3 and follow a similar scenario to obtain

$$x_1(t+4) = 1 + \frac{\epsilon(t)}{4}, \quad x_2(t+4) = x_3(t+4) = 1 - \frac{\epsilon(t)}{8}, \quad x_4(t) = 0.$$

We repeat the above 4–step sequence of events, periodically. All assumptions, except for Assumption 4.2(a) are satisfied but $x_4(t) = 0$ for all t.

(d) Let there be two processors, let $T^1 = T^2$ be the set of all nonnegative integers and let $\tau_j^i(t) = t$ for all t. Under the requirements $x_1(t) \geq x_2(t) + s_{12}(t), \quad x_2(t) \geq x_1(t) + s_{21}(t)$, we could have:

$$x_1(0) = 1, \quad s_{12}(0) = 1, \quad x_1(1) = 0, \quad s_{12}(1) = 0, \quad x_1(2) = 1,$$

$$x_2(1) = 0, \quad s_{21}(0) = 0, \quad x_2(1) = 1, \quad s_{21}(1) = 1, \quad x_2(2) = 0.$$

Repeating periodically, we see that convergence fails to hold.

7.4.2:

Let there be four fully connected processors. Suppose that $L = 3, \quad x_1(0) = x_2(0) = x_3(0) = 1$, and $x_4(0) = 0$. Let $\tau_j^i(t) = t$ for all t. Thus $M(0) = 1$. Suppose that processors $i = 1, 2, 3$, send

at time 0 an amount $s_{i4}(0) = \big(x_i(0) - x_4(0)\big)/2 = 1/2$ of load to processor 4. If these loads are immediately received by processor 4, we will have $x_4(1) = 3/2 > 1$ and $M(1) > M(0)$.

7.4.3:

Follow the hint.

7.4.4:

The proof is by induction in d. The algorithm clearly works if $d = 1$. Suppose it also works for some general d. Consider a $(d+1)$-cube and let H_0, H_1 be the subcubes in which the first bit in the identity of each processor is zero (respectively one). After the first phase, for every processor in H_0 there exists a processor in H_1 (the one obtained by setting the first bit of its identity) who has the same load. It follows that the load in each subcube H_0, H_1 is exactly one half of the total. After stage 1, the subcubes H_0, H_1 do not interact and by the induction hypothesis the load is equalized in each one of them after the additional d phases are over. Since the load in each subcube is equalized and since the total loads in the two subcubes are equal it follows that all processors have the same load.

SECTION 7.5

7.5.1:

The condition $\gamma > 0$ is necessary for convergence even for the case of a synchronous iteration. This is because, for the synchronous iteration, convergence obtains if and only if $\rho(I - \gamma A) < 1$ and, since A is positive definite, this implies that $\gamma > 0$.

We now turn to the necessity of the condition $\gamma \leq c/B$, where c is a constant idenpendent of γ or B. As shown in Example 1.3 of Section 7.1, a necessary condition for the convergence of the partially asynchronous algorithm is $|g(B, \gamma)| < 1$, where

$$g(B,\gamma) = \big(1 - \gamma(1+\epsilon)\big)^B - 2\frac{1 - \big(1 - \gamma(1+\epsilon)\big)^B}{1+\epsilon} = \Big(1 + \frac{2}{1+\epsilon}\Big)\big(1 - \gamma(1+\epsilon)\big)^B - \frac{2}{1+\epsilon}.$$

Let us use the notation $\gamma(B) = \sup\{\gamma > 0 \mid \ |g(B,\gamma)| < 1\}$. (In particular, for any B, the condition $\gamma \leq \gamma(B)$ is necessary for convergence.) Since g is continuous, we have $\big|g\big(B,\gamma(B)\big)\big| \leq 1$ for all B. Recall that $\epsilon < 1$, which implies that $2/(1+\epsilon) > 1$. The condition $\big|g\big(B,\gamma(B)\big)\big| \leq 1$ implies that there exists some $c \in (0,1)$ such that $\big(1-\gamma(B)(1+\epsilon)\big)^B \geq c$ for all B. Thus, $1-\gamma(B)(1+\epsilon) \geq c^{1/B}$ for all B. Taking the limit as $B \to \infty$, the expression $c^{1/B}$ converges to 1, and this implies that $\lim_{B\to\infty} \gamma(B) = 0$. Now, notice that

$$\begin{aligned} c \leq \liminf_{B\to\infty} \big(1-\gamma(B)(1+\epsilon)\big)^B &= \liminf_{B\to\infty} \big(1-\gamma(B)(1+\epsilon)\big)^{B\gamma(B)(1+\epsilon)/\gamma(B)(1+\epsilon)} \\ &= \liminf_{B\to\infty} e^{-B\gamma(B)(1+\epsilon)}. \end{aligned}$$

[We have used here the well known fact $\lim_{x\downarrow 0}(1-x)^{1/x} = e^{-1}$.] We conclude that

$$\limsup_{B\to\infty} B\gamma(B) \leq |\log c|/(1+\epsilon) < \infty.$$

This implies that there exists a constant d such that $B\gamma(B) \leq d$ for all B. Since the condition $\gamma \leq \gamma(B)$ is necessary for convergence, we obtain the necessary condition $\gamma \leq d/B$.

7.5.2:

The only changes needed are the following. In the third line of Eq. (5.8), the term $s_i(t)\nabla_i F\big(x^i(t)\big)$ becomes $s_i(t)'\nabla_i F\big(x^i(t)\big)$ and is bounded above by $\|s_i(t)\|^2/K_3$, because of Assumption 5.5(a). Furthermore, $|s_i(t)|$ should be replaced throughout by $\|s_i(t)\|$. Finally, once Eq. (5.13) is established, we invoke Assumption 5.5(b) [instead of Assumption 5.2(b)] to conclude that $\nabla F\big(x^i(t)\big)$ converges to zero.

SECTION 7.6

7.6.1:

Let t_0 be an integer. Then $\bar{x}_p(t)$ remains constant during the interval (t_0, t_0+1), and so does $\rho_p(t)$. We have,

$$\begin{aligned} \frac{dx_p(t)}{dt} &= \delta\lambda\rho_p(t) - \mu x_p(t) \\ &= \delta\lambda\frac{\bar{x}_p(t_0)}{r^*} - \mu x_p(t) \\ &= \mu\big(\bar{x}_p(t_0) - x_p(t)\big), \qquad t \in (t_0, t_0+1). \end{aligned}$$

We solve this differential equation and we obtain

$$x_p(t_0+1) = \bar{x}_p(t_0) + \big(x_p(t_0) - \bar{x}_p(t_0)\big)e^{-\mu}.$$

This shows that Eq. (6.11) holds with $a_p(t_0) = 1 - e^{-\mu} > 0$, and since $a_p(t_0)$ does not depend on t_0, (6.10) is also valid.

7.6.2:

See [TsB86].

7.6.3:

Omitted.

SECTION 7.8

7.8.1:

We are dealing with the difference equation

$$V(t+1) = \big(1-\gamma(t)\big)^2 V(t) + \sigma^2\gamma^2(t). \tag{1}$$

Since $\sum_{t=1}^{\infty}\gamma^2(t) < \infty$, we obtain $\lim_{t\to\infty}\gamma(t) = 0$. Therefore, without loss of generality, we can assume that $\gamma(t) < 1$ for all t. We then obtain from Eq. (1), $V(t+1) \le V(t) + \sigma^2\gamma^2(t)$, which leads to $V(t) \le V(1) + \sum_{\tau=1}^{\infty}\gamma^2(\tau)$ for all t. Since $\sum_{t=1}^{\infty}\gamma^2(t) < \infty$, we see that there exists some constant A such that $V(t) \le A$ for all t.

Using the bound on $V(t)$, Eq. (1) yields

$$V(t+1) \le V(t) - 2\gamma(t)V(t) + \gamma^2(t)(A+\sigma^2), \tag{2}$$

which leads to

$$0 \le V(t+1) \le V(1) - 2\sum_{\tau=1}^{t}\gamma(\tau)V(\tau) + (A+\sigma^2)\sum_{\tau=1}^{\infty}\gamma^2(\tau). \tag{3}$$

The last term in the right–hand side of inequality (3) is finite. Therefore, $\sum_{\tau=1}^{\infty} \gamma(\tau)V(\tau)$ must also be finite because otherwise the right–hand side of Eq. (3) would be equal to minus infinity. Since $\sum_{t=1}^{\infty} \gamma(t) = \infty$, this implies that $\liminf_{t\to\infty} V(t) = 0$.

Given some $\epsilon > 0$, let us choose some t_0 such that $V(t_0) \leq \epsilon$ and $(A + \sigma^2) \sum_{\tau=t_0}^{\infty} \gamma^2(\tau) \leq \epsilon$. Using Eq. (2), we obtain

$$V(t+1) \leq V(t_0) - 2\sum_{\tau=t_0}^{t} \gamma(\tau)V(\tau) + (A+\sigma^2)\sum_{\tau=t_0}^{\infty} \gamma^2(\tau) \leq \epsilon + \epsilon = 2\epsilon, \qquad \forall t \geq t_0.$$

Thus, $\limsup_{t\to\infty} V(t) \leq \epsilon$. Since this is true for every $\epsilon > 0$, we conclude that $\lim_{t\to\infty} V(t) = 0$.

CHAPTER 8

SECTION 8.2

8.2.1:

Consider the messages sent from some processor i to some processor j. Since the times $\{t_i \mid i \in N\}$ are possibly simultaneous, any message sent at or after time t_i is received after time t_j. Similarly, any message sent at or after time t'_i is received after time t'_j. We conclude that any message sent at or after $\max\{t_i, t'_i\}$ is received after time $\max\{t_j, t'_j\}$. This argument holds for any pair i, j of processors and shows that the times $\big\{ \max\{t_i, t'_i\} \mid i \in N \big\}$ are possibly simultaneous. The argument for $\big\{ \min\{t_i, t'_i\} \mid i \in N \big\}$ is similar.

SECTION 8.3

8.3.1:

We claim that at any time t, at most one of the nodes $i_1, \ldots, i_K$ is a sink. Let $G(t)$ be the directed acyclic graph at stage t and let $\bar{G}(t)$ be the subgraph of $G(t)$ consisting of the nodes $i_1, \ldots, i_K$ and the arcs that connect them. Notice that if a node i_k is a sink in the graph $G(t)$ then it is also a sink in the graph $\bar{G}(t)$. Thus, in order to establish our claim, it is sufficient to show that $\bar{G}(t)$ has exactly one sink at any given time. This is certainly true for $t = 1$, by assumption. We proceed by induction. Assume it is true for some t. If none of the nodes $i_1, \ldots, i_K$ performs an arc reversal at stage t, then $\bar{G}(t+1) = \bar{G}(t)$ and $\bar{G}(t+1)$ has exactly one sink. If on the other hand, one of the nodes $i_1, \ldots, i_K$ performs an arc reversal at time t, it is easily seen that $\bar{G}(t+1)$ also has exactly one sink (see the figure).

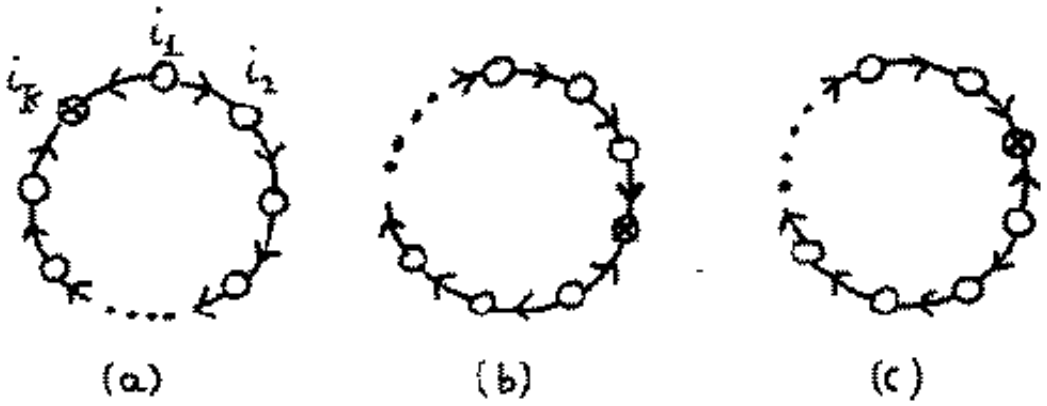

Figure For Exercise 8.3.1. (a) The graph $\bar{G}(1)$. (b) Suppose that $\bar{G}(t)$ has the structure shown and that node i_k performs an arc reversal at stage t. (c) The graph $\bar{G}(t+1)$ has the same structure as $\bar{G}(t)$ and has exactly one sink.

Having established that at most one of the nodes $i_1, \ldots, i_K$ can be a sink at any given time, we obtain $\sum_{k=1}^{K} \big(X_{i_k}(t+1) - X_{i_k}(t)\big) \leq 1$ for all t. This implies that $\sum_{k=1}^{K} X_{i_k}(t) \leq t$. We divide by t, take the limit as $t \to \infty$, and use the fact $\lim_{t\to\infty} \big(X_i(t)/t\big) = M/n$, to obtain $KM/n \leq 1$, or $M \leq n/K$.

8.3.2:

We can assume that the mesh–points are points in d–dimensional space with integer coordinates. We say that a node in the mesh with coordinates $(x_1, \ldots, x_d)$ is black or white depending on whether $x_1 + \cdots + x_d$ is even or odd, respectively. Notice that any arc in the graph G joins nodes with different colors. Let us orient the arcs so that each arc is directed from a black to a white node. Then, all white nodes are sinks and, when an arc reversal is performed, all black nodes become sinks. Thus, at any time, half of the nodes are sinks and $M = n/2$. We argue that this is the best possible. Indeed, no node can perform arc reversals at two consecutive time stages because an arc reversal of its neighbors must occur in between. Thus, $X_i(t) \leq (t/2) + 1$ for every i, from which it follows that $M \leq n/2$.

SECTION 8.4

8.4.1:

Let $M(t)$ be the number of messages and antimessages with timestamp less than or equal to t and let $R(t)$ be the number of rollbacks that reset some τ_i to a value less than or equal to t. We have $M(-1) = 0$ and $R(-1) = 0$. Notice that a rollback can reset some τ_i to a value less than or equal to t, if and only if processor i receives a message or antimessage with timestamp less than or equal to t. Furthermore, distinct rollbacks are triggered by different messages. Thus, $R(t) \leq M(t)$. A processor i can send an antimessage with timestamp less than equal to $t+1$ only after a rollback occurs that resets τ_i to a value less than or equal to t. Thus, the number of such antimessages is bounded by $ntR(t)$. (The factor of nt is present because a rollback by a processor leads, in the worst case, to the cancellation of nt messages.) Furthermore, for any τ, the number of messages from processor i to processor j with timestamp τ, is at most 1 plus the number of antimessages with the same timestamp. (This is because exactly one antimessage has to be sent between two consecutive messages.) Thus, the total number of messages with timestamp less than or equal to $t+1$ is bounded by the number of corresponding antimessages plus $n^2(t+1)$. This shows that $M(t+1) \leq 2ntR(t)+(t+1)n^2 \leq 2ntM(t)+(t+1)n^2$. We conclude that there exists a bound $f(T,n)$ such that $M(T) \leq f(T,n)$.

8.4.2:

(a) The modification is as follows. Whenever a processor executes a simulation step, it can "flag" the particular message that was used in that step. Then, if an antimessage arrives, invalidating an unflagged message, both the message and the antimessage are discarded without any further effects and without a rollback. This modification has the same effect as artificially delaying the processing of the unflagged message until after the corresponding antimessage is received, at which time both of them are discarded. Therefore, the modification cannot affect the correctness of the algorithm.

(b) Suppose that processor i has received a message (t, m, j, i) and later receives another message (t, m', j, i). The second message must have been generated by processor P_j after a rollback that invalidated the first message. Since the invalidation of old messages subsequent to a rollback comes before the reevaluation of the messages, processor P_j must have sent an antimessage $(t, m, j, i, *)$ in between. Assuming that messages and antimessages are received in the order that they are transmitted, then the antimessage $(t, m, j, i, *)$ must have arrived before the new message (t, m', j, i). If we make the further assumption (that was omitted from the statement of the problem) that messages and antimessages are processed by the receiving processor in the order that they are received, it follows that by the time that (t, m', j, i) enters the buffer of messages received and processed, the message (t, m, j, i) will have already been deleted from that buffer.

SECTION 8.5

8.5.1:

Let X_i be the number of arc reversals performed by node i. We argue, by induction on i, that $X_i = i$ for each i. Consider the case $i = 1$. The arc $(1, 0)$ initially points towards node 1 and in the end it points towards node 0. Thus, node 1 performs at least one arc reversal. Subsequent to the first arc reversal by node 1, the arc $(1, 0)$ will point towards node 0 and its orientation can never again change, since node 0 never performs any arc reversals. This shows that node 1 performs exactly one arc reversal. Suppose that $X_k = k$. Consider the arc $(k, k + 1)$. Initially this arc points towards node $k + 1$. Just before the first arc reversal by node k, it must point towards node k. Thus, node $k + 1$ performs at least one arc reversal before the first arc reversal by node k. Furthermore, since neighbors have to take turns in performing arc reversals, node $k + 1$ performs $k - 1$ arc reversals in between the arc reversals of node k. Finally, just after the last arc reversal of node k, the arc $(k, k+1)$ points towards node $k+1$, and when the algorithm terminates this same arc points towards node k. Thus, node $k + 1$ must perform one more arc reversal, subsequent to the last arc reversal by node k. This shows that $X_{k+1} \geq k + 1$. On the other hand, we have $|X_{k+1} - X_k| \leq 1$, since neighbors take turns in performing arc reversals. This shows that $X_{k+1} = k + 1$. The total number of arc reversals is given by $\sum_{k=1}^{n} k = (n + 1)n/2$. For the case $n = 3$, the answer is 6, in agreement with Fig. 8.5.5.

8.5.2:

Suppose that $G = (N, A)$ is acyclic. If the graph (N, A') is not acyclic, it must have a positive cycle that uses arc (i, j). Thus, there exists a positive path from node j to node i in the graph G. Similarly, if the graph (N, A'') is not acyclic, there exists a positive path from node i to node j in the graph G. Thus, if neither (N, A') nor (N, A'') is acyclic, we can join the above described two paths to form a positive cycle that goes through nodes i and j, which would contradict the acyclicity of G.

8.5.3:

(a) Consider three consecutive partial arc reversals by processor i that take place at times t_1, t_2, t_3, respectively. Just after time t_1, the list maintained by processor i is empty. If processor j has not

performed an arc reversal between t_1 and t_2, then at time t_2 the arc (i, j) is not in the list maintained by processor i. Therefore, this arc must be reversed at the second arc reversal by processor i (at time t_2). At time t_3, processor i must be a sink and this implies that processor j must have reversed the direction of arc (i, j) at some time between t_2 and t_3.

(b) Suppose that processor i has performed $X_i(t)$ partial arc reversals, the kth such reversal occuring at time t_k. Processor j must have performed at least one arc reversal during each one of the time intervals (t_1, t_3), (t_3, t_5), ... There exist at least $\big(X_i(t) - 2\big)/2$ such intervals. Therefore, $X_j(t) \geq \big(X_i(t) - 2\big)/2$, or $X_i(t) \leq 2X_j(t) + 2$.

(c) Let $Y_d(t) = \max_i X_i(t)$, where the maximum is taken over all nodes i whose shortest distance from node 0 is at most d. Clearly, $Y_0(t) = 0$ for all t, because the only node at distance 0 from the center is the center itself, which never performs any arc reversals. Using the result of part (b), we obtain

$$Y_{d+1}(t) \leq 2Y_d(t) + 2, \qquad \forall t. \tag{1}$$

Let D be the diameter of the graph, which is also the largest value of interest for the variable d. The difference equation (1) is easily solved to obtain $Y_D(t) \leq 2^{D+1} - 2$. Thus, the total number of arc reversals is bounded by $nY_D(t) \leq n2^{D+1}$, where n is the number of nodes. Since the bound is independent of t, we see that the total number of arc reversals is finite and the algorithm must eventually terminate.

8.5.4:

In between two consecutive arc reversals, the orientation of the arcs remains the same and any message that gets forwarded along outgoing arcs can only travel $n - 1$ arcs, because the graph is acyclic. Thus, the total number of arcs travelled is at most $(n - 1)A$, where A is a bound on the number of arc reversals throughout the algorithm. In particular, we can take $A = nD$ for the full reversal algorithm and (using the result of Exercise 8.5.3) $A = n2^{D+1}$ for the partial reversal algorithm, where D is the diameter of the graph.