Algorithms for Analysing Matrix-Analytic Models

Peter Taylor

Department of Mathematics and Statistics, The University of Melbourne
Outline

1. To know $G$ is to know all
2. A simple procedure
3. A switch to discrete time
4. Computations and interpretations
5. The LR Algorithm
6. Acceleration
To know $G$ is to know all

Recall that $\pi_n = \pi_0 R^n$ where

$$\pi_n = (\pi_{n,1}, \pi_{n,2}, \ldots, \pi_{n,M}).$$

Also, $\pi_0(B + RQ_2) = 0$, $\pi_0(I - R)^{-1}e = 1$.

So, to know $R$ is to know the whole distribution.

How do we compute it?
To know $G$ is to know all

$R$ is a solution of

$$Q_0 + RQ_1 + R^2Q_2 = 0.$$ 

Also, $R = Q_0N$, where $N_{ij}$ is the expected time spent in $(1, j)$, starting from $(1, i)$, before the QBD hits level 0 again.

Define $V_{ij} = (Q_1)_{ij} + \sum_k (Q_0)_{ik}G_{kj}$. Then $V$ is the generator of the transient process censored at level 1 under taboo of level 0 and $N = (-V)^{-1}$, so that

$$R = Q_0(-Q_1 - Q_0G)^{-1}$$

and to know $G$ is to know $R$. 
A simple procedure

$G$ is a solution of $Q_2 + Q_1 G + Q_0 G^2 = 0$. How do we solve this?

An obvious start is to transform it into a fixed-point equation:

\[
-Q_1 G = Q_2 + Q_0 G^2
\]

\[
G = (-Q_1)^{-1} Q_2 + (-Q_1)^{-1} Q_0 G^2
\]

\[
= C_2 + C_0 G^2,
\]

where

\[
C_0 = (-Q_1)^{-1} Q_0
\]

\[
C_2 = (-Q_1)^{-1} Q_2,
\]

et voilà!
A simple procedure

The iterative procedure

\[ K_0 = 0, \]
\[ K_{n+1} = C_2 + C_0(K_n)^2 \]

should give us something.

Questions are Does it? and, if so, What?

Before thinking about those questions, let’s look at \( C_0 \) and \( C_2 \).
A simple procedure

\[ C_0 = (-Q_1)^{-1}Q_0 \text{ and } C_2 = (-Q_1)^{-1}Q_2. \]

Assuming \( X(0) = 1, \varphi(0) = i \), define \( \tau \) as the first passage time out of level 1, then \( (C_0)_{ij} = P[X(\tau) = 0, \varphi(\tau) = j] \) and \( (C_2)_{ij} = P[X(\tau) = 2, \varphi(\tau) = j] \) and the equation \( G = C_2 + C_0G^2 \) actually is about a discrete-time QBD.
A switch to discrete time

Here, $C_0$, $C_1$ and $C_2$ are all nonnegative, and $C_0 + C_1 + C_2$ is stochastic.
A switch to discrete time

We still have \( \pi_n = \pi_0 R^n \) for \( n \geq 0 \), and now
\[
R = C_0(I - C_1 - C_0G)^{-1}
\]
so that everything still depends on \( G \).

The equation for \( G \) is \( G = C_2 + C_1G + C_0G^2 \).

If we define \( K_0 = 0 \) and \( K_{n+1} = C_2 + C_1K_n + C_0K_n^2 \), it is easy to see that \( K_0 \leq K_1 \leq K_2 \leq \ldots \leq K_n \leq \ldots \leq G \), so that
\[
\lim_{n \to \infty} K_n = K \leq G.
\]

We can prove that \( K \geq G \) (which implies that \( K = G \)) but it is messy, so we’ll take another tack.
A switch to discrete time

Observe that \( G = P(1 \rightsquigarrow 0) = P(2 \rightsquigarrow 1) = P(3 \rightsquigarrow 2) = \cdots \)

Thus

\[
G = C_2 + C_1G + C_0G^2 = C_2 + (C_1 + C_0G)G \\
= C_2 + UG
\]

where

\[
U = C_1 + C_0G = C_1 + C_0P(2 \rightsquigarrow 1) = P(1 \rightsquigarrow 1 \text{ [avoiding level 0]})
\]

With this,

\[
G = C_2 + UG = C_2 + UC_2 + U^2G = C_2 + UC_2 + U^2C_2 + \cdots \\
= (I - U)^{-1}C_2.
\]
If we take the fixed-point equation

\[(G, U) = ((I - U)^{-1}C_2, C_1 + C_0G)\]

and iterate, we get

\[G_1 = 0,\]
\[U_n = C_1 + C_0 G_{n-1},\]
\[G_n = (I - U_n)^{-1}C_2.\]

We claim

\[G_n = P(1 \rightsquigarrow 0 \text{ [avoiding level } n\text{]}),\]
\[U_n = P(1 \rightsquigarrow 1 \text{ [avoiding levels 0 and } n\text{]}).\]
Proof: We use mathematical induction. For the inductive step we have

\[ U_n = C_1 + C_0 P(1 \rightsquigarrow 0 \text{ [avoiding level 0 and } n - 1]) \]
\[ = C_1 + C_0 P(2 \rightsquigarrow 1 \text{ [avoiding level 0 and } n]). \]

Also

\[ G_n = C_2 + P(1 \rightsquigarrow 1 \text{ [avoiding level 0 and } n])C_2 \]
\[ + P(1 \rightsquigarrow 1 \text{ [avoiding level 0 and } n])^2C_2 + \cdots \]
\[ = C_2 + U_nC_2 + U_n^2C_2 + \cdots \]
Obviously, $G_n \uparrow G$ and $U_n \uparrow U$.

In addition, $G$ and $U$ are minimal nonnegative solutions to their equations.

If drift is negative, iterate until $G_n$ is a stochastic matrix; if drift is positive ...
The LR Algorithm

Since

\[ G = C_2 + C_1 G + C_0 G^2 \]

we know that

\[ (I - C_1)G = C_2 + C_0 G^2 \]

and

\[
G = (I - C_1)^{-1} C_2 + (I - C_1)^{-1} C_0 G^2 \\
= P(1 \leadsto 0 \text{ [avoiding level 2]}) + P(1 \leadsto 2 \text{ [avoiding level 0]})P(2 \leadsto 0) \\
= P(1 \leadsto 0[2]) + P(1 \leadsto 2[0])P(2 \leadsto 0).
\]
The LR Algorithm

To compute $P(2 \rightsquigarrow 0)$, we look at the process

\[
P(2 \rightsquigarrow 0) = (I - C_1')^{-1}C_2' + (I - C_1')^{-1}C_0'P(2 \rightsquigarrow 0)^2
= P(2 \rightsquigarrow 0[4]) + P(2 \rightsquigarrow 4[0])P(4 \rightsquigarrow 0).
\]
The LR Algorithm

The upshot is that

\[ G = P(1 \sim 0[2]) + P(1 \sim 2[0])P(2 \sim 0[4]) \]
\[ + P(1 \sim 2[0])P(2 \sim 4[0])P(4 \sim 0) \]

and we need now only look at levels which are multiples of 4.
The LR Algorithm

By repeating this ad infinitum, we find that

\[ G = P(1 \rightsquigarrow 0[2]) + P(1 \rightsquigarrow 2[0])P(2 \rightsquigarrow 0[4]) + P(1 \rightsquigarrow 2[0])P(2 \rightsquigarrow 4[0])P(4 \rightsquigarrow 0[8]) + P(1 \rightsquigarrow 2[0])P(2 \rightsquigarrow 4[0])P(4 \rightsquigarrow 8[0])P(8 \rightsquigarrow 0[16]) + \cdots \]

where each factor is easily computed.

The sum of the first \( k \) terms is \( P(1 \rightsquigarrow 0[2^k+1]) \) and is equal to \( G_{2^{k+1}} \) of the earlier sequence. Hence the Logarithmic Reduction name given to the resulting algorithm.
Acceleration

In the first algorithm, instead of

\[
\begin{align*}
G_1 & = 0, \\
U_n & = C_1 + C_0 \, G_{n-1} \\
G_n & = (I - U_n)^{-1},
\end{align*}
\]

we could compute

\[
\begin{align*}
G_1 & = I, \\
U_n & = C_1 + C_0 \, G_{n-1} \\
G_n & = (I - U_n)^{-1}.
\end{align*}
\]

This usually gives a good speed-up when the QBD is positive-recurrent.
Define $\xi_i$ to be the roots of $\det(C_2 + zC_1 + z^2C_0)$, with $|\xi_i| \leq |\xi_{i+1}|$. One has

$$\cdots \leq |\xi_{M-1}| < \xi_M = 1 \leq |\xi_{M+1}| \leq \cdots$$

and the rate of convergence of all the procedures is determined by the ratio $\rho = |\xi_M|/|\xi_{M+1}|$ (slow if $|\xi_{M+1}|$ is close to one).

Take any vector $u \geq 0$ with $ue = 1$; replace $C_2$ by $\tilde{C}_2 = C_2 - e \cdot u$; apply LR algorithm and obtain $\tilde{G}$. The required solution is $\tilde{G} + e \cdot u$ and the convergence rate is $\tilde{\rho} = |\xi_{M-1}|/|\xi_{M+1}| < \rho$.

Cost: one destroys the non-negativity of computations.
The present game

There are still useful things to do at the intersection of probability and numerical analysis. The matrix-analytic approach has branched to other processes, which have in turn brought us into contact with other nonlinear equations (Riccati), which might open more vistas.