Figure 1: Andrei Andreevich Markov: (1856 – 1922)

Stochastic Processes and Markov Chains

Stochastic process— a family of random variables $\{X(t), t \in T\}$.

Each $X(t)$ is a random variable defined on some probability space
$X(t)$ is the value assumed by the random variable at time t.

T can be discrete, e.g., $T = \{0, 1, 2, \ldots\}$

or T can be continuous, e.g., $T = \{t : 0 \leq t \leq +\infty\}$.

The values assumed by the random variables $X(t)$ are called *states*
— and the *state space* can be discrete or continuous.

If *discrete*, the process is referred to as a *chain*
— states are identified with the set $\{0, 1, 2, \ldots\}$ or a subset of it.

A stochastic process may have:
— a discrete state space or a continuous state space,
— may evolve at a discrete set of time points or continuously in time.

Time dependence: — a process whose evolution depends on the time at which it is initiated is said to be *nonstationary*
— $stationary$ when it is invariant under a shift of the time origin.

Mathematically: A stochastic process is stationary if its joint distribution is invariant to shifts in time, i.e., if for any constant α,

$$\text{Prob}\{X(t_1) \leq x_1,\ X(t_2) \leq x_2,\ \ldots,\ X(t_n) \leq x_n\} =$$

$$\text{Prob}\{X(t_1+\alpha) \leq x_1,\ X(t_2+\alpha) \leq x_2,\ \ldots,\ X(t_n+\alpha) \leq x_n\}$$

for all n and all t_i and x_i with $i = 1, 2, \ldots, n$.

Stationarity is not the same as Homogeneity.

In a stationary process, transition probabilities can depend on the elapsed time

— such a stochastic process is said to be $nonhomogeneous$.

When transitions are independent of the elapsed time

— the stochastic process is said to be $homogeneous$.

In either case (homogeneous or nonhomogeneous)
the stochastic process may, or may not, be stationary.

A $Markov\ process$ is a stochastic process whose conditional probability distribution function satisfies the *Markov* or *memoryless* property.

Markov property: the sojourn time in any state is $memoryless$

— at any time t, the remaining time that the chain will spend in its current state is independent of the time already spent in that state.

Discrete-Time Markov Chains: Definitions

The successive observations define the random variables, $X_0, X_1, \ldots, X_n, \ldots$, at time steps $0, 1, \ldots, n, \ldots$, respectively.

Definition A discrete-time Markov chain, $\{X_n,\ n = 0, 1, 2, \ldots\}$, is a stochastic process that satisfies the following *Markov property*:

For all natural numbers n and all states x_n,

$$\begin{aligned} &\text{Prob}\{X_{n+1} = x_{n+1} | X_n = x_n, X_{n-1} = x_{n-1}, \ldots, X_0 = x_0\} \qquad (1) \\ &= \text{Prob}\{X_{n+1} = x_{n+1} | X_n = x_n\}. \end{aligned}$$

The only history of the process relevant to its future evolution is that the Markov chain is in state x_n at time step n.

Simplify by using i, j and k as states rather than using x_i.

Single-step transition probabilities:

$$\text{Prob}\{X_{n+1} = x_{n+1} | X_n = x_n\} \iff \text{Prob}\{X_{n+1} = j | X_n = i\},$$

— conditional probability of making a transition from state $x_n = i$ to state $x_{n+1} = j$ when the time parameter increases from n to $n+1$.

$$p_{ij}(n) = \text{Prob}\{X_{n+1} = j | X_n = i\}. \tag{2}$$

Transition probability matrix:

$$P(n) = \begin{pmatrix} p_{00}(n) & p_{01}(n) & p_{02}(n) & \cdots & p_{0j}(n) & \cdots \\ p_{10}(n) & p_{11}(n) & p_{12}(n) & \cdots & p_{1j}(n) & \cdots \\ p_{20}(n) & p_{21}(n) & p_{22}(n) & \cdots & p_{2j}(n) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ p_{i0}(n) & p_{i1}(n) & p_{i2}(n) & \cdots & p_{ij}(n) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix}.$$

$$0 \leq p_{ij}(n) \leq 1 \quad \text{and, for all } i: \quad \sum_{\text{all } j} p_{ij}(n) = 1.$$

— a *Markov matrix* or *stochastic matrix*.

Homogeneous Markov chain: for all states i and j

$$\text{Prob}\{X_{n+1} = j | X_n = i\} = \text{Prob}\{X_{n+m+1} = j | X_{n+m} = i\}$$

$$\text{for } n = 0, 1, 2, \ldots \quad \text{and} \quad m \geq 0,$$

— can replace $p_{ij}(n)$ with p_{ij}.

Homogeneous Markov chain:

$$p_{ij} = \text{Prob}\{X_1 = j | X_0 = i\} = \text{Prob}\{X_2 = j | X_1 = i\} = \text{Prob}\{X_3 = j | X_2 = i\} = \cdots$$

Non-homogeneous Markov chain:

$$p_{ij}(0) = \text{Prob}\{X_1 = j | X_0 = i\} \neq \text{Prob}\{X_2 = j | X_1 = i\} = p_{ij}(1).$$

The probabilities $p_{ij}(n) = \text{Prob}\{X_{n+1} = j | X_n = i\}$ are independent of n in a homogeneous discrete-time Markov chain so drop the (n).

$$p_{ij} = \text{Prob}\{X_{n+1} = j | X_n = i\}, \quad \text{for all } n = 0, 1, 2, \ldots.$$

The Markov chain begins at some initial time and in some initial state i.

Given some (infinite) set of time steps, it may change state at each of these steps but only at these time steps.

If at time step n it is in state i:
— with probability $p_{ij}(n)$, its next move will be to state j
— with probability $p_{ii}(n)$, it will remain in its current state.

Example: A Social Mobility Model:

Partition population into upper-, middle- and lower-class brackets
— monitor the movement of successive generations.

Discrete-time Markov chain, $\{X_n \; n \geq 0\}$, where X_n gives the class of the n^{th} generation. Markov property?? Possible P:

$$P = \begin{pmatrix} 0.45 & 0.50 & 0.05 \\ 0.15 & 0.65 & 0.20 \\ 0.00 & 0.50 & 0.50 \end{pmatrix}$$

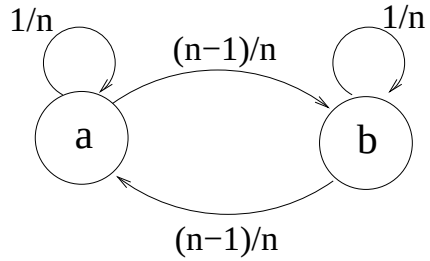
$$Prob\{X_{n+2} = U, X_{n+1} = M | X_n = L\} = p_{LM} p_{MU} = 0.5 \times 0.15 = 0.075,$$

— the probability of the sample path: $L \rightarrow M \rightarrow U$.

Example: A Non-homogeneous Markov chain:

At time step n: probability of no change: $p_{aa}(n) = p_{bb}(n) = 1/n$

— probability that it changes state: $p_{ab}(n) = p_{ba}(n) = (n-1)/n$.



Transition probability matrices:

$$P(1) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad P(2) = \begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix}, \quad P(3) = \begin{pmatrix} 1/3 & 2/3 \\ 2/3 & 1/3 \end{pmatrix},$$

$$P(4) = \begin{pmatrix} 1/4 & 3/4 \\ 3/4 & 1/4 \end{pmatrix}, \quad \cdots \quad P(n) = \begin{pmatrix} 1/n & (n-1)/n \\ (n-1)/n & 1/n \end{pmatrix}, \quad \cdots$$

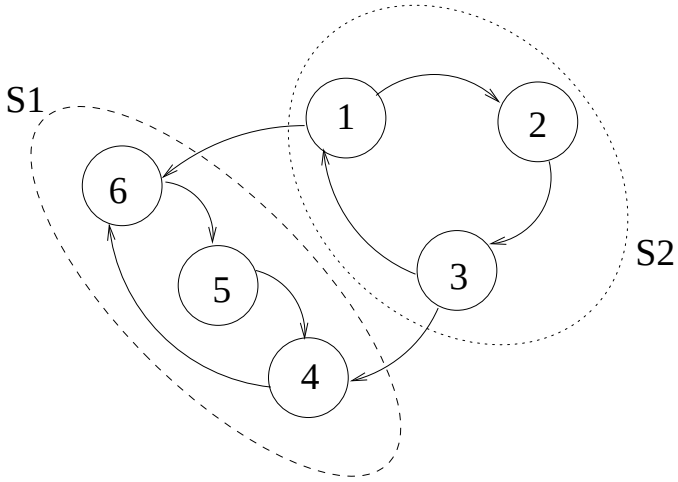
— probability of changing state at each time step increases with time

— probability of remaining in the same state decreases with time.

- But still Markovian

Irreducibility

S_1 is *closed* if no one-step transition is possible from any state in S_1 to any state in S_2.



Any nonempty subset S_1 of S is closed if *no state* in S_1 leads to any state outside S_1 (in any number of steps):

$$p_{ij}^{(n)} = 0, \quad \text{for } i \in S_1,\ j \notin S_1,\ n \geq 1.$$

A set of transient states is open.

A single state that is not absorbing state is an open set.

If a closed subset has only one state, that state is an $absorbing$ state.

NASC for state i to be an absorbing state: $p_{ii} = 1$.

If the set of all states S is closed and does not contain any proper subset that is closed, then the Markov chain is *irreducible*.

If S contains proper subsets that are closed, the chain is $reducible$.

A closed subset of states is an *irreducible subset* if it contains no proper subset that is closed.

Any proper subset of an irreducible subset constitutes a set of states that is open.

$$P = \left(\begin{array}{ccc|ccc} 0 & * & 0 & 0 & 0 & * \\ 0 & 0 & * & 0 & 0 & 0 \\ * & 0 & 0 & * & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & * \\ 0 & 0 & 0 & * & 0 & 0 \\ 0 & 0 & 0 & 0 & * & 0 \end{array}\right) = \left(\begin{array}{cc} D_{11} & U_{12} \\ L_{21} & D_{22} \end{array}\right).$$

Two diagonal blocks: D_{11} and D_{22}
Two off-diagonal blocks: U_{12} and L_{21} — all of size 3×3.

$L_{21} = 0 \implies$ no transition is possible from any state represented by diagonal block D_{22} to any state represented by diagonal block D_{11}.

$U_{12} \neq 0 \implies$ transitions can occur from the states of D_{11} to the states of D_{22}.

Irreducible in terms of the reachability of the states.

State j is *reachable* or *accessible* from state i if there exists a path from state i to state j: Written as $i \rightarrow j$.

There exists a path from state 3 through states 4 and 6 to state 5 (with probability $p_{34}p_{46}p_{65} > 0$) so state 5 is accessible from state 3.

There is no path from state 5 to state 3 so state 3 is not reachable from state 5.

A DTMC is $irreducible$ if every state is reachable from every other state: — there exists an n for which $p_{ij}^{(n)} > 0$ for every pair of states i and j.

Probability Distributions

Distributions defined on the states of a discrete-time Markov chain.

Let $\pi_i(n)$ be the probability that a Markov chain is in state i at step n:

$$\pi_i(n) = \text{Prob}\{X_n = i\}.$$

From the theorem of total probability:

$$\pi_i(n) = \sum_{\text{all } k} \text{Prob}\{X_n = i | X_0 = k\}\pi_k(0) = \sum_{\text{all } k} p_{ki}^{(n)}\pi_k(0), \qquad (3)$$

Row vector: $\pi(n) = (\pi_1(n), \pi_2(n), \ldots, \pi_i(n), \ldots)$ and

$$\pi(n) = \pi(0)P^{(n)} = \pi(0)P^n,$$

— $\pi(n)$ is called the *transient distribution* at step n.

Definition: [Stationary distribution] Let P be the transition probability matrix of a discrete-time Markov chain, and let the vector z, whose elements z_j denote the probability of being in state j, be a probability distribution; i.e.,

$$z_j \in \Re, \quad 0 \leq z_j \leq 1, \quad \text{and} \quad \sum_{\text{all } j} z_j = 1.$$

Then z is said to be a *stationary distribution* if and only if $zP = z$.

Thus $z = zP = zP^2 = \cdots = zP^n = \cdots$.
— if z is chosen as the initial state distribution,
i.e., $\pi_j(0) = z_j$ for all j, then for all n, $\pi_j(n) = z_j$.

Example: Let

$$P = \begin{pmatrix} 0.4 & 0.6 & 0 & 0 \\ 0.6 & 0.4 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0.5 & 0.5 \end{pmatrix}$$

The following probability distributions all satisfy $zP = z$ and hence are all stationary distributions:

$$\begin{aligned} z &= (1/2,\ 1/2,\ 0,\ 0), \\ z &= (0,\ 0,\ 1/2,\ 1/2), \\ z &= (\alpha/2,\ \alpha/2,\ (1-\alpha)/2,\ (1-\alpha)/2), \quad 0 \leq \alpha \leq 1 \end{aligned}$$

The vector $(1,\ 1,\ -1,\ -1)$ is an invariant vector but it is not a stationary distribution.

Many discrete-time Markov chain have a *unique* stationary distribution.

Example: The social mobility model.

$$P = \begin{pmatrix} 0.45 & 0.50 & 0.05 \\ 0.15 & 0.65 & 0.20 \\ 0.00 & 0.50 & 0.50 \end{pmatrix}$$

$zP = z$ gives rise to the homogeneous system of equations $z(P - I) = 0$
— singular coefficient matrix $P - I$, for otherwise $z = 0$
— only two (not three) linearly independent equations

$$\begin{aligned} z_1 &= 0.45z_1 + 0.15z_2 \\ z_2 &= 0.5z_1 + 0.65z_2 + 0.5z_3 \end{aligned}$$

Temporarily setting $z_1 = 1$ gives $z_2 = 0.55/0.15 = 3.666667$.

Given z_1 and z_2, we have $z_3 = 2[-0.5 + 0.35(0.55/0.15)] = 1.566667$.

Now normalize so that $\sum_{\text{all } j} z_j = 1$: $z = (0.1604,\ 0.5882,\ 0.2513)$.

Incorporating a normalization converts the system of equations with singular coefficient matrix $z(P - I) = 0$, into a system of equation with non-singular coefficient matrix and non-zero right-hand side:

$$(z_1, z_2, z_3) \begin{pmatrix} -0.55 & 0.50 & 1.0 \\ 0.15 & -0.35 & 1.0 \\ 0.00 & 0.50 & 1.0 \end{pmatrix} = (0, 0, 1).$$

with a unique, nonzero solution — the unique stationary distribution.

$$zP = (0.1604, 0.5882,\ 0.2513) \begin{pmatrix} 0.45 & 0.50 & 0.05 \\ 0.15 & 0.65 & 0.20 \\ 0.00 & 0.50 & 0.50 \end{pmatrix}$$

$$= (0.1604, 0.5882,\ 0.2513) = z.$$

Approximately 16% of the population is in the upper class bracket, 59% is in the middle class bracket and 25% is in the lower class bracket.

Definition [Limiting distribution]:

Given an initial probability distribution $\pi(0)$, if the limit

$$\lim_{n\to\infty} \pi(n),$$

exists, then this limit is called the *limiting distribution*, and we write

$$\pi = \lim_{n\to\infty} \pi(n).$$

Notice that

$$\pi = \lim_{n\to\infty} \pi(n) = \lim_{n\to\infty} \pi(0)P^{(n)} = \pi(0) \lim_{n\to\infty} P^{(n)}.$$

Example:

$$P = \begin{pmatrix} 0.4 & 0.6 & 0 & 0 \\ 0.6 & 0.4 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0.5 & 0.5 \end{pmatrix}, \qquad \lim_{n\to\infty} P^{(n)} = \begin{pmatrix} 0.5 & 0.5 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0.5 & 0.5 \end{pmatrix}$$

Some limiting distributions:

$$\begin{aligned} (1,0,0,0) \lim_{n\to\infty} P^{(n)} &= (.5,.5,0,0) \\ (0,0,.5,.5) \lim_{n\to\infty} P^{(n)} &= (0,0,.5,.5) \\ (.375,.375,.125,.125) \lim_{n\to\infty} P^{(n)} &= (.25,.25,.25,.25) \\ (\alpha,1-\alpha,0,0) \lim_{n\to\infty} P^{(n)} &= (.5,.5,0,0), \quad \text{for } 0 \le \alpha \le 1 \end{aligned}$$

Example:

$$P = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

The $\lim_{n\to\infty} P^{(n)}$ does not exist and P does not have a limiting distribution.

Successive powers of P alternate:

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \longrightarrow \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \longrightarrow \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \longrightarrow \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \longrightarrow \cdots$$

$$P, \quad P^2, \quad P^3, \quad P^4 = P, \quad P^5 = P^2, \quad P^6 = P^3, \quad P^7 = P, \quad \cdots$$

Definition [Steady State Distribution]

A limiting distribution π is a steady state distribution if it converges, *independently of the initial starting distribution* $\pi(0)$, to a vector whose components are strictly positive (i.e., $\pi_i > 0$ for all states i) and sum to one. If a steady state distribution exists, it is *unique*.

The existence of a steady state distribution implies that both $\pi(n)$ and $P^{(n)}$ converge *independently* of the initial starting distribution $\pi(0)$.

— also called *equilibrium distributions* and *long-run distributions*.

When a unique steady state distribution exists it is also the unique stationary distribution of the chain:

— a steady state distribution is the unique vector π that satisfies:

$$\pi = \pi(0) \lim_{n\to\infty} P^{(n)} = \pi(0) \lim_{n\to\infty} P^{(n+1)} = \left[\pi(0) \lim_{n\to\infty} P^{(n)}\right] P = \pi P,$$

$\pi = \pi P$ and so π is the (unique) stationary probability vector.

The equations $\pi = \pi P$ are called the *global balance equations* — they equate the flow into and out of states.

Write the $i^{(th)}$ equation, $\pi_i = \sum_{all\ j} \pi_j p_{ji}$, as

$$\pi_i(1 - p_{ii}) = \sum_{j,\ i \neq j} \pi_j p_{ji}$$

or

$$\pi_i \sum_{j,\ i \neq j} p_{ij} = \sum_{j,\ i \neq j} \pi_j p_{ji}$$

LHS: the total flow from out of state i into states other than i

RHS: the total flow out of all states $j \neq i$ into state i.

Irreducible, Ergodic Markov Chains

— all the states are positive-recurrent and aperiodic.

The probability distribution $\pi(n)$, as a function of n, always converges to a limiting distribution π, which is independent of $\pi(0)$.
— π is also the stationary distribution of the Markov chain.

From Equation (3): $\pi_j(n+1) = \sum_{\text{all } i} p_{ij}\pi_i(n)$
Limit as $n \to \infty$:

$$\pi_j = \sum_{\text{all } i} p_{ij}\pi_i.$$

Equilibrium probabilities may be uniquely obtained from

$$\pi = \pi P, \quad \text{with} \quad \pi > 0 \quad \text{and} \quad \|\pi\|_1 = 1. \tag{4}$$

Stationary distribution is replicated on each row of $\lim_{n\to\infty} P^n$.

$$\pi_j = \lim_{n\to\infty} p_{ij}^{(n)}, \quad \text{for all } i \text{ and } j.$$

Some performance measurements for irreducible, ergodic Markov chains:

- $\nu_j(\tau)$: average time in state j during observation period of length τ.

$$\nu_j(\tau) = \pi_j \tau.$$

— π_i: long-run proportion of time that the process spends in state i.
Mean number of sunny days/week: 0.48125; rainy days: 5.33750.

- $1/\pi_j$: average number of steps between successive visits to state j.
— average from one sunny day to the next: $1/.06875 = 14.55$.
— average number of days *between* two sunny days is 13.55.

- ν_{ij}: average time spent in state i at steady-state between two successive visits to state j, (the $visit\ ratio$):

$$\nu_{ij} = \pi_i/\pi_j.$$

— the average number of visits to state i between two successive visits to state j.
Mean number of rainy days between two sunny days 11.09.

Continuous-time Markov Chains

$\{X(t),\ t \geq 0\}$ is a continuous-time Markov chain if, for all integers n, and for any sequence $t_0, t_1, \ldots, t_n,\ t_{n+1}$ such that $t_0 < t_1 < \ldots < t_n < t_{n+1}$, we have

$$\begin{aligned}\text{Prob}\{X(t_{n+1}) = x_{n+1} | X(t_n) = x_n, X(t_{n-1}) = x_{n-1}, \ldots, X(t_0) = x_0\} \\ = \text{Prob}\{X(t_{n+1}) = x_{n+1} | X(t_n) = x_n\}.\end{aligned}$$

Transition probabilities Nonhomogeneous CTMC:

$$p_{ij}(s,t) = \text{Prob}\{X(t) = j | X(s) = i\},$$

where $X(t)$ denotes the state of the Markov chain at time $t \geq s$.

Homogeneous CTMC: Transition probabilities depend on $\tau = t - s$.

$$p_{ij}(\tau) = \text{Prob}\{X(s+\tau) = j | X(s) = i\}; \quad \text{and} \quad \sum_{\text{all } j} p_{ij}(\tau) = 1 \ \forall\ \tau.$$

Transition Probabilities and Transition Rates

The *probability* that a transition occurs from a given source state depends not only on the source state itself but also on the length of the interval of observation.

Let $\tau = \Delta t$ be an observation period and $p_{ij}(t, t + \Delta t)$ the probability that a transition occurs from state i to state j in $[t, t + \Delta t)$.
Consider what happens as $\Delta t \to 0$:

$$p_{ij}(t, t + \Delta t) \to 0 \text{ for } i \neq j; \qquad p_{ii}(t, t + \Delta t) \to 1 \text{ as } \Delta t \to 0.$$

As Δt becomes large, the probability of observing a transition increases.

Duration of observation intervals must be sufficiently small that the probability of observing ≥ 2 transitions within this interval is negligible, i.e., the probability of observing multiple transitions is $o(\Delta t)$.

Transition *rates* do not depend on the length of an observation period.

Let $q_{ij}(t)$ be the rate at which transitions occur from state i to state j at time t.

$$q_{ij}(t) = \lim_{\Delta t \to 0} \left\{ \frac{p_{ij}(t, t+\Delta t)}{\Delta t} \right\}, \quad \text{for } i \neq j. \tag{5}$$

It follows that

$$p_{ij}(t, t+\Delta t) = q_{ij}(t)\Delta t + o(\Delta t), \quad \text{for } i \neq j, \tag{6}$$

If homogeneous, then

$$q_{ij} = \lim_{\Delta t \to 0} \left\{ \frac{p_{ij}(\Delta t)}{\Delta t} \right\}, \quad \text{for } i \neq j.$$

From conservation of probability and equation (5),

$$1 - p_{ii}(t, t+\Delta t) = \sum_{j \neq i} p_{ij}(t, t+\Delta t) \quad (7)$$

$$= \sum_{j \neq i} q_{ij}(t)\Delta t + o(\Delta t). \quad (8)$$

Dividing by Δt and taking the limit as $\Delta t \to 0$, we get

$$\lim_{\Delta t \to 0} \left\{ \frac{1 - p_{ii}(t, t+\Delta t)}{\Delta t} \right\} = \lim_{\Delta t \to 0} \left\{ \frac{\sum_{j \neq i} q_{ij}(t)\Delta t + o(\Delta t)}{\Delta t} \right\} = \sum_{j \neq i} q_{ij}(t).$$

Define

$$q_{ii}(t) \equiv -\sum_{j \neq i} q_{ij}(t). \quad (9)$$

If homogeneous, then

$$q_{ii} = \lim_{\Delta t \to 0} \left\{ \frac{p_{ii}(\Delta t) - 1}{\Delta t} \right\}.$$

To summarize:

$$\begin{aligned} p_{ij}(t, t+\Delta t) &= q_{ij}(t)\Delta t + o(\Delta t), \quad \text{for } i \neq j, \\ p_{ii}(t, t+\Delta t) &= 1 + q_{ii}(t)\Delta t + o(\Delta t). \end{aligned}$$

In matrix form:

$$Q(t) = \lim_{\Delta t \to 0} \left\{ \frac{P(t, t+\Delta t) - I}{\Delta t} \right\},$$

where $P(t, t+\Delta t)$ is the transition probability matrix,

— its ij^{th} element is $p_{ij}(t, t+\Delta t)$.

Numerical Solution of Markov Chains

Discrete-time Markov chain with transition probability matrix P:

$$\pi P = \pi \quad \text{with} \quad \pi e = 1. \tag{10}$$

Continuous-time Markov chain with infinitesimal generator matrix Q:

$$\pi Q = 0 \quad \text{with} \quad \pi e = 1. \tag{11}$$

$$\text{From} \quad \pi P = \pi : \quad \pi(P - I) = 0, \quad \text{with} \quad \pi e = 1$$

$(P - I)$ has all the properties of an infinitesimal generator.

$$\text{From} \quad \pi Q = 0 : \quad \pi(Q\Delta t + I) = \pi, \quad \text{with} \quad \pi e = 1 \tag{12}$$

— Δt is sufficiently small that the probability of two transitions taking place in time Δt is negligible, i.e., of order $o(\Delta t)$.

$$\Delta t \leq \frac{1}{\max_i |q_{ii}|}.$$

In this case, the matrix $(Q\Delta t + I)$ is stochastic.

Example: The *power method*

$$Q = \begin{pmatrix} -4 & 4 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ 0 & 2 & -4 & 2 \\ 0 & 0 & 1 & -1 \end{pmatrix}.$$

Set $\Delta t = 1/6$: $\quad Q\Delta t + I =$

$$\begin{pmatrix} 1-4/6 & 4/6 & 0 & 0 \\ 3/6 & 1-6/6 & 3/6 & 0 \\ 0 & 2/6 & 1-4/6 & 2/6 \\ 0 & 0 & 1/6 & 1-1/6 \end{pmatrix} = \begin{pmatrix} 1/3 & 2/3 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 \\ 0 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & 1/6 & 5/6 \end{pmatrix}.$$

$\pi^{(0)} = \{1,\ 0,\ 0,\ 0\}$: — successively compute
$\pi^{(n)} = \pi^{(n-1)}(Q\Delta t + I)$ for $n = 1, 2, \ldots$

$$
\begin{array}{rcllll}
\pi^{(0)} & = & (1.0000 & 0.0000 & 0.0000 & 0.0000) \\
\pi^{(1)} & = & (0.3333 & 0.6667 & 0.0000 & 0.0000) \\
\pi^{(2)} & = & (0.4444 & 0.2222 & 0.3333 & 0.0000) \\
\pi^{(3)} & = & (0.2593 & 0.4074 & 0.2222 & 0.1111) \\
\vdots & & & & \vdots & \\
\pi^{(10)} & = & (0.1579 & 0.1929 & 0.2493 & 0.3999) \\
\vdots & & & & \vdots & \\
\pi^{(25)} & = & (0.1213 & 0.1612 & 0.2403 & 0.4773) \\
\vdots & & & & \vdots & \\
\pi^{(50)} & = & (0.1200 & 0.1600 & 0.2400 & 0.4800)
\end{array}
$$

which is correct to four decimal places.

The stationary distribution of a Markov chain (discrete-time *or* continuous-time) is found by solving either $\pi P = \pi$ or $\pi Q = 0$.

The first is an *eigenvalue/eigenvector* problem — the stationary solution vector π is the left-hand eigenvector corresponding to a unit eigenvalue of the transition probability matrix P.

Second is a *linear equation* problem — π is obtained by solving a homogeneous system of linear equations with singular coefficient matrix Q.

Stochastic Matrices

$P \in \Re^{n \times n}$ is a *stochastic* matrix if:

1. $p_{ij} \geq 0$ for all i and j.
2. $\sum_{\text{all } j} p_{ij} = 1$ for all i.
3. At least one element in each column differs from zero.

Matrices that obey condition (1) are called nonnegative matrices.

Condition (2) implies that a transition is guaranteed to occur from state i to at least one state in the next time period.

Condition (3) specifies there are no *ephemeral* states, i.e., states that could not possibly exist after the first time transition.

Property 1: Every stochastic matrix has an eigenvalue equal to unity.

Corollary: Every infinitesimal generator has at least one zero eigenvalue.

Property 2: The eigenvalues of a stochastic matrix must have modulus less than or equal to 1.

Proof: For any matrix A,

$$\rho(A) \leq \|A\|_\infty = \max_j \left(\sum_{\text{all } k} |a_{jk}| \right).$$

For a stochastic matrix P, $\|P\|_\infty = 1$, and therefore

$$\rho(P) \leq 1.$$

Property 3: The right-hand eigenvector corresponding to a unit eigenvalue $\lambda_1 = 1$ of P is given by $e = (1, 1, \ldots)^T$.

Property 4: The vector π is a stationary probability vector of P iff it is a left-hand eigenvector corresponding to a unit eigenvalue.
Proof: π does not change when post-multiplied by P:

$$\pi = \pi P,$$

Therefore π satisfies the eigenvalue equation

$$\pi P = \lambda_1 \pi \quad \text{for } \lambda_1 = 1.$$

Additional properties for *irreducible* Markov chains:

Property 5: From Perron Frobenius theorem:
The stochastic matrix of an irreducible Markov chain has a simple unit eigenvalue.

Property 6:

$$P(\alpha) = I - \alpha(I - P) \tag{13}$$

where $\alpha \in \Re' \equiv (-\infty, \infty)\backslash\{0\}$ (i.e., the real line with zero deleted).

Then 1 is a simple eigenvalue of every $P(\alpha)$ with a uniquely defined positive left-hand eigenvector of unit 1-norm, i.e., π.

Proof: Spectrum of $P(\alpha)$: $\lambda(\alpha) = 1 - \alpha(1 - \lambda)$. Furthermore

$$x^T P = \lambda x^T \quad \text{if and only if} \quad x^T P(\alpha) = \lambda(\alpha) x^T \quad \text{for all } \alpha.$$

Thus, regardless of whether or not $P(\alpha)$ is a stochastic matrix, $\lambda(\alpha) = 1$ is a simple eigenvalue of $P(\alpha)$ for all α, because $\lambda = 1$ is a simple eigenvalue of P.

Consequently, the entire family $P(\alpha)$ has a unique positive left-hand eigenvector of unit 1-norm associated with $\lambda(\alpha) = 1$, and this eigenvector is precisely the stationary distribution π of P.

Example:

$$P = \begin{pmatrix} .99911 & .00079 & .00010 \\ .00061 & .99929 & .00010 \\ .00006 & .00004 & .99990 \end{pmatrix}. \tag{14}$$

Eigenvalues $1.0,\ .9998$ and $.9985$.

Maximum value of α for which $P(\alpha)$ is stochastic: $\alpha_1 = 1/.00089$.

$$P(\alpha_1) = \begin{pmatrix} .0 & .88764 & .11236 \\ .68539 & .20225 & .11236 \\ .06742 & .04494 & .88764 \end{pmatrix},$$

Its eigenvalues are $1.0;\ .77528$ and $-.68539$.

Another (non-stochastic) choice: $\alpha_2 = 10,000$:

$$P(\alpha_2) = \begin{pmatrix} -7.9 & 7.9 & 1.0 \\ 6.1 & -6.1 & 1.0 \\ 0.6 & 0.4 & 0.0 \end{pmatrix}.$$

Its eigenvalues are $1.0;\ -1.0$ and -14.0.

The left-hand eigenvector corresponding to the unit eigenvalue for all three matrices:

$$\pi = (.22333,\ .27667,\ .50000),$$

— the stationary probability vector of the original stochastic matrix.

The Effect of Discretization

Values of $\Delta t > 0$ for which $P = Q\Delta t + I$ is stochastic.

Example: $q_1, q_2 \geq 0$

$$Q = \begin{pmatrix} -q_1 & q_1 \\ q_2 & -q_2 \end{pmatrix},$$

$$P = Q\Delta t + I = \begin{pmatrix} 1 - q_1\Delta t & q_1\Delta t \\ q_2\Delta t & 1 - q_2\Delta t \end{pmatrix}.$$

Row sums are equal to 1.

$0 \leq q_1\Delta t \leq 1 \;\Rightarrow\; 0 \leq \Delta t \leq q_1^{-1}; \quad 0 \leq q_2\Delta t \leq 1 \;\Rightarrow\; 0 \leq \Delta t \leq q_2^{-1}.$

Assume that $q_1 \geq q_2$. Then $0 \leq \Delta t \leq q_1^{-1}$ satisfies both.

Also, $0 \leq 1 - q_1\Delta t \leq 1$ and $0 \leq 1 - q_2\Delta t \leq 1$ requires that $\Delta t \leq q_1^{-1}$.

Maximum value of Δt for stochastic P: $\Delta t = 1/\max_i |q_i|$.

Similar results hold for general $P = Q\Delta t + I$

(1) The row sums of P are unity for any value of Δt.

(2) Conditions that guarantee the elements of P are in the interval [0,1]:

Let q be the size of the largest off-diagonal element:

$$q = \max_{i,j \;\; i \neq j}(q_{ij}) \quad \text{and} \quad q_{ij} \geq 0, \quad \text{for all } i, j.$$

Then $0 \leq p_{ij} \leq 1$ holds if $0 \leq q_{ij}\Delta t \leq 1$, which is true if $\Delta t \leq q^{-1}$.

For a diagonal element $p_{ii} = q_{ii}\Delta t + 1$:

$$0 \leq q_{ii}\Delta t + 1 \leq 1 \quad \text{or} \quad -1 \leq q_{ii}\Delta t \leq 0.$$

— the right inequality holds for all $\Delta t \geq 0$, since q_{ii} is negative.
— left inequality $q_{ii}\Delta t \geq -1$ is true if $\Delta t \leq -q_{ii}^{-1}$, i.e., if $\Delta t \leq |q_{ii}|^{-1}$.

Thus P is stochastic if $0 \leq \Delta t \leq (\max_i |q_{ii}|)^{-1}$.
(Since the diagonal elements of Q equal the negated sum of the off-diagonal elements in a row, we have $\max_i |q_{ii}| \geq \max_{i \neq j}(q_{ij})$.)

Example: The (2×2) case. Eigenvalues are the roots of $|P - \lambda I| = 0$:

$$\begin{vmatrix} 1 - q_1\Delta t - \lambda & q_1\Delta t \\ q_2\Delta t & 1 - q_2\Delta t - \lambda \end{vmatrix} = 0.$$

Roots: $\lambda_1 = 1$ and $\lambda_2 = 1 - \Delta t(q_1 + q_2)$. As $\Delta t \to 0$, $\lambda_2 \to \lambda_1 = 1$.

The left eigenvector corresponding to λ_1 is independent of Δt:

$$\left(\frac{q_2}{q_1 + q_2}, \quad \frac{q_1}{q_1 + q_2}\right) \begin{pmatrix} 1 - q_1\Delta t & q_1\Delta t \\ q_2\Delta t & 1 - q_2\Delta t \end{pmatrix} = \left(\frac{q_2}{q_1 + q_2}, \quad \frac{q_1}{q_1 + q_2}\right).$$

— π must be independent of Δt.

— Δt only affects the speed of convergence to π.

It is advantageous to choose Δt as large as possible, subject only to the constraint that P be stochastic.

Direct Methods for Stationary Distributions

Iterative versus Direct Solution Methods

Iterative methods begin with an initial approximation.
At each step the approximation becomes closer to the true solution.

Direct methods attempts to go straight to the final solution.
A certain number of well defined steps must be taken, at the end of which the solution has been computed.

Iterative methods are most common for a number of reasons:

(1) The only operation with matrices, are multiplications with one or more vectors, or with preconditioners.

(2) They do not alter the form of the matrix so compact storage schemes can be implemented. Direct methods cause *Fill-in*.

(3) Use may be made of good initial approximations which is especially beneficial when a series of related experiments is conducted.

(4) An iterative process may be halted once a prespecified tolerance criterion has been satisfied. A direct method must continue until the final specified operation has been carried out.

(5) With iterative methods, the matrix is never altered and hence the build-up of rounding error is, to all intents and purposes, non-existent.

Disadvantage of iterative methods:

Often need a long time to converge and this is not known before hand.

Direct methods have an upper bound on the time required which may be determined before the calculation is initiated.

For certain classes of problems, direct methods give more accurate answer in *less time*.

Gaussian Elimination and LU Decompositions

Direct methods are applied to the system of equations

$$\pi Q = 0, \quad \pi \geq 0, \quad \pi e = 1, \tag{15}$$

i.e., a homogeneous system of n linear equations in n unknowns $\pi_i,\ i = 1, 2, \ldots n$.

Gaussian Elimination

Standard form: $Ax = b$. Transposing both sides of $\pi Q = 0$, we get

$$Q^T \pi^T = 0$$

— now fits the standard form with coefficient matrix $A = Q^T$, the unknown vector $x = \pi^T$ and right-hand side $b = 0$.

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n &= b_2 \\
a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \cdots + a_{3n}x_n &= b_3 \\
\vdots \qquad\qquad\qquad\qquad\qquad\quad & \vdots \quad \vdots \\
a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \cdots + a_{nn}x_n &= b_n
\end{aligned}$$

Step 1: use one equation to eliminate one of the unknowns in the other $n-1$ equations — by adding a multiple of one row into the other rows.

Pivotal equation and *pivot*.

The equations that are modified are said to be *reduced*.

The operations of multiplying one row by a scalar and adding or substracting two rows are called *elementary operations*:
— they leave the system of equations invariant.

Use the first equation to eliminate x_1 from each of the other equations.

— a_{21} becomes zero when we multiple the first row by $-a_{21}/a_{11}$ and then add the first row into the second.

The other coefficients in the second row, and the right-hand side, are also affected:

a_{2j} becomes $a'_{2j} = a_{2j} - a_{1j}/a_{11}$ for $j = 2, 3, \ldots, n$
b_2 becomes $b'_2 = b_2 - b_1/a_{11}$.

Similar effects occur in the others rows:
— row 1 is multiplied by $-a_{31}/a_{11}$ and added to row 3 to eliminate a_{31}

To eliminate the coefficient a_{i1} in column i,
— row 1 must be multiplied by $-a_{i1}/a_{11}$ and added into row i.

When completed for all rows $i = 2, 3, \ldots, n$, the first step of Gaussian elimination is over and we move to Step 2.

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\ 0 + a'_{22}x_2 + a'_{23}x_3 + \cdots + a'_{2n}x_n &= b'_2 \\ 0 + a'_{32}x_2 + a'_{33}x_3 + \cdots + a'_{3n}x_n &= b'_3 \\ \vdots \qquad\qquad\qquad\qquad\qquad & \vdots \quad \vdots \\ 0 + a'_{n2}x_2 + a'_{n3}x_3 + \cdots + a'_{nn}x_n &= b'_n \end{aligned} \tag{16}$$

One equation involves all n unknowns, the others involve $n-1$ unknowns and may be treated independently of the first.

System of $n-1$ linear equation in $n-1$ unknowns, which can be solved using Gaussian elimination.

Use one to eliminate an unknown in the $n-2$ others.

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\
0 \quad + a'_{22}x_2 + a'_{23}x_3 + \cdots + a'_{2n}x_n &= b'_2 \\
0 \quad + \quad 0 \quad + a''_{33}x_3 + \cdots + a''_{3n}x_n &= b''_3 \\
\vdots & \quad \vdots \quad \vdots \\
0 \quad + \quad 0 \quad + a''_{n3}x_3 + \cdots + a''_{nn}x_n &= b''_n
\end{aligned} \tag{17}$$

—one equation in n unknowns,

— one equation in $n-1$ unknowns

— $n-2$ equations in $n-2$ unknowns.

Continue until we have 1 equation in 1 unknown, 2 in 2 unknowns, and so on up to n equations in n unknowns.

This completes the *elimination phase* or *reduction phase*.

— now on to the *backsubstitution phase*.

Backsubstitution Phase:

With 1 equation in 1 unknown: $\bar{a}_{nn}x_n = \bar{b}_n \quad \Rightarrow \quad x_n = \bar{b}_n/\bar{a}_{nn}$.

Given x_n, find x_{n-1} from

$$\bar{a}_{n-1,n-1}x_{n-1} + \bar{a}_{n-1,n}x_n = \bar{b}_{n-1}$$

$$x_{n-1} = \frac{\bar{b}_{n-1} - \bar{a}_{n-1,n}x_n}{\bar{a}_{n-1,n-1}}.$$

Continue in this fashion until all the unknowns have been evaluated.

The computationally intensive part is the reduction phase.

The complexity of reduction is of the order $n^3/3$,
whereas the complexity of the backsubstitution phase is n^2.

Example:

$$\begin{aligned} -4.0x_1 + 4.0x_2 + 0.0x_3 + 0.0x_4 &= 0.0 \\ 1.0x_1 - 9.0x_2 + 1.0x_3 + 0.0x_4 &= 0.0 \\ 2.0x_1 + 2.0x_2 - 3.0x_3 + 5.0x_4 &= 0.0 \\ 0.0x_1 + 2.0x_2 + 0.0x_3 + 0.0x_n &= 2.0 \end{aligned}$$

$$\begin{pmatrix} -4.0 & 4.0 & 0.0 & 0.0 \\ 1.0 & -9.0 & 1.0 & 0.0 \\ 2.0 & 2.0 & -3.0 & 5.0 \\ 0.0 & 2.0 & 0.0 & 0.0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0.0 \\ 0.0 \\ 0.0 \\ 2.0 \end{pmatrix}.$$

The reduction consists of 3 steps.

— eliminate the unknown x_1 from rows 2, 3 and 4;

— eliminate x_2 from rows 3 and 4

— eliminate x_3 from row 4.

Reduction Phase:

$$\begin{array}{r|c} \text{Multipliers} & \left(\begin{array}{rrrr} -4.0 & 4.0 & 0.0 & 0.0 \\ \hline 1.0 & -9.0 & 1.0 & 0.0 \\ 2.0 & 2.0 & -3.0 & 5.0 \\ 0.0 & 2.0 & 0.0 & 0.0 \end{array}\right) \\ \hline 0.25 & \\ 0.50 & \\ 0.00 & \end{array}$$

$$\Longrightarrow \left(\begin{array}{r|rrr} -4.0 & 4.0 & 0.0 & 0.0 \\ \hline 0.0 & -8.0 & 1.0 & 0.0 \\ 0.0 & 4.0 & -3.0 & 5.0 \\ 0.0 & 2.0 & 0.0 & 0.0 \end{array}\right) \left(\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \end{array}\right) = \left(\begin{array}{c} 0.0 \\ 0.0 \\ 0.0 \\ 2.0 \end{array}\right)$$

$$\begin{array}{r|c}
\text{Multipliers} & \left(\begin{array}{cc|cc}
-4.0 & 4.0 & 0.0 & 0.0 \\
0.0 & -8.0 & 1.0 & 0.0 \\
\hline
0.0 & 4.0 & -3.0 & 5.0 \\
0.0 & 2.0 & 0.0 & 0.0
\end{array}\right) \\
0.50 & \\
0.25 &
\end{array}$$

$$\Longrightarrow \left(\begin{array}{cc|cc}
-4.0 & 4.0 & 0.00 & 0.0 \\
0.0 & -8.0 & 1.00 & 0.0 \\
\hline
0.0 & 0.0 & -2.50 & 5.0 \\
0.0 & 0.0 & 0.25 & 0.0
\end{array}\right)
\left(\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \end{array}\right) =
\left(\begin{array}{c} 0.0 \\ 0.0 \\ 0.0 \\ 2.0 \end{array}\right)$$

$$\begin{array}{r|c}
\text{Multipliers} & \left(\begin{array}{cc|cc}
-4.0 & 4.0 & 0.00 & 0.0 \\
0.0 & -8.0 & 1.00 & 0.0 \\
0.0 & 0.0 & -2.50 & 5.0 \\
\hline
0.0 & 0.0 & 0.25 & 0.0
\end{array}\right) \\
0.10 &
\end{array}$$

$$\Longrightarrow \left(\begin{array}{ccc|c}
-4.0 & 4.0 & 0.0 & 0.0 \\
0.0 & -8.0 & 1.0 & 0.0 \\
0.0 & 0.0 & -2.5 & 5.0 \\
\hline
0.0 & 0.0 & 0.0 & 0.5
\end{array}\right)
\left(\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \end{array}\right) =
\left(\begin{array}{c} 0.0 \\ 0.0 \\ 0.0 \\ 2.0 \end{array}\right)$$

At the end of these three steps:

$$\left(\begin{array}{rrrr} -4.0 & 4.0 & 0.0 & 0.0 \\ 0.0 & -8.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & -2.5 & 5.0 \\ 0.0 & 0.0 & 0.0 & 0.5 \end{array}\right) \left(\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \end{array}\right) = \left(\begin{array}{c} 0.0 \\ 0.0 \\ 0.0 \\ 2.0 \end{array}\right).$$

Backsubstitution Phase:

$$\begin{array}{lrcl} \text{Equation } 4: & 0.5\,x_4 = 2 & \Longrightarrow & x_4 = 4 \\ \text{Equation } 3: & -2.5\,x_3 + 5 \times 4 = 0 & \Longrightarrow & x_3 = 8 \\ \text{Equation } 2: & -8\,x_2 + 1 \times 8 = 0 & \Longrightarrow & x_2 = 1 \\ \text{Equation } 1: & -4\,x_1 + 4 \times 1 = 0 & \Longrightarrow & x_1 = 1 \end{array}$$

Therefore the complete solution is $x^T = (1,\ 1,\ 8,\ 4)$.

Observe in this example, that the multipliers do not exceed 1.

LU Decompositions

Example: cont.

$$L = \begin{pmatrix} 1.00 & 0.00 & 0.0 & 0.0 \\ -0.25 & 1.00 & 0.0 & 0.0 \\ -0.50 & -0.50 & 1.0 & 0.0 \\ 0.00 & -0.25 & -0.1 & 1.0 \end{pmatrix}, \quad U = \begin{pmatrix} -4.0 & 4.0 & 0.0 & 0.0 \\ 0.0 & -8.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & -2.5 & 5.0 \\ 0.0 & 0.0 & 0.0 & 0.5 \end{pmatrix}.$$

L lower triangular with diagonal elements equal to 1 and subdiagonal elements equal to the negated multipliers:
U is the reduced matrix: Then $A = LU$.

$$A = LU = \begin{pmatrix} -4.0 & 4.0 & 0.0 & 0.0 \\ 1.0 & -9.0 & 1.0 & 0.0 \\ 2.0 & 2.0 & -3.0 & 5.0 \\ 0.0 & 2.0 & 0.0 & 0.0 \end{pmatrix}.$$

An *LU-decomposition* or *LU-factorization* of A.

Given $Ax = b$ and $A = LU$, x is found by means of a forward substitution step on L, followed by a backward substitution step on U.

$$Ax = LUx = b.$$

Let $z = Ux$. Then the solution x is obtained in two easy steps:

$$\begin{aligned} \text{Forward substitution}: &\ \text{Solve} \quad Lz = b \quad \text{for } z \\ \text{Backward substitution}: &\ \text{Solve} \quad Ux = z \quad \text{for } x \end{aligned}$$

Example: Solving $Lz = b$ (by forward substitution) for z:

$$\begin{pmatrix} 1.00 & 0.00 & 0.0 & 0.0 \\ -0.25 & 1.00 & 0.0 & 0.0 \\ -0.50 & -0.50 & 1.0 & 0.0 \\ 0.00 & -0.25 & -0.1 & 1.0 \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 2 \end{pmatrix}$$

— successively gives $z_1 = 0, \quad z_2 = 0, \quad z_3 = 0$ and $z_4 = 2$.

Now solving $Ux = z$ (by backward substitution) for x:

$$\begin{pmatrix} -4.0 & 4.0 & 0.0 & 0.0 \\ 0.0 & -8.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & -2.5 & 5.0 \\ 0.0 & 0.0 & 0.0 & 0.5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 2 \end{pmatrix}$$

— successively gives $x_4 = 4, \quad x_3 = 8, \quad x_2 = 1$ and $x_1 = 1$.

An LU decomposition, when it exists, is not unique.

The *Doolittle decomposition* is characterized by ones along the diagonal of L.

The *Crout decomposition* assigns the value of one to each of the diagonal elements of U.

Application of GE and LU Decompositions to Markov Chains

The system of equations, $\pi Q = 0$, is homogeneous and the coefficient matrix is singular.

Example:

$$\begin{pmatrix} \pi_1 & \pi_2 & \pi_3 & \pi_4 \end{pmatrix} \begin{pmatrix} -4.0 & 1.0 & 2.0 & 1.0 \\ 4.0 & -9.0 & 2.0 & 3.0 \\ 0.0 & 1.0 & -3.0 & 2.0 \\ 0.0 & 0.0 & 5.0 & -5.0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix}$$

In standard Gaussian elimination form:

$$\begin{pmatrix} -4.0 & 4.0 & 0.0 & 0.0 \\ 1.0 & -9.0 & 1.0 & 0.0 \\ 2.0 & 2.0 & -3.0 & 5.0 \\ 1.0 & 3.0 & 2.0 & -5.0 \end{pmatrix} \begin{pmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \\ \pi_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

The first three equations are the same as the first three of last example.

Reduction Phase:

$$\begin{array}{r|cccc} \text{Multipliers} & -4.0 & 4.0 & 0.0 & 0.0 \\ \hline 0.25 & 1.0 & -9.0 & 1.0 & 0.0 \\ 0.50 & 2.0 & 2.0 & -3.0 & 5.0 \\ 0.25 & 1.0 & 3.0 & 2.0 & -5.0 \end{array}$$

$$\Longrightarrow \left(\begin{array}{c|ccc} -4.0 & 4.0 & 0.0 & 0.0 \\ \hline 0.0 & -8.0 & 1.0 & 0.0 \\ 0.0 & 4.0 & -3.0 & 5.0 \\ 0.0 & 4.0 & 2.0 & -5.0 \end{array}\right) \begin{pmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \\ \pi_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$
\begin{array}{r|c}
\text{Multipliers} & \left(\begin{array}{cc|cc}
-4.0 & 4.0 & 0.0 & 0.0 \\
0.0 & -8.0 & 1.0 & 0.0 \\
\hline
0.0 & 4.0 & -3.0 & 5.0 \\
0.0 & 4.0 & 2.0 & -5.0
\end{array}\right) \\
0.50 & \\
0.50 &
\end{array}
$$

$$
\Longrightarrow \left(\begin{array}{cc|cc}
-4.0 & 4.0 & 0.0 & 0.0 \\
0.0 & -8.0 & 1.0 & 0.0 \\
\hline
0.0 & 0.0 & -2.5 & 5.0 \\
0.0 & 0.0 & 2.5 & -5.0
\end{array}\right)
\left(\begin{array}{c} \pi_1 \\ \pi_2 \\ \pi_3 \\ \pi_4 \end{array}\right) =
\left(\begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \end{array}\right)
$$

$$
\begin{array}{r|c}
\text{Multipliers} & \left(\begin{array}{cc|cc}
-4.0 & 4.0 & 0.0 & 0.0 \\
0.0 & -8.0 & 1.0 & 0.0 \\
0.0 & 0.0 & -2.5 & 5.0 \\
\hline
0.0 & 0.0 & 2.5 & -5.0
\end{array}\right) \\
1.0 &
\end{array}
$$

$$
\Longrightarrow \left(\begin{array}{ccc|c}
-4.0 & 4.0 & 0.0 & 0.0 \\
0.0 & -8.0 & 1.0 & 0.0 \\
0.0 & 0.0 & -2.5 & 5.0 \\
\hline
0.0 & 0.0 & 0.0 & 0.0
\end{array}\right)
\left(\begin{array}{c} \pi_1 \\ \pi_2 \\ \pi_3 \\ \pi_4 \end{array}\right) =
\left(\begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \end{array}\right)
$$

At the end of these three steps:

$$\begin{pmatrix} -4.0 & 4.0 & 0.0 & 0.0 \\ 0.0 & -8.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & -2.5 & 5.0 \\ \underline{0.0} & \underline{0.0} & \underline{0.0} & \underline{0.0} \end{pmatrix} \begin{pmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \\ \pi_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

Backsubstitution phase:

Last row contains all zeros so we cannot compute π_4.

The coefficient matrix is singular

$\Longrightarrow$ can only compute the solution to a multiplicative constant.

Set $\pi_4 = 1$

— compute other π_i in terms of this value.

$$\begin{aligned}
&\text{Equation } 4: && && \pi_4 = 1 \\
&\text{Equation } 3: && -2.5\,\pi_3 + 5 \times 1 = 0 && \Longrightarrow \quad \pi_3 = 2 \\
&\text{Equation } 2: && -8\,\pi_2 + 1 \times 2 = 0 && \Longrightarrow \quad \pi_2 = 0.25 \\
&\text{Equation } 1: && -4\,\pi_1 + 4 \times 0.25 = 0 && \Longrightarrow \quad \pi_1 = 0.25
\end{aligned}$$

Computed solution: $\pi^T = (1/4,\ 1/4,\ 2,\ 1)$

Sum of its elements does not add to 1:
— stationary distribution vector is obtained after normalization,
divide each element of this vector by the sum of its components.

$$\|\pi\|_1 = |\pi_1| + |\pi_2| + |\pi_3| + |\pi_4| = 3.5,$$

The stationary distribution vector, the normalized computed solution, is

$$\pi = \frac{2}{7}\,(1/4,\ 1/4,\ 2,\ 1)\,.$$

$\pi Q = 0$ does not tell the whole story.

We also have $\pi e = 1$,

— so substitute this equation for any equation in $\pi Q = 0$.

The coefficient matrix becomes nonsingular, the right-hand side becomes nonzero and a unique solution, π, exists.

In the original example, the last equation was replaced with

$$0x_1 + 2x_2 + 0x_3 + 0x_4 = 2$$

— solution was normalized to make the second component equal to one.

In an irreducible Markov chain, it matters little what is used to substitute for the last row of zeros, as long as the final vector is normalized so that its 1-norm is equal to one.

— setting the last component equal to one requires the least amount of computation. Alternative approaches are possible.

Gaussian elimination with $A = Q^T$

— the element a_{ij} is the rate of transition from state j into state i.

ALGORITHM 10.1: GAUSSIAN ELIMINATION FOR CTMC:

1. The reduction step:

   For $i = 1, 2, \ldots, n-1$:

   $a_{ji} = -a_{ji}/a_{ii}$ for all $j > i$ % multiplier for row j

   $a_{jk} = a_{jk} + a_{ji}a_{ik}$ for all $j, k > i$ % reduce using row i

2. The back-substitution step:

   $x_n = 1$ % set last component equal to 1

   For $i = n-1, n-2, \ldots, 1$:

   $x_i = -[\sum_{j=i+1}^{n} a_{ij}x_j]/a_{ii}$ % back-substitute to get x_i

3. The final normalization step:

   $norm = \sum_{j=1}^{n} x_j$ % sum components

   For $i = 1, 2, \ldots, n$:

   $\pi_i = x_i/norm$ % component i of stationary

   % probability vector

Scaled Gaussian Elimination

Each equation of $Q^T\pi = 0$ is scaled so that the element on each diagonal is equal to -1.

Example: The two sets of equations

$$\begin{pmatrix} -4.0 & 4.0 & 0.0 & 0.0 \\ 1.0 & -9.0 & 1.0 & 0.0 \\ 2.0 & 2.0 & -3.0 & 5.0 \\ 1.0 & 3.0 & 2.0 & -5.0 \end{pmatrix} \begin{pmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \\ \pi_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

and

$$\begin{pmatrix} -1 & 1 & 0 & 0 \\ 1/9 & -1 & 1/9 & 0 \\ 2/3 & 2/3 & -1 & 5/3 \\ 1/5 & 3/5 & 2/5 & -1 \end{pmatrix} \begin{pmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \\ \pi_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

are identical from an equation solving point of view.

Simplifying the back-substitution phase and facilitates coding:

ALGORITHM 10.1: SCALED GAUSSIAN ELIMINATION FOR CTMC

1. The reduction step:

   For $i = 1, 2, \ldots, n-1$:
   $a_{ik} = -a_{ik}/a_{ii}$ for all $k > i$ % scale row i
   $a_{jk} = a_{jk} + a_{ji}a_{ik}$ for all $j, k > i$ % reduce using row i

2. The back-substitution step:

   $x_n = 1$ % set last component equal to 1
   For $i = n-1, n-2, \ldots, 1$:
   $x_i = \sum_{j=i+1}^{n} a_{ij}x_j$ % back-substitute to get x_i

3. The final normalization step:

   $norm = \sum_{j=1}^{n} x_j$ % sum components
   For $i = 1, 2, \ldots, n$:
   $\pi_i = x_i/norm$ % component i of stationary
   % probability vector

During the reduction, no attempt is made to actually set the diagonal elements to -1: it suffices to realize that this must be the case.

Element below the diagonal are not explicitly to zero.

At the end of the reduction step, the elements above the diagonal of A contain that portion of the upper triangular matrix U needed for the back-substitution step.

Example:

$$A = \begin{pmatrix} -4.0 & 4.0 & 0.0 & 0.0 \\ 1.0 & -9.0 & 1.0 & 0.0 \\ 2.0 & 2.0 & -3.0 & 5.0 \\ 1.0 & 3.0 & 2.0 & -5.0 \end{pmatrix},$$

Matrices obtained at different steps of the reduction phase:

$$A_1 = \begin{pmatrix} -4.0 & 1.0 & 0.0 & 0.0 \\ 1.0 & -8.0 & 1.0 & 0.0 \\ 2.0 & 4.0 & -3.0 & 5.0 \\ 1.0 & 4.0 & 2.0 & -5.0 \end{pmatrix}, \quad A_2 = \begin{pmatrix} -4.0 & 1.0 & 0.0 & 0.0 \\ 1.0 & -8.0 & 0.125 & 0.0 \\ 2.0 & 4.0 & -2.5 & 5.0 \\ 1.0 & 4.0 & 2.5 & -5.0 \end{pmatrix},$$

$$A_3 = \begin{pmatrix} -4.0 & 1.0 & 0.0 & 0.0 \\ 1.0 & -8.0 & 0.125 & 0.0 \\ 2.0 & 4.0 & -2.5 & 2.0 \\ 1.0 & 4.0 & 2.5 & 0.0 \end{pmatrix}.$$

In A_3, only the elements above the diagonal contribute.
Diagonal elements are taken equal to -1.

$$U = \begin{pmatrix} -1.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & -1.0 & 0.125 & 0.0 \\ 0.0 & 0.0 & -1.0 & 2.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \end{pmatrix}$$

Back-substitution using $x_4 = 1$ (+ normalization) gives correct result.

LU decomposition for Markov chains:

$$Q^T x = (LU)x = 0.$$

Set $Ux = z$ and solve $Lz = 0.$ for z

But L is nonsingular $\Rightarrow z = 0.$

Back substitution on $Ux = z = 0$ when $u_{nn} = 0.$

Let $x_n = \eta$ — other elements are found in terms of η.

$x_i = c_i\eta$ for some constants $c_i, i = 1, 2, ..., n$, and $c_n = 1.$

Normalizing yields the desired unique stationary probability vector, π.

From Algorithm 10.1, once A has been found:
— elements on and above the diagonal of A define U,
— elements below the diagonal define L,
— the diagonal elements of L are equal to one.

Example:

$$Q^T = \begin{pmatrix} -4.0 & 4.0 & 0.0 & 0.0 \\ 1.0 & -9.0 & 1.0 & 0.0 \\ 2.0 & 2.0 & -3.0 & 5.0 \\ 1.0 & 3.0 & 2.0 & -5.0 \end{pmatrix} = LU$$

$$L = \begin{pmatrix} 1.00 & 0.0 & 0.0 & 0.0 \\ -0.25 & 1.0 & 0.0 & 0.0 \\ -0.50 & -0.5 & 1.0 & 0.0 \\ -0.25 & -0.5 & -1.0 & 1.0 \end{pmatrix}, \quad U = \begin{pmatrix} -4.0 & 4.0 & 0.0 & 0.0 \\ 0.0 & -8.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & -2.5 & 5.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \end{pmatrix}$$

— U is just that given by Gaussian elimination,

— L is the identity matrix with the negated multipliers below the diagonal.

Forward substitution on $Lz = 0$ gives $z_1 = 0,\ z_2 = 0,\ z_3 = 0$ and $z_4 = 0$.

Back-substitution to compute x from $Ux = z = 0$ is same as previously.

Other considerations

Reducible Markov chains.

Given k irreducible closed communicating classes, all k should independently as separate irreducible Markov chains.

Coefficient matrix has k zero eigenvalues.

Data structure to store transition matrices

Direct methods can be applied when
— transition matrices are small, (order of a few hundred)
— transition matrices are *banded*.

Not recommended when the transition matrix is large and *not* banded, due to the amount of fill-in that can occur.

Alternative when coefficient matrix is generated row by row:

— generate row 1 and store it in a compacted form.
— generate row 2 and eliminate the element in position (2,1).
— now compact row 2 and store it.
and so on.

When row i is generated, rows 1 through $(i-1)$ have been derived, reduced to upper triangular form, compacted and stored.

The first $(i-1)$ rows may therefore be used to eliminate all nonzero elements in row i from column positions $(i,1)$ through $(i,i-1)$.

Advantage: — once a row has been generated in this fashion, no more fill-in will occur into this row.

Use an array to hold temporarily a single unreduced row
— perform reduction in this array — then compact.

Not appropriate for solving general systems of linear equations.

The GTH Advantage

— the diagonal elements are obtained by summing off-diagonal elements.

At the end of each reduction step, the unreduced portion of the matrix is the transpose of a transition rate matrix in its own right
— probabilistic interpretation based on the restriction of the Markov chain to a reduced set of states.

Diagonal elements are formed by adding off-diagonal elements and placing a minus sign in front of this sum instead of performing a single subtraction.

If scaling is also introduced the need to actually form the diagonal elements disappears (all are taken to be equal to -1).

In this case, the scale factor is obtained by taking the reciprocal of the sum of off-diagonal elements.

Example, cont. Previously, Gaussian elimination with scaling:

$$\left(\begin{array}{|cccc} \mathbf{-4.0} & 4.0 & 0.0 & 0.0 \\ \mathbf{1.0} & -9.0 & 1.0 & 0.0 \\ \mathbf{2.0} & 2.0 & -3.0 & 5.0 \\ \mathbf{1.0} & 3.0 & 2.0 & -5.0 \end{array}\right) \longrightarrow \left(\begin{array}{c|ccc} -4.0 & 1.0 & 0.0 & 0.0 \\ \hline 1.0 & \mathbf{-8.0} & 1.0 & 0.0 \\ 2.0 & \mathbf{4.0} & -3.0 & 5.0 \\ 1.0 & \mathbf{4.0} & 2.0 & -5.0 \end{array}\right)$$

$$\longrightarrow \left(\begin{array}{cc|cc} -4.0 & 1.0 & 0.0 & 0.0 \\ 1.0 & -8.0 & 0.125 & 0.0 \\ \hline 2.0 & 4.0 & \mathbf{-2.5} & 5.0 \\ 1.0 & 4.0 & \mathbf{2.5} & -5.0 \end{array}\right) \longrightarrow \left(\begin{array}{ccc|c} -4.0 & 1.0 & 0.0 & 0.0 \\ 1.0 & -8.0 & 0.125 & 0.0 \\ 2.0 & 4.0 & -2.5 & 2.0 \\ \hline 1.0 & 4.0 & 2.5 & \mathbf{0.0} \end{array}\right).$$

Each lower right-hand block is a transposed transition rate matrix.

Reduction step 1 to eliminate a_{21}, a_{31} and a_{41}:

— scale off-diagonal elements in row 1 by dividing by the sum $a_{21} + a_{31} + a_{41} = 1 + 2 + 1$, then eliminate a_{21}, a_{31} and a_{41}.

Should put $-9 + 1$ into position a_{22}, but ignore.

— Diagonal elements are left unaltered.

$$\left(\begin{array}{r|rrr} \mathbf{-4.0} & 4.0 & 0.0 & 0.0 \\ \mathbf{1.0} & -9.0 & 1.0 & 0.0 \\ \mathbf{2.0} & 2.0 & -3.0 & 5.0 \\ \mathbf{1.0} & 3.0 & 2.0 & -5.0 \end{array}\right) \quad \text{scale} \longrightarrow \quad \left(\begin{array}{r|rrr} \mathbf{-4.0} & 1.0 & 0.0 & 0.0 \\ \hline \mathbf{1.0} & -9.0 & 1.0 & 0.0 \\ \mathbf{2.0} & 2.0 & -3.0 & 5.0 \\ \mathbf{1.0} & 3.0 & 2.0 & -5.0 \end{array}\right)$$

$$\text{reduce} \longrightarrow \left(\begin{array}{r|r|rr} -4.0 & 1.0 & 0.0 & 0.0 \\ \hline 1.0 & \mathbf{-9.0} & 1.0 & 0.0 \\ 2.0 & \mathbf{4.0} & -3.0 & 5.0 \\ 1.0 & \mathbf{4.0} & 2.0 & -5.0 \end{array}\right) \quad \text{scale} \longrightarrow \quad \left(\begin{array}{r|r|rr} -4.0 & 1.0 & 0.0 & 0.0 \\ 1.0 & \mathbf{-9.0} & 0.125 & 0.0 \\ \hline 2.0 & \mathbf{4.0} & -3.0 & 5.0 \\ 1.0 & \mathbf{4.0} & 2.0 & -5.0 \end{array}\right)$$

$$\text{reduce} \longrightarrow \left(\begin{array}{rr|r|r} -4.0 & 1.0 & 0.0 & 0.0 \\ 1.0 & -9.0 & 0.125 & 0.0 \\ \hline 2.0 & 4.0 & \mathbf{-3.0} & 5.0 \\ 1.0 & 4.0 & \mathbf{2.5} & -5.0 \end{array}\right) \quad \text{scale} \longrightarrow \quad \left(\begin{array}{rr|r|r} -4.0 & 1.0 & 0.0 & 0.0 \\ 1.0 & -9.0 & 0.125 & 0.0 \\ 2.0 & 4.0 & -3.0 & 2.0 \\ \hline 1.0 & 4.0 & 2.5 & \mathbf{-5.0} \end{array}\right)$$

Algorithm 10.2: GTH for Continuous-Time Markov Chains with $A = Q^T$

1. The reduction step:

   For $i = 1, 2, \ldots, n-1$:
   $a_{ik} = a_{ik} / \sum_{j=i+1}^{n} a_{ji}$ for all $k > i$ % scale row i
   $a_{jk} = a_{jk} + a_{ji} a_{ik}$ for all $j, k > i,\ k \neq j$ % reduce using row i

2. The back-substitution step:

   $x_n = 1$ % set last component equal to 1
   For $i = n-1, n-2, \ldots, 1$:
   $x_i = \sum_{j=i+1}^{n} a_{ij} x_j$ % back-substitute to get x_i

3. The final normalization step:

   $norm = \sum_{j=1}^{n} x_j$ % sum components
   For $i = 1, 2, \ldots, n$:
   $\pi_i = x_i / norm$ % component i of stationary
   % probability vector

Comparing with scaled Gaussian elimination:

— only the first step has changed,

— and within that step only in the computation of the scale factor.

Requires more numerical operations than the standard implementation

— is offset by a gain in precision when Q is ill-conditioned.

Implementing GTH in certain compacted representations is not so simple.

Matlab code for Gaussian Elimination

```
function [pi] = GE(Q)
A = Q';   n = size(A);
for i=1:n-1
    for j=i+1:n
        A(j,i) = -A(j,i)/A(i,i);
    end
    for j=i+1:n
        for k=i+1:n
            A(j,k) = A(j,k) + A(j,i)*A(i,k);
        end
    end
end
x(n) = 1;
for i=n-1:-1:1
    for j=i+1:n
        x(i) = x(i) + A(i,j)*x(j);
    end
    x(i) = -x(i)/A(i,i);
end
pi = x/norm(x,1);
```

Matlab code for Scaled Gaussian Elimination

```
function [pi] = ScaledGE(Q)

A = Q';    n = size(A);
for i=1:n-1
   for k=i+1:n
      A(i,k) = -A(i,k)/A(i,i);
   end
   for j=i+1:n
      for k=i+1:n
         A(j,k) = A(j,k) + A(j,i)*A(i,k);
      end
   end
end
x(n) = 1;
for i=n-1:-1:1
    for j=i+1:n
        x(i) = x(i) + A(i,j)*x(j);
    end
end
pi = x/norm(x,1);
```

Matlab code for GTH

```
function [pi] = GTH(Q)
A = Q';   n = size(A);
for i=1:n-1
    scale = sum(A(i+1:n,i));
    for k=i+1:n
        A(i,k) = A(i,k)/scale;
    end
    for j=i+1:n
        for k=i+1:n
            A(j,k) = A(j,k) + A(j,i)*A(i,k);
        end
    end
end
x(n) = 1;
for i=n-1:-1:1
    for j=i+1:n
        x(i) = x(i) + A(i,j)*x(j);
    end
end
pi = x/norm(x,1);
```

Basic Iterative Methods for Stationary Distributions

The Power Method

Let the chain evolve over time, step-by-step, until no change is observed from one step to the next: — at that point $zP = z$.

Example:

$$P = \begin{pmatrix} .0 & .8 & .2 \\ .0 & .1 & .9 \\ .6 & .0 & .4 \end{pmatrix}. \tag{18}$$

If the system starts in state 1: $\pi^{(0)} = (1,\ 0,\ 0)$.

Transition 1: $\rightarrow$ state 2, with probability .8; $\rightarrow$ state 3, with probability .2.

$$\pi^{(1)} = (0,\ .8,\ .2).$$

— obtained by forming the product $\pi^{(0)}P$.

Probability of state 1 after two time steps:
—probability of being in state $i = 1, 2, 3$ after 1 step, $\pi_i^{(1)}$, times probability of a transition from i to 1:

$$\sum_{i=1}^{3} \pi_i^{(1)} p_{i1} = \pi_1^{(1)} \times .0 + \pi_2^{(1)} \times .0 + \pi_3^{(1)} \times .6 = .12.$$

— or state 2 after two time steps: $.08$ $(= .0 \times .8 + .8 \times .1 + .2 \times .0)$

— or state 3 after two time steps: $.8$ $(= .0 \times .2 + .8 \times .9 + .2 \times .4)$.

$$\pi^{(2)} = (.12,\ .08,\ .8).$$

— obtained by forming the product $\pi^{(1)} P$.

$$\pi^{(2)} = (.12,\ .08,\ .8) = (0.0,\ 0.8,\ 0.2) \begin{pmatrix} .0 & .8 & .2 \\ .0 & .1 & .9 \\ .6 & .0 & .4 \end{pmatrix} = \pi^{(1)} P.$$

For any integer k, the state of the system after k transitions is

$$\pi^{(k)} = \pi^{(k-1)}P = \pi^{(k-2)}P^2 = \ldots = \pi^{(0)}P^k.$$

At step $k = 25$:

$$\pi = (.2813,\ .2500,\ .4688)$$

and thereafter, correct to 4 decimal places,

$$(.2813,\ .2500,\ .4688)\begin{pmatrix} .0 & .8 & .2 \\ .0 & .1 & .9 \\ .6 & .0 & .4 \end{pmatrix} = (.2813,\ .2500,\ .4688)$$

$=$ stationary distribution (correct to 4 decimal places).

Finite, aperiodic, and irreducible MC
— $\pi^{(k)}$ converge to π regardless of the choice of initial vector.

$$\lim_{k \to \infty} \pi^{(k)} = \pi.$$

This method is called the *power method*
— computes the right-hand eigenvector corresponding to a dominant eigenvalue of a matrix, A.

In a Markov chain context, the matrix A must be replaced with P^T.

The power method is described by the iterative procedure:

$$z^{(k+1)} = \frac{1}{\xi_k} A z^{(k)} \tag{19}$$

ξ_k is a normalizing factor, e.g., $\xi_k = \|Az^{(k)}\|_\infty$.

Normalization need not be performed at each iteration
— for Markov chain problems, it may be eliminated completely.

Rate of convergence of the power method:

$$Ax_i = \lambda_i x_i, \qquad i = 1, 2, \ldots, n,$$

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|.$$

Assume that

$$z^{(0)} = \sum_{i=1}^{n} \alpha_i x_i.$$

The rate of convergence is determined from:

$$z^{(k)} = A^k z^{(0)} = \sum_{i=1}^{n} \alpha_i \lambda_i^k x_i = \lambda_1^k \left\{ \alpha_1 x_1 + \sum_{i=2}^{n} \alpha_i \left(\frac{\lambda_i}{\lambda_1} \right)^k x_i \right\}. \quad (20)$$

— which converges to the dominant eigenvector x_1.

Convergence rate depends on the ratios $|\lambda_i|/|\lambda_1|$ for $i = 2, 3, \ldots, n$.

— the smaller these ratios, the quicker the convergence

The power method will not perform satisfactorily when $|\lambda_2| \approx |\lambda_1|$.

— major difficulties arise when $|\lambda_2| = |\lambda_1|$.

Example, cont.

$$P = \begin{pmatrix} .0 & .8 & .2 \\ .0 & .1 & .9 \\ .6 & .0 & .4 \end{pmatrix}.$$

Eigenvalues of P: $\lambda_1 = 1$ and $\lambda_{2,3} = -.25 \pm .5979i$.

$$|\lambda_2| \approx .65$$
$$.65^{10} \approx .01,$$
$$.65^{25} \approx 2 \times 10^{-5}$$
$$.65^{100} \approx 2 \times 10^{-19}$$

— two decimal places of accuracy after 10 iterations

— four places after 25 iterations,

— accurate to machine machine after 100 iterations.

Step	Initial State			Initial State			Initial State		
	1.0	.0	.0	.0	1.0	.0	.0	.0	1.0
1:	.0000	.8000	.2000	.0000	.1000	.9000	.6000	.0000	.4000
2:	.1200	.0800	.8000	.5400	.0100	.4500	.2400	.4800	.2800
3:	.4800	.1040	.4160	.2700	.4330	.2970	.1680	.2400	.5920
4:	.2496	.3944	.3560	.1782	.2593	.5626	.3552	.1584	.4864
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
10:	.2860	.2555	.4584	.2731	.2573	.4696	.2827	.2428	.4745
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
25:	.2813	.2500	.4688	.2813	.2500	.4688	.2813	.2500	.4688

It is usually necessary to normalize successive iterates,
— otherwise λ_1^k may cause approximations to become too large ($\lambda_1 > 1$) or too small ($\lambda_1 < 1$) and may result in overflow or underflow.

— Also, normalization is needed for convergence testing.

For Markov chains, $\lambda_1 = 1$ and periodic normalization is not needed.

— if $\pi^{(0)}$ is a probability vector, so also are $\pi^{(k)},\ k \geq 1$.

$$z^{(k+1)} = P^T z^{(k)} \tag{21}$$

For irreducible MC, there is one eigenvalue equal to 1.

When cyclic, there are other eigenvalues of unit modulus: — the power method will fail.

$$P = (Q\Delta t + I), \qquad \Delta t \leq 1/\max_i |q_{ii}|$$

Choose Δt so that $\Delta t < 1/\max_i |q_{ii}| \quad \Rightarrow \quad p_{ii} > 0.$

For irreducible, acyclic MCs, convergence is guaranteed

— rate of convergence is governed by the ratio $|\lambda_2|/|\lambda_1|$, i.e., by $|\lambda_2|$,

but can be incredibly slow, particularly for NCD systems.

What about repeatedly squaring P: Let $k = 2^m$ for some integer m.

— $\pi^{(k)} = \pi^{(k-1)}P$ requires k matrix-vector multiplications to obtain $\pi^{(k)}$.

— repeatedly squaring P requires only m matrix products to determine P^{2^m}, from which $\pi^{(k)} = \pi^{(0)}P^k$ is quickly computed.

Matrix-vector product: n^2 mults, Matrix-matrix product : n^3 mults,

$\Longrightarrow$ use matrix squaring when $mn^3 < 2^m n^2$, i.e., when $nm < 2^m$.

But this neglects sparsity considerations:

— matrix-vector product requires only n_z mults

— matrix-squaring increases the number of nonzeros.

The Iterative Methods of Jacobi and Gauss-Seidel

Generic iterative methods for solving $f(x) = 0$:
$f(x)$ a linear function, a nonlinear function, a system of equations,
$f(x) = Ax - b \ldots$

Take $f(x) = 0$ and write it in the form $x = g(x)$.
Now iterate with

$$x^{(k+1)} = g(x^{(k)})$$

for some initial approximation $x^{(0)}$.

Iterative methods for the solution of systems of linear equations:
Jacobi, Gauss–Seidel, and successive overrelaxation (SOR).

Given a nonhomogeneous system of linear equations

$$Ax = b, \quad \text{or, equivalently} \quad Ax - b = 0,$$

Jacobi, Gauss-Seidel and SOR use

$$x^{(k+1)} = Hx^{(k)} + c, \qquad k = 0, 1, \ldots \tag{22}$$

— by splitting the coefficient matrix A.

Given a splitting (with nonsingular M)

$$A = M - N$$

$$(M - N)x = b, \quad \text{or} \quad Mx = Nx + b,$$

which leads to the iterative procedure

$$x^{(k+1)} = M^{-1}Nx^{(k)} + M^{-1}b = Hx^{(k)} + c, \qquad k = 0, 1, \ldots.$$

$H = M^{-1}N$ is called the *iteration* matrix

— its eigenvalues determine the rate of convergence.

Example:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 &= b_3 \\ a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 &= b_4 \end{aligned}$$

Bring terms with off-diagonal components $a_{ij},\ i \neq j$ to the RHS:

$$\begin{aligned} a_{11}x_1 &= -a_{12}x_2 - a_{13}x_3 - a_{14}x_4 + b_1 \\ a_{22}x_2 &= -a_{21}x_1 - a_{23}x_3 - a_{24}x_4 + b_2 \\ a_{33}x_3 &= -a_{31}x_1 - a_{32}x_2 - a_{34}x_4 + b_3 \\ a_{44}x_4 &= -a_{41}x_1 - a_{42}x_2 - a_{43}x_3 + b_4 \end{aligned}$$

Now convert to an iterative process.

$$\begin{aligned}
a_{11}x_1^{(k+1)} &= -a_{12}x_2^{(k)} - a_{13}x_3^{(k)} - a_{14}x_4^{(k)} + b_1 \\
a_{22}x_2^{(k+1)} &= -a_{21}x_1^{(k)} - a_{23}x_3^{(k)} - a_{24}x_4^{(k)} + b_2 \\
a_{33}x_3^{(k+1)} &= -a_{31}x_1^{(k)} - a_{32}x_2^{(k)} - a_{34}x_4^{(k)} + b_3 \\
a_{44}x_4^{(k+1)} &= -a_{41}x_1^{(k)} - a_{42}x_2^{(k)} - a_{43}x_3^{(k)} + b_4
\end{aligned} \qquad (23)$$

This is the *Jacobi* iterative method

— in matrix form, A is *split* as $A = D - L - U$ where

- D is a diagonal matrix (and must be non singular)
- L is a strictly lower triangular matrix
- U is a strictly upper triangular matrix.

and so the method of Jacobi becomes equivalent to

$$Dx^{(k+1)} = (L + U)x^{(k)} + b$$

or

$$x^{(k+1)} = D^{-1}(L + U)x^{(k)} + D^{-1}b$$

For Markov chains: is

$$\pi Q = 0, \quad \text{or, equivalently, } \; Q^T \pi^T = 0$$

Set $x = \pi^T$ and $Q^T = D - (L + U)$.

Jacobi corresponds to the splitting $M = D$ and $N = (L + U)$
— its iteration matrix is given by

$$H_J = D^{-1}(L + U).$$

Note that D^{-1} exists, since $d_{jj} \neq 0$ for all j.

Given $x^{(k)}$, $x^{(k+1)}$ is found from

$$Dx^{(k+1)} = (L + U)x^{(k)}, \quad \text{i.e.,} \quad x^{(k+1)} = D^{-1}(L + U)x^{(k)},$$

In scalar form:

$$x_i^{(k+1)} = \frac{1}{d_{ii}} \left\{ \sum_{j \neq i} (l_{ij} + u_{ij})\, x_j^{(k)} \right\}, \qquad i = 1, 2, \ldots, n. \tag{24}$$

Example:

$$P = \begin{pmatrix} 0.5 & 0.5 & 0.0 & 0.0 \\ 0.0 & 0.5 & 0.5 & 0.0 \\ 0.0 & 0.0 & 0.5 & 0.5 \\ .125 & .125 & .25 & .5 \end{pmatrix}.$$

Write $\pi P = \pi$ as $\pi(P - I) = 0$ and take $Q = P - I$:

$$Q = \begin{pmatrix} -.5 & .5 & 0 & 0 \\ 0 & -.5 & .5 & 0 \\ 0 & 0 & -.5 & .5 \\ .125 & .125 & .25 & -.5 \end{pmatrix}$$ and transpose to get

$$\begin{pmatrix} -.5 & 0 & 0 & .125 \\ .5 & -.5 & 0 & .125 \\ 0 & .5 & -.5 & .250 \\ 0 & 0 & .5 & -.500 \end{pmatrix} \begin{pmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \\ \pi_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

$$\begin{aligned} -.5\pi_1 + 0\pi_2 + 0\pi_3 + .125\pi_4 &= 0 \\ .5\pi_1 - .5\pi_2 + 0\pi_3 + .125\pi_4 &= 0 \\ 0\pi_1 + .5\pi_2 - .5\pi_3 + .25\pi_4 &= 0 \\ 0\pi_1 + 0\pi_2 + .5\pi_3 - .5\pi_4 &= 0 \end{aligned}$$

$$\begin{aligned} -.5\pi_1 &= -.125\pi_4 \\ -.5\pi_2 &= -.5\pi_1 - .125\pi_4 \\ .5\pi_3 &= -.5\pi_2 - .25\pi_4 \\ -.5\pi_4 &= -.5\pi_3 \end{aligned} \qquad (25)$$

— which allows us to write the iterative version.

Iterative version:

$$\begin{aligned}
-.5\pi_1^{(k+1)} &= & & & -.125\pi_4^{(k)} \\
-.5\pi_2^{(k+1)} &= & -.5\pi_1^{(k)} & & -.125\pi_4^{(k)} \\
-.5\pi_3^{(k+1)} &= & & -.5\pi_2^{(k)} & -.25\pi_4^{(k)} \\
-.5\pi_4^{(k+1)} &= & & -.5\pi_3^{(k)} &
\end{aligned}$$

Simplifying:

$$\pi_1^{(k+1)} = .25\pi_4^{(k)}; \quad \pi_2^{(k+1)} = \pi_1^{(k)} + .25\pi_4^{(k)}; \quad \pi_3^{(k+1)} = \pi_2^{(k)} + .5\pi_4^{(k)}; \quad \pi_4^{(k+1)} = \pi_3^{(k)}.$$

Begin iterating with $\pi^{(0)} = (.5,\ .25,\ .125,\ .125)$

$$\begin{aligned}
\pi_1^{(1)} &= & & & .25\pi_4^{(0)} &= .25 \times .125 = .03125 \\
\pi_2^{(1)} &= & \pi_1^{(0)} & & + .25\pi_4^{(0)} &= .5 + .25 \times .125 = .53125 \\
\pi_3^{(1)} &= & & \pi_2^{(0)} & + .5\pi_4^{(0)} &= .25 + .5 \times .125 = .31250 \\
\pi_4^{(1)} &= & & \pi_3^{(0)} & &= .12500
\end{aligned}$$

In this particular example, normalization is not necessary
— at any iteration $k+1$,

$$\sum_{i=1}^{4} \pi_i^{(k+1)} = .25\pi_4^{(k)} + \pi_1^{(k)} + .25\pi_4^{(k)} + \pi_2^{(k)} + .50\pi_4^{(k)} + \pi_3^{(k)} = \sum_{i=1}^{4} \pi_i^{(k)} = 1$$

Continuing with the iterations:

$$\begin{aligned}
\pi^{(0)} &= (.50000,\ .25000,\ .12500,\ .12500) \\
\pi^{(1)} &= (.03125,\ .53125,\ .31250,\ .12500) \\
\pi^{(2)} &= (.03125,\ .06250,\ .59375,\ .31250) \\
\pi^{(3)} &= (.078125,\ .109375,\ .21875,\ .59375) \\
&\vdots
\end{aligned}$$

Substituting $Q^T = D - (L + U)$ into $Q^T x = 0$ gives $(L + U)x = Dx$, and the eigenvalue equation:

$$D^{-1}(L + U)x = x \tag{26}$$

x is the right-hand eigenvector corresponding to a unit eigenvalue of $H_J = D^{-1}(L + U)$ — the Jacobi iteration matrix.

H_J obviously has a unit eigenvalue.

Also, from the zero column-sum property of Q^T:

$$d_{jj} = \sum_{i=1,\ i\neq j}^{n} (l_{ij} + u_{ij}), \qquad j = 1, 2, \ldots$$

with $l_{ij},\ u_{ij} \leq 0$ for all $i, j;\ i \neq j$.

From Gerschgorin, no eigenvalue of H_J can have modulus > 1.

So π is the eigenvector corresponding to a dominant eigenvalue of H_J — Jacobi is identical to the power method applied to H_J.

Example, cont. Jacobi iteration matrix:

$$H_J = \begin{pmatrix} -.5 & 0 & 0 & 0 \\ 0 & -.5 & 0 & 0 \\ 0 & 0 & -.5 & 0 \\ 0 & 0 & 0 & -.5 \end{pmatrix}^{-1} \begin{pmatrix} 0 & 0 & 0 & -.125 \\ -.5 & 0 & 0 & -.125 \\ 0 & -.5 & 0 & -.5 \\ 0 & 0 & -.5 & 0 \end{pmatrix}$$

$$= \begin{pmatrix} 0 & 0 & 0 & .25 \\ 1.0 & 0 & 0 & .25 \\ 0 & 1.0 & 0 & .5 \\ 0 & 0 & 1.0 & 0 \end{pmatrix}$$

$$\text{Eigenvalues :} \qquad \lambda_1 = 1.0, \quad \lambda_2 = .7718, \quad \lambda_{3,4} = -0.1141 \pm 0.5576i$$

Since $|\lambda_2| = 0.7718$ and $0.7718^{50} = .00000237$

— implies about 5 to 6 places of accuracy after 50 iterations.

Method of Gauss-Seidel

In Jacobi, the components of $x^{(k+1)}$ are obtained one after the other: $x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_n^{(k+1)}$.

When evaluating $x_i^{(k+1)}$, only components from $x^{(k)}$ are used

— even though more accurate elements from the current iteration $x_j^{(k+1)}$ for $j < i$ are available.

Gauss–Seidel makes use of the most recently available approximations

— accomplished by overwriting each element as soon as a new approximation to it is computed.

$$
\begin{aligned}
a_{11}x_1^{(k+1)} &= & -a_{12}x_2^{(k)} & - a_{13}x_3^{(k)} & - a_{14}x_4^{(k)} + b_1 \\
a_{22}x_2^{(k+1)} &= -a_{21}x_1^{(k+1)} & & - a_{23}x_3^{(k)} & - a_{24}x_4^{(k)} + b_2 \\
a_{33}x_3^{(k+1)} &= -a_{31}x_1^{(k+1)} & - a_{32}x_2^{(k+1)} & & - a_{34}x_4^{(k)} + b_3 \\
a_{44}x_4^{(k+1)} &= -a_{41}x_1^{(k+1)} & - a_{42}x_2^{(k+1)} & - a_{43}x_3^{(k+1)} & + b_4
\end{aligned}
$$

In equation 2, we use $x_1^{(k+1)}$ rather than $x_1^{(k)}$.

In equation 3, the new values of x_1 and x_2 are used.

In equation 4, new values of $x_1,\ x_2$ and x_3 are used.

With n linear equations in n unknowns:

$$
a_{ii}x_i^{(k+1)} = \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)} \right), \qquad i = 1, 2, \ldots, n. \tag{27}
$$

Rearranging:

$$\begin{aligned}
a_{11}x_1^{(k+1)} &= -a_{12}x_2^{(k)} - a_{13}x_3^{(k)} - a_{14}x_4^{(k)} + b_1 \\
a_{21}x_1^{(k+1)} + a_{22}x_2^{(k+1)} &= -a_{23}x_3^{(k)} - a_{24}x_4^{(k)} + b_2 \\
a_{31}x_1^{(k+1)} + a_{32}x_2^{(k+1)} + a_{33}x_3^{(k+1)} &= -a_{34}x_4^{(k)} + b_3 \\
a_{41}x_1^{(k+1)} + a_{42}x_2^{(k+1)} + a_{44}x_4^{(k+1)} + a_{43}x_3^{(k+1)} &= b_4
\end{aligned} \tag{28}$$

Using the same $D - L - U$ splitting:

$$(D - L)x^{(k+1)} = Ux^{(k)} + b \tag{29}$$

$$x^{(k+1)} = (D - L)^{-1}Ux^{(k)} + (D - L)^{-1}b. \tag{30}$$

Iteration matrix for Gauss-Seidel:

$$H_{GS} = (D - L)^{-1}U.$$

— corresponds to the splitting $M = (D - L)$ and $N = U$.

Gauss-Seidel usually, but not always, converges faster than Jacobi.

For Markov chains:

$$x^{(k+1)} = (D-L)^{-1}Ux^{(k)}, \quad \text{i.e.,} \;\; x^{(k+1)} = H_{GS}x^{(k)}.$$

and $(D-L)^{-1}$ exists.

$\pi = x^T$ satisfies the eigenvalue equation $H_{GS}x = x$.

The unit eigenvalue of H_{GS} is a dominant eigenvalue
— Gauss–Seidel is identical to the power method applied to H_{GS}.

Example, cont.

$$\begin{aligned}
-.5\pi_1 &= & & & -.125\pi_4 \\
-.5\pi_2 &= & -.5\pi_1 & & -.125\pi_4 \\
-.5\pi_3 &= & & -.5\pi_2 & -.25\pi_4 \\
-.5\pi_4 &= & & -.5\pi_3 &
\end{aligned}$$

Gauss-Seidel iteration:

$$\begin{aligned}
-.5\pi_1^{(k+1)} &= & & & -.125\pi_4^{(k)} \\
-.5\pi_2^{(k+1)} &= & -.5\pi_1^{(k+1)} & & -.125\pi_4^{(k)} \\
-.5\pi_3^{(k+1)} &= & & -.5\pi_2^{(k+1)} & -.25\pi_4^{(k)} \\
-.5\pi_4^{(k+1)} &= & & -.5\pi_3^{(k+1)} &
\end{aligned}$$

$$\pi_1^{(k+1)} = .25\pi_4^{(k)}; \qquad \pi_2^{(k+1)} = \pi_1^{(k+1)} + .25\pi_4^{(k)};$$
$$\pi_3^{(k+1)} = \pi_2^{(k+1)} + .5\pi_4^{(k)}; \qquad \pi_4^{(k+1)} = \pi_3^{(k+1)}.$$

$$\pi^{(0)} = (.5,\ .25,\ .125,\ .125)$$

$$\begin{aligned}
\pi_1^{(1)} &= & & & .25\pi_4^{(0)} &= .25 \times .125 = .03125 \\
\pi_2^{(1)} &= & \pi_1^{(1)} & & + .25\pi_4^{(0)} &= .03125 + .25 \times .125 = .06250 \\
\pi_3^{(1)} &= & & \pi_2^{(1)} & + .5\pi_4^{(0)} &= .06250 + .5 \times .125 = .1250 \\
\pi_4^{(1)} &= & & \pi_3^{(1)} & &= .12500
\end{aligned}$$

The elements of $\pi^{(1)}$ do not add up to one $\Rightarrow$ must normalize.

Dividing each element by $\|\pi^{(1)}\| = 0.90625$

$$\pi^{(1)} = (0.090909,\ 0.181818,\ 0.363636,\ .363636) = \frac{1}{11}(1,\ 2,\ 4,\ 4)$$

Continuing the iterations:

$$\begin{aligned}
\pi^{(0)} &= (0.090909,\ 0.181818,\ 0.363636,\ .363636) \\
\pi^{(1)} &= (0.090909,\ 0.181818,\ 0.363636,\ .363636) \\
\pi^{(2)} &= (0.090909,\ 0.181818,\ 0.363636,\ .363636) \\
\pi^{(3)} &= (0.090909,\ 0.181818,\ 0.363636,\ .363636)
\end{aligned}$$

and so on.

For this particular example, Gauss-Seidel converges in only 1 iteration!

To see why, we need examine the iteration matrix

$$H_{GS} = (D - L)^{-1}U. \tag{31}$$

$$D = -.5I_4, \quad L = \begin{pmatrix} 0 & 0 & 0 & 0 \\ -.5 & 0 & 0 & 0 \\ 0 & -.5 & 0 & 0 \\ 0 & 0 & -.5 & 0 \end{pmatrix}, \quad U = \begin{pmatrix} 0 & 0 & 0 & -.125 \\ 0 & 0 & 0 & -.125 \\ 0 & 0 & 0 & -.25 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

and

$$H_{GS} = (D - L)^{-1}U = \begin{pmatrix} 0 & 0 & 0 & .25 \\ 0 & 0 & 0 & .50 \\ 0 & 0 & 0 & 1.0 \\ 0 & 0 & 0 & 1.0 \end{pmatrix}$$

The only nonzero eigenvalue of H_{GS} is the last diagonal element.

Rate of convergence depends on the magnitude of the subdominant eigenvalue (here equal to 0), convergence must occur immediately after the first iteration.

Gauss–Seidel corresponds to computing the i^{th} component of the current approximation from $i = 1$ through $i = n$, i.e., from top to bottom — *Forward* Gauss–Seidel.

Backward Gauss–Seidel iteration takes the form

$$(D - U)x^{(k+1)} = Lx^{(k)}, \qquad k = 0, 1, \ldots$$

— corresponds to computing the components from bottom to top.

Use forward when most of the elemental mass is below the diagonal — the method essentially works with the inverse of the lower triangular portion, $(D - L)^{-1}$, and intuitively, the closer this is to the inverse of the entire matrix, the faster the convergence.

Choose splitting such that M is as close to Q^T as possible, subject only to the constraint that M^{-1} be easy to find.

The Method of Successive Over Relaxation

For $Ax = b$:

$$a_{ii}x_i^{(k+1)} = a_{ii}(1-\omega)x_i^{(k)} + \omega\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right), \qquad i = 1, 2, \ldots, n.$$

Expression within the parenthesis is Gauss-Seidel:
— SOR reduces to Gauss–Seidel when $\omega = 1$.

A *backward* SOR relaxation may also be written.

For Markov chains $Q^T x = (D - L - U)x = 0$:

$$x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \omega\left\{\frac{1}{d_{ii}}\left(\sum_{j=1}^{i-1} l_{ij}x_j^{(k+1)} + \sum_{j=i+1}^{n} u_{ij}x_j^{(k)}\right)\right\}, \qquad i = 1, 2, \ldots, n,$$

$$x^{(k+1)} = (1-\omega)x^{(k)} + \omega\left\{D^{-1}(Lx^{(k+1)} + Ux^{(k)})\right\}. \qquad (32)$$

Rewriting

$$(D - \omega L)x^{(k+1)} = [(1 - \omega)D + \omega U]x^{(k)},$$

$$x^{(k+1)} = (D - \omega L)^{-1}[(1 - \omega)D + \omega U]x^{(k)}, \tag{33}$$

SOR iteration matrix:

$$H_\omega = (D - \omega L)^{-1}[(1 - \omega)D + \omega U].$$

— corresponds to the splitting

$$M = \omega^{-1}\,[D - \omega L]$$

$$N = \omega^{-1}\,[(1 - \omega)D + \omega U]$$

π is an eigenvector corresponding to a unit eigenvalue of H_ω
— not necessarily the dominant eigenvalue

Eigenvalues depends on the choice of ω.

SOR method converges only if $0 < \omega < 2$
— a necessary, but not sufficient, condition for convergence. When 1 is the dominant eigenvalue, the SOR method is identical to the power method applied to H_ω.

Choose ω to maximize the difference between 1 and the subdominant eigenvalue of H_ω.

Difficult to find the optimal, or even a reasonable, value for ω.

Adaptive procedures work for a related series of problems.

Power method can find π using P^T, H_J, H_{GS}, and H_ω.

Computational effort per iteration step is the same for all four — apply to the matrix that yields convergence in the smallest number of iterations.

MATLAB code, on next slide, performs a fixed number of iterations of the basic Jacobi, forward Gauss–Seidel, or SOR on $Ax = b$.

Input matrix A $(= Q^T)$, initial approximation x_0, right-hand vector b ($b = 0$ is permitted — in which case, a normalization (line 15) must also be performed), the number of iterations to be carried out, $itmax$.

Returns the computed solution, $soln$, and a vector containing the residual computed at each iteration, $resid$.

```
function [soln,resid] = gs(A,x0,b,itmax)
%   Performs ``itmax'' iterations of Jacobi/Gauss-Seidel/SOR on Ax = b

[n,n] = size(A); L = zeros(n,n); U = L;  D = diag(diag(A));
for i = 1:n,
    for j = 1:n,
         if i<j, U(i,j) = -A(i,j); end
         if i>j, L(i,j) = -A(i,j); end
    end
end
 M = inv(D-L);  B = M*U;          %  B is GS iteration matrix
 %M = inv(D);   B = M*(L+U);      %  Use this for Jacobi
 %w = 1.1; M = inv(D-wL);  B = M*((1-w)*D + w*U)  % Use this for SOR

for iter = 1:itmax,
    soln = B*x0+M*b;
    if norm(b,2) == 0 soln = soln/norm(soln,1); end %  Normalize when b = 0.
    resid(iter) = norm(A*soln-b,2);
    x0 = soln;
end
resid = resid';
if norm(b,2) == 0 soln = soln/norm(soln,1); end %  Normalize when b = 0.
```

Data Structures for Large Sparse Matrices

For large-scale Markov chains a two-dimensional array will not work
— need to take advantage of sparsity and store a compacted version.

Only numerical operation on A is $z = Ax$ and $z = A^T x$
— iterative methods do not modify the matrix.

Simple approach: use a real (double-precision) one-dimensional array aa to store nonzero elements and two integer arrays ia and ja to indicate row and column positions .

If a_{ij} is stored in position k of aa, i.e., $aa(k) = a_{ij}$
— then $ia(k) = i$ and $ja(k) = j$.

Example:

$$A = \begin{pmatrix} -2.1 & 0.0 & 1.7 & 0.4 \\ 0.8 & -0.8 & 0.0 & 0.0 \\ 0.2 & 1.5 & -1.7 & 0.0 \\ 0.0 & 0.3 & 0.2 & -0.5 \end{pmatrix}$$

may be stored as

$aa:$	-2.1	1.7	0.4	-0.8	0.8	-1.7	0.2	1.5	-0.5	0.3	0.2
$ia:$	1	1	1	2	2	3	3	3	4	4	4
$ja:$	1	3	4	2	1	3	1	2	4	2	3

or as

$aa:$	-2.1	0.8	0.2	-0.8	1.5	0.3	1.7	-1.7	0.2	0.4	-0.5
$ia:$	1	2	3	2	3	4	1	3	4	1	4
$ja:$	1	1	1	2	2	2	3	3	3	4	4

or yet again as

$aa:$	-2.1	-0.8	-1.7	-0.5	1.7	0.8	0.4	0.2	0.3	1.5	0.2
$ia:$	1	2	3	4	1	2	1	4	4	3	3
$ja:$	1	2	3	4	3	1	4	3	2	2	1

Algorithm: Sparse Matrix-Vector Multiply I, $z = Ax$

1. Set $z(i) = 0$ for all i.
2. For $next = 1$ to n_z do
   - Set $nrow = ia(next)$.
   - Set $ncol = ja(next)$.
   - Compute $z(nrow) = z(nrow) + aa(next) \times x(ncol)$.

— where n_z is the number of nonzero elements in A.

To compute $z = A^T x$, it suffices to interchange the arrays ia and ja.

When multiplying a matrix by a vector, each element of the matrix is used only once:
— a_{ij} is multiplied with x_j and constitutes one term of $\sum_{k=1}^{n} a_{ij}x_j$.

Must begin by setting the elements of z to zero
— the inner products are computed in a piecemeal fashion.

Imposing an ordering removes this constraint
— store the nonzero elements of A by rows.

Can now replace ia with a smaller array
— the k^{th} element of ia denotes the position in aa and ja at which the first element of row k is stored: $ia(1) = 1$.

Store the first empty position of aa and ja in position $(n+1)$ of ia — $ia(n+1) = n_z + 1$.

The number of nonzero elements in row i is $ia(i+1) - ia(i)$
— easy to immediately go to any row of the matrix

The $ia(i+1) - ia(i)$ non-zero elements of row i begin at $aa[ia(i)]$.

Example, cont.

$$A = \begin{pmatrix} -2.1 & 0.0 & 1.7 & 0.4 \\ 0.8 & -0.8 & 0.0 & 0.0 \\ 0.2 & 1.5 & -1.7 & 0.0 \\ 0.0 & 0.3 & 0.2 & -0.5 \end{pmatrix}$$

Harwell Boeing format:

$aa:$	-2.1	1.7	0.4	-0.8	0.8	-1.7	0.2	1.5	-0.5	0.3	0.2
$ja:$	1	3	4	2	1	3	1	2	4	2	3
$ia:$	1	4	6	9	12						

The elements in any row can be in any order:
— it suffices that all the nonzero elements of row i come before those of row $i+1$ and after those of row $i-1$.

Sparse Matrix-Vector Multiply II.

1. For $i = 1$ to n do
   - Set $sum = 0$.
   - Set $initial = ia(i)$.
   - Set $last = ia(i+1) - 1$.
   - For $j = initial$ to $last$ do
     - Compute $sum = sum + aa(j) \times x(ja(j))$.
   - Set $z(i) = sum$.

Incorporating Harwell-Boeing into SOR is no more difficult than for the power method, Jacobi or Gauss-Seidel.

SOR formula:

$$x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \frac{\omega}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right).$$

Scale the matrix ($a_{ii} = 1$ for all i) and setting $b_i = 0$, for all i:

$$x_i^{(k+1)} = (1-\omega)x_i^{(k)} - \omega\left(\sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} + \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right)$$

$$= x_i^{(k)} - \omega\left(\sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} + a_{ii}x_i^{(k)} + \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right).$$

Code for a single iteration now follows.

Sparse SOR. At iteration k:

1. For $i = 1$ to n do
   - Set $sum = 0$.
   - Set $initial = ia(i)$.
   - Compute $last = ia(i+1) - 1$.
   - For $j = initial$ to $last$ do
     - Compute $sum = sum + aa(j) \times x(ja(j))$.
   - Compute $x(i) = x(i) - \omega \times sum$.

Only difference with straightforward matrix-vector multiply is in the very last line.

For Markov chains, A must be replaced by Q^T
— may pose a problem if Q is generated by rows.

Initial Approximations, Normalizations and Convergence

For iterative methods

- What should we choose as the initial vector?
- What happens to this vector at each iteration?
- How do we know when convergence has occurred?

Choice of an initial starting vector: something simple? Be careful!

Example:

$$Q = \begin{pmatrix} -\lambda & \lambda \\ \mu & -\mu \end{pmatrix}.$$

Gauss–Seidel with $\pi^{(0)} = (1,\ 0)$ gives $\pi^{(k)} = (0,\ 0)$ for all $k \geq 1$.

$$x^{(1)} = (D-L)^{-1}Ux^{(0)} = (D-L)^{-1}\begin{pmatrix} 0 & -\mu \\ 0 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = (D-L)^{-1}\begin{pmatrix} 0 \\ 0 \end{pmatrix} = 0$$

— all successive vectors $x^{(k)},\ k = 1, 2, \ldots$ are identically equal to zero.

If some approximation to the solution is known, it should be used
— otherwise assigned random numbers uniformly distributed between 0 and 1 and normalized
— alternatively set each element equal to $1/n$.

In some methods, successive iterates may not be probability vectors
— normalize by dividing each component by the sum of the components.

In large-scale Markov chains, this can be costly
— it can almost double the complexity per iteration.

In most cases, it is not necessary to normalize at *each* iteration.

Normalization is required only:
(1) prior to convergence testing
(2) to prevent problems of overflow and underflow.

Underflow: — periodically check the magnitude of the elements and set those less than a certain threshold (e.g., 10^{-25}) to zero.

Underflow of some components can arise even when the approximations are normalized at each iteration
— when normalization is not performed all the elements can underflow.

Solution: Ensure that at least one element exceeds a certain minimum threshold and initiate a normalization when this test fails.

Overflow: — unlikely since the eigenvalues of the iteration matrices should not exceed 1.

Keep check on the magnitude of the largest element and normalize if it exceeds a certain maximum threshold, say 10^{10}.

Convergence Testing and Numerical Accuracy.

Number of iterations k needed to satisfy a tolerance ϵ :

$$\rho^k = \epsilon, \quad \text{i.e., } k = \frac{\log \epsilon}{\log \rho},$$

where ρ is the spectral radius of the iteration matrix.

For MCs, use the magnitude of the subdominant eigenvalue in place of ρ.

For $\epsilon = 10^{-6}$, number of iterations needed for different spectral radii:

ρ	.1	.5	.6	.7	.8	.9	.95	.99	.995	.999
k	6	20	27	39	62	131	269	1,375	2,756	13,809

But size of the subdominant eigenvalue is not known
— so examine some norm of the difference of successive iterates.

OK for rapidly converges processes but not for slow convergence.

Example:

$$Q = \begin{pmatrix} -.6 & 0 & .6 & 0 \\ .0002 & -.7 & 0 & .6998 \\ .1999 & .0001 & -.2 & 0 \\ 0 & .5 & 0 & -.5 \end{pmatrix}$$

Gauss-Seidel with $\pi^{(0)} = (.25, .25, .25, .25)$:

$$\begin{aligned} \pi^{(199)} &= (0.112758,\ 0.228774,\ 0.338275,\ 0.3201922) \\ \pi^{(200)} &= (0.112774,\ 0.228748,\ 0.338322,\ 0.3201560) \end{aligned}$$

— looks like about 4 decimal places of accuracy.

For $\epsilon = 0.001$, the iterative procedure stops prior to iteration 200.

$$\pi = (0.131589,\ 0.197384,\ 0.394768,\ 0.276259)$$

Gauss-Seidel will eventually converge onto this solution.

$\lambda_2 = 0.9992 \ \Rightarrow\ 6,000$ iterations are needed for accuracy of $\epsilon = .001$

Do not successive iterates: $\|\pi^{(k)} - \pi^{(k-1)}\| < \epsilon$,

— but iterates spaced further apart, e.g.,

$$\|\pi^{(k)} - \pi^{(k-m)}\| < \epsilon.$$

Choose m as a function of the convergence rate

— alternatively, when the iteration number k is

$$\begin{aligned} k &< 100 && \text{let } m = 5 \\ 100 \leq k &< 500 && \text{let } m = 10 \\ 500 \leq k &< 1,000 && \text{let } m = 20 \\ k &\geq 1,000 && \text{let } m = 50 \end{aligned}$$

Convergence to a vector with all small elements:

— e.g., $n = 100,000$ approximately equally probable states.

Solution $\approx 10^{-5}$, $\epsilon = 10^{-3}$ and $\pi_i^{(0)} = 1/n$

— can spell trouble.

Recommended approach: use a relative measure; e.g.,

$$\max_i \left(\frac{|\pi_i^{(k)} - \pi_i^{(k-m)}|}{|\pi_i^{(k)}|} \right) < \epsilon.$$

— removes the $exponent$ from consideration in the convergence test

— gives a better estimate of the $precision$ that has been achieved.

Another convergence criterion: Residuals

— the magnitude of $\|\pi^{(k)}Q\|$ which should be small

— but a small residual does not always imply a small error!

A small residual is a necessary condition for the error in the solution vector to be small — but it is not a sufficient condition.

Envisage a battery of convergence tests, all of which must be satisfied before the approximation is accepted as being sufficiently accurate.

Frequency of convergence testing:

During each iteration? — may be wasteful, especially when the matrix is very large and the iterative method is converging slowly.

Sometimes it is possible to estimate the rate of convergence and calculate the approximate numbers of iterations that must be performed.

Alternatively, implement a relatively inexpensive convergence test (e.g., using the relative difference in the first nonzero component of successive approximations) at each iteration.

When this simple test is satisfied, begin more rigorous testing.

Block Iterative Methods

Point iterative methods versus *block* iterative methods. — beneficial when the state space can be meaningfully partitioned into subsets.

They require more computation per iteration, but convergence faster.

$$\pi Q = (\pi_1,\ \pi_2,\ \ldots,\ \pi_N) \begin{pmatrix} Q_{11} & Q_{12} & \cdots & Q_{1N} \\ Q_{21} & Q_{22} & \cdots & Q_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ Q_{N1} & Q_{N2} & \cdots & Q_{NN} \end{pmatrix} = 0.$$

Block splitting:

$$Q^T = D_N - (L_N + U_N),$$

$D_N,\ L_N,\ U_N$

— are diagonal, strictly lower, and strictly upper *block* matrices.

$$D_N = \begin{pmatrix} D_{11} & 0 & \cdots & 0 \\ 0 & D_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & D_{NN} \end{pmatrix}$$

$$L_N = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ L_{21} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ L_{N1} & L_{N2} & \cdots & 0 \end{pmatrix}, \qquad U_N = \begin{pmatrix} 0 & U_{12} & \cdots & U_{1N} \\ 0 & 0 & \cdots & U_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}.$$

Block Gauss–Seidel:

$$(D_N - L_N)x^{(k+1)} = U_N x^{(k)},$$

— corresponds to the splitting $M = (D_N - L_N)$; $N = U_N$.

The i^{th} block equation:

$$D_{ii}x_i^{(k+1)} = \left(\sum_{j=1}^{i-1} L_{ij}x_j^{(k+1)} + \sum_{j=i+1}^{N} U_{ij}x_j^{(k)} \right), \qquad i = 1, 2, \ldots, N \tag{34}$$

— subvectors x_i are partitioned conformally with $D_{ii},\ i = 1, 2, \ldots, N$.

Need to solve N systems of linear equations at each iteration:

$$D_{ii}x_i^{(k+1)} = z_i, \qquad i = 1, 2, \ldots, N, \tag{35}$$

where

$$z_i = \left(\sum_{j=1}^{i-1} L_{ij}x_j^{(k+1)} + \sum_{j=i+1}^{N} U_{ij}x_j^{(k)} \right), \qquad i = 1, 2, \ldots, N.$$

RHS, z_i — can always be computed before i^{th} system has to be solved.

Irreducible Markov chain: all N systems of equations are nonhomogeneous with nonsingular coefficient matrices
— solve using direct or iterative methods.

Direct method: form an LU decomposition of each block D_{ii} once
— solving $D_{ii}x_i^{(k+1)} = z_i,\ i = 1, \ldots, N$ in each iteration is a forward and backward substitution procedure.

Iterative method (e.g., point Gauss-Seidel):
— local iterative methods inside a global iterative method!

Block Jacobi and SOR methods:

$$D_{ii}x_i^{(k+1)} = \left(\sum_{j=1}^{i-1} L_{ij}x_j^{(k)} + \sum_{j=1+1}^{N} U_{ij}x_j^{(k)} \right), \qquad i = 1, 2, \ldots, N,$$

$$x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \omega \left\{ D_{ii}^{-1} \left(\sum_{j=1}^{i-1} L_{ij}x_j^{(k+1)} + \sum_{j=i+1}^{N} U_{ij}x_j^{(k)} \right) \right\}, \quad i = 1, 2, \ldots, N.$$

Example: Block Gauss-Seidel

$$Q = \begin{pmatrix} -4.0 & 2.0 & 1.0 & 0.5 & 0.5 \\ 0.0 & -3.0 & 3.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & -1.0 & 0.0 & 1.0 \\ 1.0 & 0.0 & 0.0 & -5.0 & 4.0 \\ 1.0 & 0.0 & 0.0 & 1.0 & -2.0 \end{pmatrix}.$$

$$Q^T x = \begin{pmatrix} D_{11} & -U_{12} \\ -L_{21} & D_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$= \left(\begin{array}{ccc|cc} -4.0 & 0.0 & 0.0 & 1.0 & 1.0 \\ 2.0 & -3.0 & 0.0 & 0.0 & 0.0 \\ 1.0 & 3.0 & -1.0 & 0.0 & 0.0 \\ \hline 0.5 & 0.0 & 0.0 & -5.0 & 1.0 \\ 0.5 & 0.0 & 1.0 & 4.0 & -2.0 \end{array}\right) \left(\begin{array}{c} \pi_1 \\ \pi_2 \\ \pi_3 \\ \hline \pi_4 \\ \pi_5 \end{array}\right) = \left(\begin{array}{c} 0 \\ 0 \\ 0 \\ \hline 0 \\ 0 \end{array}\right).$$

Equation (34):

$$D_{ii}x_i^{(k+1)} = \left(\sum_{j=1}^{i-1} L_{ij}x_j^{(k+1)} + \sum_{j=i+1}^{2} U_{ij}x_j^{(k)} \right), \qquad i = 1, 2.$$

The two block equations:

$$i = 1: \qquad D_{11}x_1^{(k+1)} = \left(\sum_{j=1}^{0} L_{1j}x_j^{(k+1)} + \sum_{j=2}^{2} U_{1j}x_j^{(k)} \right) = U_{12}x_2^{(k)},$$

$$i = 2: \qquad D_{22}x_2^{(k+1)} = \left(\sum_{j=1}^{1} L_{2j}x_j^{(k+1)} + \sum_{j=3}^{2} U_{2j}x_j^{(k)} \right) = L_{21}x_1^{(k+1)}.$$

Writing these block equations in full:

$$\begin{pmatrix} -4.0 & 0.0 & 0.0 \\ 2.0 & -3.0 & 0.0 \\ 1.0 & 3.0 & -1.0 \end{pmatrix} \begin{pmatrix} \pi_1^{(k+1)} \\ \pi_2^{(k+1)} \\ \pi_3^{(k+1)} \end{pmatrix} = - \begin{pmatrix} 1.0 & 1.0 \\ 0.0 & 0.0 \\ 0.0 & 0.0 \end{pmatrix} \begin{pmatrix} \pi_4^{(k)} \\ \pi_5^{(k)} \end{pmatrix}$$

$$\begin{pmatrix} -5.0 & 1.0 \\ 4.0 & -2.0 \end{pmatrix} \begin{pmatrix} \pi_4^{(k+1)} \\ \pi_5^{(k+1)} \end{pmatrix} = - \begin{pmatrix} 0.5 & 0.0 & 0.0 \\ 0.5 & 0.0 & 1.0 \end{pmatrix} \begin{pmatrix} \pi_1^{(k+1)} \\ \pi_2^{(k+1)} \\ \pi_3^{(k+1)} \end{pmatrix}.$$

Solve using *LU* decompositions

— D_{11} is already lower triangular, so just *forward substitute*.

$$D_{11} = \begin{pmatrix} -4.0 & 0.0 & 0.0 \\ 2.0 & -3.0 & 0.0 \\ 1.0 & 3.0 & -1.0 \end{pmatrix} = L \times I.$$

LU decomposition of D_{22}:

$$D_{22} = \begin{pmatrix} -5.0 & 1.0 \\ 4.0 & -2.0 \end{pmatrix} = \begin{pmatrix} 1.0 & 0.0 \\ -0.8 & 1.0 \end{pmatrix} \begin{pmatrix} -5.0 & 1.0 \\ 0.0 & -1.2 \end{pmatrix} = L \times U.$$

Take initial approximation to be

$$\pi^{(0)} = (0.2,\ 0.2,\ 0.2,\ 0.2,\ 0.2,)^T$$

First block equation:

$$\begin{pmatrix} -4.0 & 0.0 & 0.0 \\ 2.0 & -3.0 & 0.0 \\ 1.0 & 3.0 & -1.0 \end{pmatrix} \begin{pmatrix} \pi_1^{(1)} \\ \pi_2^{(1)} \\ \pi_3^{(1)} \end{pmatrix} = - \begin{pmatrix} 1.0 & 1.0 \\ 0.0 & 0.0 \\ 0.0 & 0.0 \end{pmatrix} \begin{pmatrix} 0.2 \\ 0.2 \end{pmatrix} = - \begin{pmatrix} 0.4 \\ 0.0 \\ 0.0 \end{pmatrix}.$$

Forward substitution successively gives

$$\pi_1^{(1)} = 0.1000, \quad \pi_2^{(1)} = 0.0667, \quad \text{and} \quad \pi_3^{(1)} = 0.3000.$$

Second block system: $LUx = b$

$$\begin{pmatrix} 1.0 & 0.0 \\ -0.8 & 1.0 \end{pmatrix} \begin{pmatrix} -5.0 & 1.0 \\ 0.0 & -1.2 \end{pmatrix} \begin{pmatrix} \pi_4^{(1)} \\ \pi_5^{(1)} \end{pmatrix}$$

$$= - \begin{pmatrix} 0.5 & 0.0 & 0.0 \\ 0.5 & 0.0 & 1.0 \end{pmatrix} \begin{pmatrix} 0.1000 \\ 0.0667 \\ 0.3000 \end{pmatrix} = - \begin{pmatrix} 0.0500 \\ 0.3500 \end{pmatrix}.$$

Find z from $Lz = b$ and then x from $Ux = z$:

$Lz = b$

$$\begin{pmatrix} 1.0 & 0.0 \\ -0.8 & 1.0 \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = - \begin{pmatrix} 0.0500 \\ 0.3500 \end{pmatrix}$$

we obtain

$$z_1 = -0.0500 \quad \text{and} \quad z_2 = -0.3900.$$

$Ux = z$

$$\begin{pmatrix} -5.0 & 1.0 \\ 0.0 & -1.2 \end{pmatrix} \begin{pmatrix} \pi_4^{(1)} \\ \pi_5^{(1)} \end{pmatrix} = \begin{pmatrix} -0.0500 \\ -0.3900 \end{pmatrix}$$

gives

$$\pi_4^{(1)} = 0.0750 \quad \text{and} \quad \pi_5^{(1)} = 0.3250.$$

We now have

$$\pi^{(1)} = (0.1000,\ 0.0667,\ 0.3000,\ 0.0750,\ 0.3250)$$

Normalize so that the elements sum to 1:

$$\pi^{(1)} = (0.1154,\ 0.0769,\ 0.3462,\ 0.0865,\ 0.3750).$$

Now start the next iteration

— actually, all subsequent iterations yield exactly the same result.

The iteration matrix has one unit eigenvalue and four zero eigenvalues!

If diagonal blocks are large and without structure
— use an iterative method to solve them.

Many inner iterative methods (one per block) within an outer (or global) iteration.

Use the solution computed for D_{ii} at iteration k as the initial approximation at iteration $k+1$.

Do not compute a highly accurate solution in early (outer) iterations.

MATLAB implementation for block iterative methods:

Input: P — a stochastic matrix;
ni — a partitioning vector, $ni(i) =$ length of block i;
$itmax1$ — the number of outer iterations to perform,
$itmax2$ — the number of iterations for inner Gauss-Seidel (if used).

Returns: the solution vector π and a vector of residuals.

It calls the Gauss–Seidel program, Algorithm 65, given previously.

Algorithm: Block Gauss-Seidel for $P^T x = x$

```
function [x,res] = bgs(P,ni,itmax1,itmax2)
[n,n] = size(P); [na,nb] = size(ni);

bl(1) = 1;                                          %  Get beginning and end
for k = 1:nb, bl(k+1) = bl(k)+ni(k); end            %  points of each block
x = ones(n,1)/n;                                    %  Initial approximation

%%%%%%%%%%%%%%%%%%%%%%%%  BEGIN OUTER LOOP  %%%%%%%%%%%%%%%%%%%%%%%%%
for iter = 1:itmax1,
    for m = 1:nb,                                   %  All diagonal blocks
        A = P(bl(m):bl(m+1)-1,bl(m):bl(m+1)-1)';    %  Get A_mm
        b = -P(1:n,bl(m):bl(m+1)-1)'*x+A*x(bl(m):bl(m+1)-1);  %  RHS
        z = inv(A-eye(ni(m)))*b;                    %  Solve for z

%       ***  To solve the blocks using Gauss-Seidel     ***
%       ***  instead of a direct method, substitute     ***
%       ***  the next two lines for the previous one.   ***
%**     x0 = x(bl(m):bl(m+1)-1);                    %  Get starting vector
%**     [z,r] = gs(A-eye(ni(m)),x0,b,itmax2);       %  Solve for z
```

```
x(bl(m):bl(m+1)-1) = z;                          %  Update x
    end
    res(iter) = norm((P'-eye(n))*x,2);            %  Compute residual
end
x = x/norm(x,1); res = res';
```

Intuitively, the larger the blocks (and thus the smaller the number of blocks), the fewer (outer) iterations needed for convergence.

But larger blocks increase the number of operations per iteration.

In some important cases, there is no increase
— e.g.,, when the matrix is block tridiagonal (QBD processes)
and the diagonal blocks are also tridiagonal.

Decomposition and Aggregation Methods

Break the problemn into subproblems that can be solved independently. Get the global solution by "pasting" together the subproblem solutions.

Nearly-completely-decomposable (NCD):
— state space may be partitioned into disjoint subsets with strong interactions among the states of a subset but with weak interactions among the subsets themselves.

Assumption that subsystems are independent and can be solved separately does not hold.

This error will be small if the assumption is approximately true.

An NCD stochastic matrix P:

$$P = \begin{pmatrix} P_{11} & P_{12} & \dots & P_{1N} \\ P_{21} & P_{22} & \dots & P_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ P_{N1} & P_{N2} & \dots & P_{NN} \end{pmatrix},$$

where

$$\begin{aligned} ||P_{ii}|| &= O(1), \qquad i = 1, 2, \dots, N, \quad \text{and} \\ ||P_{ij}|| &= O(\epsilon), \qquad i \neq j, \end{aligned}$$

— possesses many (indeed, N) eigenvalues pathelogically close to 1.

Suppose the off-diagonal blocks are all zero, i.e.,

$$(\pi_1, \pi_2, \ldots, \pi_N) \begin{pmatrix} P_{11} & 0 & \ldots & 0 & 0 \\ 0 & P_{22} & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & P_{N-1N-1} & 0 \\ 0 & 0 & \ldots & 0 & P_{NN} \end{pmatrix} = (\pi_1, \pi_2, \ldots, \pi_N).$$

— each P_{ii} is a stochastic matrix.

There are N distinct ergodic classes

— each π_i can be found directly from

$$\pi_i P_{ii} = \pi_i, \quad i = 1, 2, \ldots, N.$$

Approximate solution procedure:

(a) Solve the diagonal blocks as if they are independent.
— the solution obtained for each block provides an approximation to the probability of being in the different states of that block, conditioned on being in one (any one) of the states of that block.

(b) Estimate the probability of being in each block.
— this will allow us to remove the condition in part (a).

(c) Combine (a) and (b) into a global approximate solution.

(a) Solve blocks as if independent.

The blocks P_{ii} are not stochastic but strictly substochastic.

Use the normalized eigenvector corresponding to the Perron root (the eigenvalue closest to 1) of block P_{ii}

Example: — the 8×8 Courtois matrix.

$$P = \left(\begin{array}{ccc|ccc|cc} .85 & .0 & .149 & .0009 & .0 & .00005 & .0 & .00005 \\ .1 & .65 & .249 & .0 & .0009 & .00005 & .0 & .00005 \\ .1 & .8 & .0996 & .0003 & .0 & .0 & .0001 & .0 \\ \hline .0 & .0004 & .0 & .7 & .2995 & .0 & .0001 & .0 \\ .0005 & .0 & .0004 & .399 & .6 & .0001 & .0 & .0 \\ \hline .0 & .00005 & .0 & .0 & .00005 & .6 & .2499 & .15 \\ .00003 & .0 & .00003 & .00004 & .0 & .1 & .8 & .0999 \\ .0 & .00005 & .0 & .0 & .00005 & .1999 & .25 & .55 \end{array}\right)$$

$$P_{11} = \begin{pmatrix} .85 & .0 & .149 \\ .1 & .65 & .249 \\ .1 & .8 & .0996 \end{pmatrix}; \quad \lambda_{1_1} = .99911; \quad u_1 = (.40143,\ .41672,\ .18185),$$

$$P_{22} = \begin{pmatrix} .7 & .2995 \\ .399 & .6 \end{pmatrix}; \quad \lambda_{2_1} = .99929; \quad u_2 = (.57140,\ .42860),$$

$$P_{33} = \begin{pmatrix} .6 & .2499 & .15 \\ .1 & .8 & .0999 \\ .1999 & .25 & .55 \end{pmatrix}; \quad \lambda_{3_1} = .9999; \quad u_3 = (.24074,\ .55563,\ .20364).$$

To summarize, in part (a), solve the N eigenvalue problems:

$$u_i P_{ii} = \lambda_{i_1} u_i, \qquad u_i e = 1, \quad i = 1, 2, \ldots, N$$

(b) Estimate block probabilities

Concatenating block probabilities does not give a probability vector
— the elements of each subvector sum to one.

Weigh each subvector by the probability of being in that subblock
— the distributions computed from the P_{ii} are conditional probabilities
— we need to remove that condition.

We need the $(N \times N)$ stochastic matrix that characterizes the interactions among blocks
— the *coupling matrix*. Shrink each P_{ij} down to a single element.

In example, find scaling factors $\alpha_1,\ \alpha_2$ and α_3 such that

$$(\alpha_1 u_1,\ \alpha_2 u_2,\ \alpha_3 u_3)$$

is an approximate solution to π.

— α_i is the proportion of time we spend in block i.

In example, shrink the (8×8) matrix to a (3×3) stochastic matrix.

Accomplished in 2 steps:

(1) replace each row of each block by the sum of its 2 elements — mathematically, for each block, form $P_{ij}e$.

The sum of the elements in row k of block P_{ij} is the probability of leaving state k of block i and entering one of the states of block j.

Example: Summing across the block rows of the Courtois matrix:

$$\left(\begin{array}{c|c|c} .999 & .0009 & .0001 \\ .999 & .0009 & .0001 \\ .9996 & .0003 & .0001 \\ \hline .0004 & .9995 & .0001 \\ .0009 & .999 & .0001 \\ \hline .00005 & .00005 & .9999 \\ .00006 & .00004 & .9999 \\ .00005 & .00005 & .9999 \end{array}\right).$$

(2) find the probability of leaving (any state of) block i
to enter (any state of) block j
— i.e., reduce each column subvector, $P_{ij}e$, to a scalar.

Each component of $P_{ij}e$ must be weighed by the probability of being in that state *given* the system is in the corresponding block.

These weighing factors are the components of $\pi_i/||\pi_i||_1$.

Thus, the coupling matrix:

$$(C)_{ij} = \frac{\pi_i}{||\pi_i||_1} P_{ij}e = \phi_i P_{ij}e,$$

If P is irreducible, then C is stochastic and irreducible.

Let $\xi C = \xi$ and $\xi e = 1$. Then $\xi = (||\pi_1||_1,\ ||\pi_2||_1,\ ...,\ ||\pi_N||_1)$.

But π is unknown $\Rightarrow$ impossible to compute the weights $||\pi_i||_1$
— they can be approximated from the eigensolution of the P_{ii}.

Example: The Courtois matrix.

$$\left(\begin{array}{c|c|c} .99911 & .00079 & .00010 \\ \hline .00061 & .99929 & .00010 \\ \hline .00006 & .00004 & .99990 \end{array}\right).$$

with eigenvalues $1.0;\ .9998$; and $.9985$ and its stationary probability vector:

$$\alpha = (.22252,\ .27748,\ .50000).$$

(c) Compute global approximation.

$(\xi_1 u_1,\ \xi_2 u_2,\ ...,\ \xi_N u_N)$

Example:

$$\pi^* = (.08932,\ .09273,\ .04046,\ .15855,\ .11893,\ .12037,\ .27781,\ .10182).$$

Exact solution:

$$\pi = (.08928,\ .09276,\ .04049,\ .15853,\ .11894,\ .12039,\ .27780,\ .10182).$$

Algorithm: NCD Decomposition Approximation.

1. Solve the individual blocks: $u_i P_{ii} = \lambda_{i_1} u_i, \quad u_i e = 1$ for $i = 1, 2, \ldots, N$.
2. (a) Form the coupling matrix: $(C)_{ij} = u_i P_{ij} e$.
   (b) Solve the coupling problem: $\xi = \xi C, \quad \xi e = 1$.
3. Construct the approximate solution: $\pi^* = (\xi_1 u_1, \ \xi_2 u_2, \ ..., \ \xi_N u_N)$.

Now incorporate this approximation into an iterative scheme:
— apply a power step to the computed approximation and begin again.

Instead of a power step, use block Gauss-Seidel (disaggregation step)
— forming and solving the matrix C is an aggregation step.

The entire procedure is called *iterative aggregation/disaggregation*.

Algorithm: Iterative Aggregation/Disaggregation

1. Let $\pi^{(0)} = (\pi_1^{(0)}, \pi_2^{(0)}, ..., \pi_N^{(0)})$ be an initial approximation, and set $m = 1$.
2. Compute $\phi^{(m-1)} = (\phi_1^{(m-1)}, \phi_2^{(m-1)}, ..., \phi_N^{(m-1)})$, where

$$\phi_i^{(m-1)} = \frac{\pi_i^{(m-1)}}{||\pi_i^{(m-1)}||_1;} \quad i = 1,\ 2,\ ...,\ N.$$

3. (a) Form the coupling matrix: $C^{(m-1)})_{ij} = \phi_i^{(m-1)} P_{ij} e$.
   (b) Solve the coupling problem: $\xi^{(m-1)} C^{(m-1)} = \xi^{(m-1)}, \quad \xi^{(m-1)} e = 1.$
4. (a) Construct the row vector

$$z^{(m)} = (\xi_1^{(m-1)} \phi_1^{(m-1)},\ \xi_2^{(m-1)} \phi_2^{(m-1)},\ \ldots,\ \xi_N^{(m-1)} \phi_N^{(m-1)}).$$

   (b) Solve the following N systems of equations to find $\pi^{(m)}$:

$$\pi_k^{(m)} = \pi_k^{(m)} P_{kk} + \sum_{j>k} z_j^{(m)} P_{jk} + \sum_{j<k} \pi_j^{(m)} P_{jk}, \quad k = 1, 2,\ ...,\ N. \quad (36)$$

5. Normalize and conduct a test for convergence.
   If satisfactory, then stop and take $\pi^{(m)}$ to be the required solution vector.
   Otherwise set $m = m + 1$ and go to step 2.

Example:

IAD and BGS Residuals for the Courtois NCD Matrix		
Iteration	IAD Residual	BGS Residual
	$1.0e-05\times$	$1.0e-05\times$
1	0.93581293961421	0.94805408435419
2	0.00052482104506	0.01093707688215
3	0.00000000280606	0.00046904081241
4	0.00000000000498	0.00002012500900
5	0.00000000000412	0.00000086349742
6	0.00000000000351	0.00000003705098
7	0.00000000000397	0.00000000158929
8	0.00000000000529	0.00000000006641
9	0.00000000000408	0.00000000000596
10	0.00000000000379	0.00000000000395

Diagonal block equations are solved using an LU decomposition.

Implementation details.

The critical points are steps 3 and 4(b).

Step 3(a): — compute $P_{ij}e$ only once for each block and store it somewhere for future iterations.

Step 3(b): — compute ξ by any of the previously discussed methods.

Step 4(b): — each of the N systems of equations can be written as $Bx = r$ where $B = (I - P_{kk})^T$ and

$$r = \sum_{j>k} z_j P_{jk} + \sum_{j<k} \pi_j P_{jk},\ k = 1, 2, \ldots, N.$$

In all cases B is nonsingular.

— If a direct method is used, compute the LU decomposition of $(I - P_{kk}),\ k = 1, 2, \ldots, N$ only once.

— If an iterative method (iterations within iterations) perform only a small number of iterations and use the final approximation at one step as the initial approximation for the next.

Example: The Courtois matrix with Gauss-Seidel for the blocks — needs an additional iteration.

IAD: Gauss-Seidel for Block Solutions	
Iteration	Residual: $\hat{\pi}(I-P)$
	$1.0e{-}03\times$
1	0.14117911369086
2	0.00016634452597
3	0.00000017031189
4	0.00000000015278
5	0.00000000000014
6	0.00000000000007
7	0.00000000000006
8	0.00000000000003
9	0.00000000000003
10	0.00000000000006

— but check the inner iterations!

Inner Iteration	Global Iteration 1	2	3	4
1	0.0131607445	0.000009106488	0.00000002197727	0.00000000002345
2	0.0032775892	0.000002280232	0.00000000554827	0.00000000000593
3	0.0008932908	0.000000605958	0.00000000142318	0.00000000000151
4	0.0002001278	0.000000136332	0.00000000034441	0.00000000000037
5	0.0001468896	0.000000077107	0.00000000010961	0.00000000000011
6	0.0001124823	0.000000051518	0.00000000003470	0.00000000000003
7	0.0001178683	0.000000055123	0.00000000003872	0.00000000000002
8	0.0001156634	0.000000053697	0.00000000003543	0.00000000000002
9	0.0001155802	0.000000053752	0.00000000003596	0.00000000000002
10	0.0001149744	0.000000053446	0.00000000003562	0.00000000000002
11	0.0001145044	0.000000053234	0.00000000003552	0.00000000000002
12	0.0001140028	0.000000052999	0.00000000003535	0.00000000000002
13	0.0001135119	0.000000052772	0.00000000003520	0.00000000000002
14	0.0001130210	0.000000052543	0.00000000003505	0.00000000000002

Matlab code for experimenting with IAD methods.

```
function [soln,res] = kms(P,ni,itmax1,itmax2)
[n,n] = size(P); [na,nb] = size(ni);

%%%%%%%%    ITERATIVE AGGREGATION/DISAGGREGATION FOR pi*P = pi   %%%%%%%%%%%%%%%%

bl(1) = 1;                                      %  Get beginning and end
for k = 1:nb, bl(k+1) = bl(k)+ni(k); end        %  points of each block


E = zeros(n,nb);                                %  Form (n x nb) matrix E
next = 0;                                       %  (This is needed in forming
for i = 1:nb,                                   %   the coupling matrix: A )
    for k = 1:ni(i); next = next+1; E(next,i) = 1; end
end
Pije = P*E;                           %  Compute constant part of coupling matrix

Phi = zeros(nb,n);                    %  Phi, used in forming the coupling matrix,
for m = 1:nb,                         %  keeps normalized parts of approximation
    for j = 1:ni(m), Phi(m,bl(m)+j-1) = 1/ni(m); end
end

A = Phi*Pije;                                   %  Form the coupling matrix A
AA = (A-eye(nb))';    en = [zeros(nb-1,1);1];
xi = inv([AA(1:nb-1,1:nb);ones(1,nb)])*en;      %  Solve the coupling matrix
z = Phi'*xi;                                    %  Initial approximation
```

```
%%%%%%%%%%%%%%%%%%%%%%%%  BEGIN OUTER LOOP  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for iter = 1:itmax1,

    for m = 1:nb,                           %  Solve all diag. blocks; Pmm y = b
        Pmm = P(bl(m):bl(m+1)-1,bl(m):bl(m+1)-1);  %  Get coefficient block, Pmm
        b = z(bl(m):bl(m+1)-1)'*Pmm-z'*P(1:n,bl(m):bl(m+1)-1);     %  RHS

        %y = inv(Pmm'-eye(ni(m)))*b';  %  Substitute this line for the 2
                      % lines below it, to solve Pmm by iterative method.
        x0 = z(bl(m):bl(m+1)-1);       %  Get new starting vector
        [y,resid] = mygs(Pmm'-eye(ni(m)),x0,b',itmax2);  % y is soln for block Pmm
        if m==1 resid
        end
        for j = 1:ni(m), z(bl(m)+j-1) = y(j); end %  Update solution vector
        y = y/norm(y,1);                          %  Normalize y



        for j = 1:ni(m),
            Phi(m,bl(m)+j-1) = y(j);              %  Update Phi
        end
    end

    pi = z;
    res(iter) = norm((P'-eye(n))*pi,2);         %  Compute residual
    A = Phi*Pije; AA = (A-eye(nb))';            %  Form the coupling matrix A
    xi = inv([AA(1:nb-1,1:nb);ones(1,nb)])*en;  %  Solve the coupling matrix
    z  = Phi'*xi;                               %  Compute new approximation

end

soln = pi; res = res';
```

For NCD problems, the states must be ordered according to the block structure

— it may be necessary to reorder the states to get this property.

A reordering can be accomplished by treating the Markov chain as a directed graph and utilizing a graph algorithm to find its strongly connected components.

The complexity of this algorithm is $O(|V| + |E|)$, where $|V|$ is the number of vertices and $|E|$ is the number of edges in the graph.