FRTN30 Network Dynamics
Hand-in 2

Due: 2019-05-08

Instructions:

- You may implement your solutions in any language you see fit, but the TAs can only guarantee you support with MATLAB/Octave. Your code should be written in a quite general manner, i.e., if a question is slightly modified, it should only require slight modifications in your code as well. Send a PDF-file with your answers together with your code to `FRTN30@control.lth.se`. As a subject for the mail, you enter `handin2` and your full name.
- Comment your code well. Clarity is more important than efficiency.
- Late submission is discouraged: you get 1 point in your exam (out of 25) for each on-time submission.
- Collaboration policy: collaboration such as exchange of ideas among students is encouraged, however, every student has to submit her/his own manuscript (in pdf format) and code, and specify whom she/he has collaborated with and on what particular part of the work.
- Up to five best hand-ins may be rewarded with an extra point.
- **Visualization** We have provided you with a code skeleton (for MATLAB) of how to visualize a single particle moving around in a network (in a completely deterministic fashion). This might be useful for you as you go about solving the problems in the hand-in, but *it is in no way needed nor compulsory to solve the hand-in.*

Markov chains and networks

Preparation: In order to be prepared for this hand-in assignment, it is recommended to read through chapter 4 (*Markov chains and random walks*) in the lecture notes, as well as solving the exercises from exercise session 6.

1 Single-particle random walk

The first part of this assignment consists of studying a single particle which performs a **random walk** in the network described by the graph in Fig. 1. A random walk means that the particle's movement is determined by stochastic variables. The properties of these variables can be obtained from the associated **transition rate matrix**

$$\Lambda = \begin{array}{c} \begin{array}{ccccc} o & a & b & c & d \end{array} \\ \begin{pmatrix} 0 & 2/5 & 1/5 & 0 & 0 \\ 0 & 0 & 3/4 & 1/4 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 1/3 & 0 & 2/3 \\ 0 & 1/3 & 0 & 1/3 & 0 \end{pmatrix} \end{array} \begin{array}{c} o \\ a \\ b \\ c \\ d \end{array} . \tag{1}$$

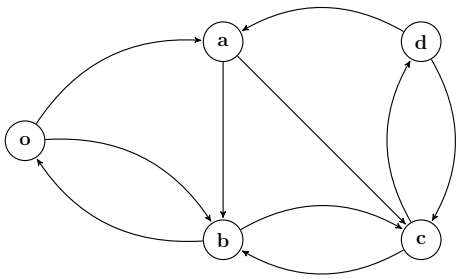

Figure 1: *Closed network in which particles move according to the transition rate matrix* (1).

In this matrix, a large element Λ_{ij} means that, if the particle is in state (node) i, it is probable that it will move to state j within a short time. More specifically, defining $\omega_i = \sum_j \Lambda_{ij}$, the particle will in average stay a time $1/\omega_i$ in state i before moving to another state. After this time, the next state j is chosen with probability proportional to the elements on row i in (1). This means that the probability that the next state is j, is given by the element P_{ij} in the normalized weight matrix $P = D^{-1}\Lambda$, where $D = \mathrm{diag}(\omega)$ and $\omega = \Lambda\mathbb{1}$. The fact that

the probabilities of the next node only depend on the current node is known as the Markov property, and the process is due to this called a **Markov chain**.

The time S during which the particle stays in a state i before moving to its next state is an exponentially distributed stochastic variable according to

$$\mathbb{P}(S \geq t) = e^{-rt}, \quad t \geq 0, \tag{2}$$

where $r = \omega_i = \sum_j \Lambda_{ij}$ is called the **rate** of the distribution. This means that, in general (unless all nodes have the same out-degree ω_i), the waiting time distributions will be different for the different states. In order to describe the waiting times with exponentially distributed variables of the same rate r, the problem can be reformulated in terms of a **transition probability matrix** $\overline{P}$, with elements

$$\overline{P}_{ij} = \frac{\Lambda_{ij}}{\omega_*}, \quad \omega_* = \max_i \omega_i, \tag{3}$$

$$\overline{P}_{ii} = 1 - \sum_{j \neq i} \overline{P}_{ij}. \tag{4}$$

Together with waiting times given by stochastic variables according to (2), with $r = w_*$, the matrix $\overline{P}$ describes the same system as the matrix (1). This works by choosing the rate $r = w_*$ corresponding to the shortest average waiting time among the nodes, and taking the longer waiting times for the remaining nodes into account by adding self-loops (4) on these so that the particle in average remains equally long as in the original formulation before changing node.

In conclusion, the random walk of the particle can be described by one random process which determines a certain waiting time, and one associated matrix which describes what happens when the waiting time has passed. The waiting-time-determining process uses the sequence

$$T_0 = 0, \quad T_k = \sum_{1 \leq i \leq k} S_i, \quad k = 1, 2, ..., \tag{5}$$

where S_i are exponentially distributed variables, according to (2), with rate $r = w_*$. The sequence (5) is called a **Poisson clock**, and its element T_k is the sum of the first k waiting times, determined by a sequence of k exponentially distributed variables S_i. The timing-process, which is a so-called **Poisson process** is then given by

$$N_t = \sup\{k \geq 0 : T_k \leq t\}, \quad t \geq 0, \tag{6}$$

where N_t is the number of "ticks" of the Poisson clock (5) at time t, i.e. the number of waiting times that have passed at time t. Each time a waiting time has passed, i.e. each time N_t increases by 1, the particle then moves from the current state i to a state j (which can be itself, in the case of a self-loop) with the probabilities described by (3) and (4).

The state of the particle at time $t \geq 0$ is given by a continuous-time stochastic process $X(t) = U(N_t)$ $(t \in \mathbb{R}_+)$, where $U(k)$ $(k = 0, 1, 2, \dots)$ is a discrete-time stochastic process

given by the Markov chain corresponding to the matrix $\overline{P}$, defined by (3) and (4). $U(k)$ is known as a **jump chain**, since it describes the sequence of moves when the Poisson clock ticks. The time T_s it takes for the particle to reach a node s is known as the **hitting time**. This is a stochastic variable which depends on which node the particle is in at time $t = 0$. The expectation value of the hitting time of node s when starting at node i is denoted $\mathbb{E}_i(T_s) = \mathbb{E}(T_s \mid X(0) = i)$.

For a *discrete-time* Markov chain, the expected hitting time $\mathbb{E}_i(T_s)$ for node s from node i is given by

$$\mathbb{E}_i(T_s) = 0, \quad \text{if} \quad i = s \tag{7}$$

$$\mathbb{E}_i(T_s) = 1 + \sum_j P_{ij}\mathbb{E}_j(T_s), \quad \text{if} \quad i \neq s. \tag{8}$$

I.e., we have that the hitting time is 0 if the particle is in node s already, or otherwise that we have to wait at least one time unit (for the next step), and then the expected hitting time from the node that is visited next. Thereby, we add to the single time unit the weighted sum of the hitting times from the nodes that could be visited next, weighted by the probabilities that they are chosen in the next step.

In order to use this result for the continuous-time case, we first note that the expected waiting time for a tick of the Poisson clock in the Poisson process (6) is the same all the time, and is given by $\mathbb{E}(S) = 1/r = 1/w_*$, for S in (2). Thus, the expectation value of the time that the jump chain $U(k)$ – corresponding to the matrix $\overline{P}$ – stays in a certain state before changing to the next one (which might be the same) is always the same. Thereby, one can study the hitting time of the jump chain, which is a discrete-time chain, and then multiply the result, i.e. the number of discrete time units, with the expected time between two jumps, in order to get the expected hitting time for the continuous-time process. Note that this means that the normalized probability matrix in (8) should be the transition probability matrix associated with the jump chain $U(k)$, i.e., the matrix $\overline{P}$.

An *alternative* approach for computing the hitting times for the continuous-time Markov chain is to directly use the analogue to the discrete-time equations (7) and (8) for the continuous-time Markov chain, taking into account the different transition rates in the different nodes, and using the normalized probability matrix P given by the transition rate matrix, i.e., $P = \text{diag}(\omega)^{-1}\Lambda$.

The **return time** T_j^+ for a given node j is the time it takes for a particle to first leave the node and then return back to it, i.e., $T_j^+ = \inf\{t \geq 0 : X(t) = j \text{ and } X(s) \neq j \text{ for some } s \in (0, t)\}$.

Your task is to simulate the particle moving around in the network in continuous time according to the transition rate matrix (1). To help you with this we have provided some hints below. You should then use these simulations to answer the following questions:

a) *What is, according to the simulations, the average time it takes a particle that starts in node a to leave the node and then return to it?*

b) *How does the result in a) compare to the theoretical return-time* $\mathbb{E}_a[T_a^+]$*? (Include a description of how this is computed.)*

c) *What is, according to the simulations, the average time it takes to move from node o to node d?*

d) *How does the result in c) compare to the theoretical hitting-time* $\mathbb{E}_o[T_d]$*? (Describe also how this is computed.)*

Hint: To simulate a Poisson process with rate r, one can simulate the time between two ticks of the Poisson clock, which we can denote t_{next}. Then, if we draw a random variable u from a uniform distribution, $u \in \mathcal{U}(0,1)$, we can compute t_{next} as

$$t_{\text{next}} = \frac{-\ln(u)}{r}.$$

It might also be useful to recall that the Poisson process is memoryless:

$$\mathbb{P}(X \geq t+s \mid X \geq t) = \frac{\mathbb{P}(X \geq t+s)}{\mathbb{P}(X \geq t)} = \frac{e^{-r(t+s)}}{e^{-rt}} = e^{-rs} = \mathbb{P}(X \geq s).$$

Furthermore, let's say that you wish to simulate the outcome of when a 6-sided dice is thrown, but where the dice has only 3 numbers $i = 1, 2, 3$, with **probability distribution** $p = (1/6,\ 3/6,\ 2/6)$. The outcome could be simulated by drawing a uniform random variable u in the interval $[0, 1]$. You can then use the **cumulative probability distribution** of u: $\mathbb{F} = (\sum_{i\leq 1} p_i,\ \sum_{i\leq 2} p_i,\ \sum_{i\leq 3} p_i) = (1/6,\ 4/6,\ 6/6)$ and choose the number $i = 1, 2$ or 3 for which $\mathbb{F}_{i-1} < u \leq \mathbb{F}_i$. You can use a similar method but with $p = (P_{i1}, P_{i2}, \ldots, P_{in})$, i.e., row i in the probability matrix P for the discrete-time Markov chain, to randomly select which node the particle should visit next, if the particle currently is in node i.

2 Multi-particle random walk – a matter of perspective

In this part we will again consider the network of Fig. 1, with weights according to (1). However, now we will simulate many particles moving around in the network in continuous time. Each of the particles in the network will move around just as the single particle moved around in Part 1: the time it will stay in a node is exponentially distributed, and on average it will stay $1/w_i$ time-units in a node. The next node it will visit is based on the probability matrix $P = \text{diag}\,(\omega)^{-1}\Lambda$, where $\omega = \Lambda\mathbb{1}$.

Your task is to simulate this system from two different perspectives: the *particle perspective*, i.e. "follow the particle", and the *node perspective*, i.e. "observe from the node".

Simulating the system from a particle perspective is exactly what we did in Part 1, but here you have to follow many particles instead.

To simulate the system from the node perspective you instead have to observe the particles from the node. When doing this you do not have to care about each unique particle, but only about the number of particles in the node. Note that at node i, each particle in that node will stay there on average $1/\omega_i$ time units. Therefore, the node will pass along particles at a rate proportional to the number of particles in the node. In fact, if at time t the number of particles in node i is $n_i(t)$, it will pass along particles at a rate of $n_i(t)\omega_i$. The departure times of the node can thus be seen as a Poisson process with rate $n_i(t)\omega_i$. At each tick of the Poisson clock of the node, it will move a particle to a neighboring node. The node to which the particle will move is again based on the normalized transition rate matrix P.

Simulate the system from the two perspectives, and then answer the following questions (we have again provided a hint below):

a) Particle perspective:

- *If 100 particles all start in node a, what is the average time for a particle to return to node a?*
- *How does this compare to the answer in Part 1, why?*

b) Node perspective:

- *If 100 particles start in node o, and the system is simulated for 60 time units, what is the average number of particles in the different nodes at the end of the simulation?*
- *Illustrate this with a plot showing the number of particles in each node during the simulation time.*

Hint: To simulate the particle perspective we could either attach a Poisson clock to each of the particles and move them individually, just as in Part 1. Alternatively, we could have a single, system-wide Poisson clock with rate 100. Then, at every tick of the system-wide clock we randomly select which particle to move. This particle is then moved to a neighbor node (which can be itself) based on the transition probability matrix $\overline{P}$. This means, that if a particle in node o is selected, there is a probability of $1 - \omega_o = 1 - 3/5 = 2/5$ that it will stay in node o, and a probability of $\omega_o = 3/5$ that it will leave the node.

For the node perspective we could have one Poisson clock attached to each node. But since the rate of this is proportional to the number of particles in the node, it would have to change rate during the simulation. An alternative way is to again have a system-wide Poisson clock with rate 100. Then, at each tick we randomly, and proportionally to the number of particles in the different nodes, select a node from which we should move a particle. Then a particle from the selected node will move according to the transition probability matrix $\overline{P}$.

3 To queue or not to queue? (optional)

In this part we will study how different particles affect each other when moving around in a network in continuous time. We will study the open network of Figure 2, with transition rate matrix Λ_{open} according to (9).

$$\Lambda_{\text{open}} = \begin{array}{c} \begin{matrix} o & a & b & c & d \end{matrix} \\ \begin{pmatrix} 0 & 2/3 & 1/3 & 0 & 0 \\ 0 & 0 & 1/4 & 1/4 & 2/4 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \begin{matrix} o \\ a \\ b \\ c \\ d \end{matrix} \tag{9}$$

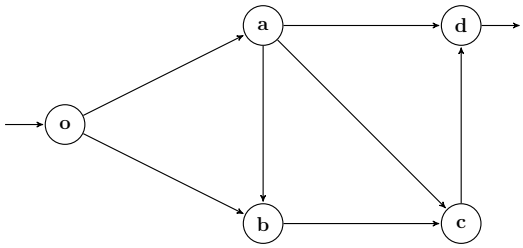

Figure 2: *Open network.*

For this system, particles will enter the system at node o according to a Poisson process with rate $\lambda = 1$. Each node will then pass along a particle according to a given rate, similar to what you did in Part 2 with the "node perspective".

You will simulate two different scenarios that differ by what rate the nodes will pass along particles: i) *proportional rate*, and ii) *fixed rate*. In scenario i), each node will pass along particles according to a Poisson process with rate equal to the number of particles in the node. In scenario ii), each node will instead pass along particles with a fixed rate of 1.

Note that since node d does not have a node to send its particles to, when the Poisson-clock ticks for this node you could simply decrease the number of particles in the node by one (if there are any particles in the node). You could equivalently think of another node d' connected to node d, such that at every tick of the Poisson clock of d, it sends a particle to node d'.

Note that since we do not know how many particles are in the system at any given time, care must be taken when simulating the process.

The goal is to simulate the two systems and answer the following questions:

a) Proportional rate:

The rate of the Poisson clock of each node is equal to the number of particles in it.

- *Simulate the system for 60 time units and plot the number of particles in each node over time.*
- *What is the largest input rate that the system can handle without having a node blow up?*

b) Fixed rate:

The rate of the Poisson clock of each node is fixed, and equal to one.

- *Simulate the system for 60 time units and plot the number of particles in each node over time.*
- *What is the largest input rate that the system can handle without having a node blow up? Why is this different from the other case?*