4 Solving Markov Processes

4.1 Introduction

As mentioned earlier, in this course our primary objective with respect to a Markovian model will be to calculate the probability distribution over the state space of the model as $t \to \infty$. In this note we will discuss how the assumptions we have made about the model make this relatively straightforward to do, using the *global balance equations*. In particular, we show how the generator matrix provides a compact way of expressing these equations. In principle at least, any linear algebra package can then be used to derive the required probability distribution. We will also see how this *steady state probability distribution* may be used, either directly or indirectly, to calculate performance measures. The note concludes with a summary of all the assumptions which we have made.

4.2 Steady state distribution

Performance analysis is usually concerned with the behaviour of systems over an extended period of time. For many of the systems we are interested in this is a reasonable assumption. The running time of the system will be very large relative to the individual inter-event times within the system. From a modelling point of view this is important because we may have to choose an initial state of our model arbitrarily. Considering the long term probability distribution will overcome any bias introduced by the chosen starting state. This will have serious impact on how we use models when we consider simulation models, because each *run* of the model is a particular *sample path* of the model over its state space.

For Markovian models we avoid the problems of bias because we do not consider individual sample paths but the probability distribution over the states of the process which we can deduce knowing the instantaneous transition probabilities between states.

We assume that when the effects of initial bias have been overcome, the model, and the system, will enter into *equilibrium* or *steady state* behaviour. This does not mean that the model has entered one state and no longer makes transitions. It does mean that the model is assumed to exhibit regularity and predictability in its behaviour over the state space. Thus, the probability distribution of the random variable $X(t)$ over the state space S, will be no longer changing.

Let us denote by $\pi_k(t)$ the probability that if we observe the Markov process at time t it will be in state x_k, for $x_k \in S$. Then, after equilibrium has been reached, for all states $x_k \in S$, $\pi_k(t + \tau) = \pi_k(t)$, i.e. the time at which we observe the model does not influence the probability that it is in a particular state. Therefore, we denote the *steady state probabilities* without the time variable, e.g. $\boldsymbol{\pi_k}$, for the probability that the model is in state x_k.

The steady state distribution is useful to us because the following theorem tells us that, for the types of model we are considering, the steady state distribution is exactly the same as the long term probability distribution over states.

Theorem: A steady state probability distribution, $\{\boldsymbol{\pi_k}, x_k \in S\}$, exists for every time homogeneous, finite, irreducible Markov process. Moreover, this distribution is the same as the limiting distribution

$$\boxed{\lim_{t\to\infty} \Pr(X(t) = x_k \mid X(0) = x_0) = \boldsymbol{\pi_k}}$$

4.2.1 Global Balance Equations

In steady state $\boldsymbol{\pi_i}$ is the proportion of time that the process spends in state x_i. Recall that the transition "rate" q_{ij} is the instantaneous probability that the model makes a transition from state x_i to state x_j. Thus, in an instant of time, the probability that a transition will occur from state x_i to state x_j is the probability that the model was in state x_i, $\boldsymbol{\pi_i}$, multiplied by the transition probability q_{ij}. This is called the *probability flux* from state x_i to state x_j.

At steady state, in order to maintain the equilibrium, we must assume that the total probability flux out of a state is equal to the total probability flux into the state. If this were not the case the distribution over states would changed. So, for any particular state x_i this implies

$$\boxed{\underbrace{\boldsymbol{\pi_i} \times \sum_{x_j \in S, j \neq i} q_{ij}}_{\text{flux } \textbf{out} \text{ of } x_i} = \underbrace{\sum_{x_j \in S, j \neq i} (\boldsymbol{\pi_j} \times q_{ji})}_{\text{flux } \textbf{into } x_i}}$$

The collection of these equations for all states $x_i \in S$ is called the *global balance equations*. If we recall that the diagonal elements of the infinitesimal generator matrix $\mathbf{Q}$ are the negative sum of the other elements in the row, i.e.

$$q_{ii} = -\sum_{x_j \in S, j \neq i} q_{ij}$$

we can rearrange the equation above to be:

$$\sum_{x_j \in S} \boldsymbol{\pi_j} q_{ji} = 0.$$

Expressing the values $\boldsymbol{\pi_i}$ as a row vector $\boldsymbol{\pi}$, we can write this as a matrix equation:

$$\boxed{\boldsymbol{\pi}\ \mathbf{Q} = 0}$$

The $\boldsymbol{\pi_i}$ are unknown; they the values we wish to find since they are the probability distribution. If there are n states in the state space there are n such equations in n unknowns. Unfortunately, this collection of equations is irreducible: we need another equation in order to solve the equations and find the unknowns. Fortunately, since $\{\boldsymbol{\pi_i}\}$ is a probability distribution we also know that

$$\boxed{\sum_{x_i \in S} \boldsymbol{\pi_i} = 1}$$

This is called that *normalisation condition*. With these $n + 1$ equations we can solve to find the n unknowns, $\boldsymbol{\pi_i}$.

Example: Consider a two-processor system consisting of two CPUs each with its own private memory, and one common memory which can be accessed only by one processor at a time. The two CPUs execute in their private memories for a random time before issuing a common memory access request. Assume that this random time is exponentially distributed with parameter λ, so that the average time spent by each CPU executing in private memory between two common memory access requests is $1/\lambda$. The common memory access duration is also assumed to be an exponentially distributed random variable, with parameter μ, so that the average duration of a common memory access is $1/\mu$.

If, initially, we consider only one processor the system has only two states:

1. the processor is executing in its private memory;
2. The processor is accessing common memory.

The system behaviour can be modelled by a 2 state Markov process whose state transition diagram and generator matrix are as shown below:

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

If we consider the probability flux in and out of state 1 we obtain: $\boldsymbol{\pi_1}\ \lambda = \boldsymbol{\pi_2}\mu$

Similarly, for state 2 [1]: $\boldsymbol{\pi_2}\ \mu = \boldsymbol{\pi_1}\lambda$

Finally, we know from the normalisation condition that: $\boldsymbol{\pi_1} + \boldsymbol{\pi_2} = 1$

It follows that the steady state probability distribution is

$$\boldsymbol{\pi} = \left(\frac{\mu}{\mu + \lambda}, \frac{\lambda}{\mu + \lambda} \right)$$

From this information we can deduce, for example that the probability that the processor is executing in its private memory is $\mu/(\mu + \lambda)$.

4.2.2 Solving the global balance equations

Once we are considering models which have more than a few of states it is unlikely that we want to solve the equations ourselves by hand. Instead we take advantage of the matrix equation formulation of the global balance equations and use suitable software packages. Any linear algebra package will solve matrix equations of the form

$$\mathbf{Ax} = \mathbf{b}$$

where $\mathbf{A}$ is an $n \times n$ matrix, $\mathbf{x}$ is a column vector of n unknowns, and $\mathbf{b}$ is a column vector of n values. There are two problems which have to be resolved before we can take advantage of such software:

[1] Note that in this case the two equations are in fact the same. There will always be a redundant equation in the global balance equations which is why we need the additional information of the normalisation condition.

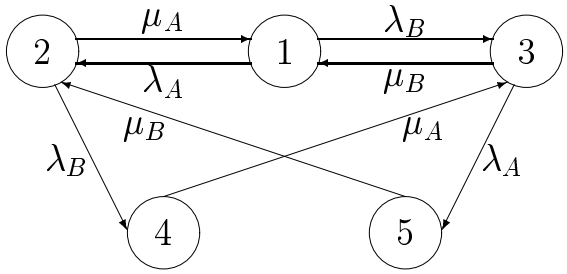
1. Our global balance equation comes in the "wrong" form $\boldsymbol{\pi}\ \mathbf{Q} = 0$: the unknowns, $\boldsymbol{\pi}$ are given as a row vector not a column vector. This problem is resolved by transposing[2] the equation, i.e. $\mathbf{Q}^T\boldsymbol{\pi} = 0$, where the right hand side is now a column vector of zeros, rather than a row vector.
2. The second problem is the redundancy amongst the global balance equations—even after transposing the equation there is insufficient information to solve for the unknowns. As already noted, we need the additional information of the normalisation condition. This problem is resolved by replacing one of the global balance equation by the normalisation condition. In terms of the transposed matrix this corresponds to replacing one row by a row of 1's. We usually choose the last row. We denote this modified matrix $\mathbf{Q}_n^T$. Similarly we change the column "solution" vector, which was all zeros to be a column vector with 1 in the last row, and zeros everywhere else. We denote this vector, $\mathbf{e}_n$.

Now we can use any linear algebra solution package, such as `maple` or `xmaple` to solve the resulting equation:

$$\mathbf{Q}_n^T\boldsymbol{\pi} = \mathbf{e}_n$$

Example: Consider the two-processor system introduced in the previous section. We now consider the system with both processors operational, denoting the processors as A and B, respectively. We assume that the processors have different timing characteristics, the private memory access of A being governed by an exponential distribution with parameter λ_A, the common memory access of B being governed by an exponential distribution with parameter μ_B, etc. Now the state space becomes:

1. A and B both executing in their private memories;
2. B executing in private memory, and A accessing common memory;
3. A executing in private memory, and B accessing common memory;
4. A accessing common memory, B waiting for common memory;
5. B accessing common memory, A waiting for common memory;



$$\mathbf{Q} = \begin{pmatrix} -(\lambda_A+\lambda_B) & \lambda_A & \lambda_B & 0 & 0 \\ \mu_A & -(\mu_A+\lambda_B) & 0 & \lambda_B & 0 \\ \mu_B & 0 & -(\mu_B+\lambda_A) & 0 & \lambda_A \\ 0 & 0 & \mu_A & -\mu_A & 0 \\ 0 & \mu_B & 0 & 0 & -\mu_B \end{pmatrix}$$

[2] i.e. rearranging the matrix so that the rows are columns and vice versa.

This matrix is still small enough to be solved by hand but for the purposes of illustration we solve it using `maple`. The transcript of the `maple` session is given at the end of this note. After appropriate transformations the modified generator matrix is as follows:

$$\mathbf{Q}_n^T = \begin{pmatrix} -(\lambda_A + \lambda_B) & \mu_A & \mu_B & 0 & 0 \\ \lambda_A & -(\mu_A + \lambda_B) & 0 & 0 & \mu_B \\ \lambda_B & 0 & -(\mu_B + \lambda_A) & \mu_A & 0 \\ 0 & \lambda_B & 0 & -\mu_A & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

If we choose the following values for the parameters:

$$\lambda_A = 0.05 \qquad \lambda_B := 0.1 \qquad \mu_A = 0.02 \qquad \mu_B = 0.05$$

solving the matrix equation, and rounding figures to 4 significant figures, we obtain:

$$\boldsymbol{\pi} = (0.0693, 0.0990, 0.1683, 0.4951, 0.1683)$$

4.3 Derivation of Performance Measures

Roughly speaking, there are three ways in which performance measures can be derived from the steady state distribution of a Markov process. These different methods can be thought of as corresponding to different types of measure:

- state-based measures, e.g. utilisation;
- rate-based measures, e.g. throughput;
- other measures which fall outside the above categories, e.g. response time.

State-based measures are those which clearly correspond to the probability that the model is in a state, or a subset of states, which satisfy some condition. For example, utilisation will correspond to those states where a resource is in use. If we consider the example from the previous section, the utilisation of the common memory, U_{mem}, is the total probability that the model is in one of the states in which the common memory is in use:

$$U_{mem} = \boldsymbol{\pi_2} + \boldsymbol{\pi_3} + \boldsymbol{\pi_4} + \boldsymbol{\pi_5} = 93.07\%$$

Other examples of state-based measures are idle time, or the number of jobs in a system. Some measures such as the number of jobs will involve a weighted sum of steady state probabilities, weighted by the appropriate value. For example, we consider jobs waiting for the common memory to queued in that subsystem, then the average number of jobs in the common memory, N_{mem}, is:

$$N_{mem} = (1 \times \boldsymbol{\pi_2}) + (1 \times \boldsymbol{\pi_3}) + (2 \times \boldsymbol{\pi_2}) + (2 \times \boldsymbol{\pi_2}) = 1.594$$

Rate-based measures are those which correspond to the predicted rate at which some event occurs. This will be the product of the rate of the event, and the probability that

the event is enabled, i.e. the probability of being in one of the states from which the event can occur. Thus, if we wish to calculate the throughput of the common memory, the average number of accesses from either processor which it processes in unit time, X_{mem}, it will be

$$X_{mem} = (\mu_A \times (\boldsymbol{\pi_2} + \boldsymbol{\pi_4})) + (\mu_B \times (\boldsymbol{\pi_3} + \boldsymbol{\pi_5})) = 0.0287$$

or, approximately one access every 35 milliseconds.

Finally, the other category of measures are those which do not readily fit into either of the above categories. In these cases, we usually use one of the operational laws to derive the information we need, based on values that we have obtained by other means. Thus, applying Little's law to the common memory we see that

$$W_{mem} = N_{mem}/X_{mem} = 1.594/0.0287 = 55.54 \text{ milliseconds}$$

4.4 Assumptions

The most fundamental assumption we have made is the **stochastic hypothesis**: *"The behaviour of a real system during a given period of time is characterised by the probability distributions of a stochastic process."* Although it can never be conclusively proved, evidence suggests that this hypothesis is true.

In contrast, the assumption that all delays and inter-event times within the model are exponentially distributed, is unlikely to fit with observations of real systems. We make this assumption nevertheless because of the nice mathematical properties of the exponential distribution, and because the exponential distribution is the only one we can use if we wish to assume that the stochastic process characterising our system is a Markov process. The Markov property says that the future behaviour of the system is only dependent on its current state, not on its past history. This seems a reasonable assumption for computer and communication systems, if we choose our states carefully.

We assume that the Markov process has various properties in order to ensure that a steady state distribution exists, and that it coincides with the long term probability distribution. These properties are that the model is finite, time homogeneous and irreducible.

4.5 Transcript of maple session

```
> with(linalg):
>
> lambdaA := 0.05;
> muA := 0.02;
> lambdaB := 0.1;
> muB := 0.05
>
> size := 5;
>
> Q := array(sparse,1..size,1..size);
> e := array(sparse,1..size);
>
> Q[1,2] := lambdaA;
>
> Q[1,3] := lambdaB;
>
> Q[2,1] := muA;
>
> Q[2,4] := lambdaB;
>
> Q[3,1] := muB;
>
> Q[3,5] := lambdaA;
>
> Q[4,3] := muA;
>
> Q[5,2] := muB;
>
> for i to size do
>     s := 0:
>     for j to size do
>         s := s + Q[i,j]
>     od;
>     Q[i,i] := -s
> od;
>
> Qt := transpose(Q);
>
> for i to size do Qt[size,i] := 1 od;
>
> e[size] := 1;
>
> p := linsolve(Qt,e);
```