The MATLAB toolbox SMCSolver for matrix-analytic methods

D. Bini*, B. Meini*, S. Steffe*, J.F. Pérez†, B. Van Houdt‡

* Dipartimento di Matematica, Universita' di Pisa, Italy

† Department of Electrical and Electronics Engineering, Universidad de los Andes, Bogotá, Colombia

‡ Department of Mathematics and Computer Science, University of Antwerp. Belgium

Discrete/Continuous time QBD

QBD transition matrix P or rate matrix Q

$$P \text{ or } Q = \begin{bmatrix} B_1 & A_2 & & & 0 \\ B_0 & A_1 & A_2 & & \\ & A_0 & A_1 & A_2 & \\ & & A_0 & A_1 & \ddots \\ 0 & & & \ddots & \ddots \end{bmatrix}$$

Iterative algorithms for QBDs

Algorithms for R and G (all implemented in SMCSolver in Matlab)

- Functional iterations (FI), (Neuts, Latouche)
- Logarithmic Reduction (LR), (Latouche, Ramaswami)
- Newton Iteration (NI), (Ramaswami, Latouche)
- Cyclic Reduction (CR), (Bini, Meini)
- Invariant Subspace (IS), (Akar, Sohraby)

Using the QBD routines of the SMCSolver (Matlab)

Without options

- Formulate your problem as a QBD, meaning determine A_0, A_1, A_2 and the boundary matrices B_0 and B_1
- Select your favorite algorithm, say CR and call

```
[G,R]=QBD_CR(A0,A1,A2);
```

- To obtain the steady state π use

```
pi=QBD_pi(B0,B1,R);
```

this will compute π such that $\pi e > 1 - 10^{-10}$

RSITÀ DI PISA

Universidad de los Andes

Universiteit Antwerpen

Using the QBD routines of the SMCSolver (Matlab)

QBD_pi options

- MaxNumComp: Maximum number of components of π (default: 500)
- Verbose: The accumulated probability mass is printed every n steps when set to n (default:0)
- Boundary: Allows solving the QBD with a more general boundary

$$P = \begin{bmatrix} B_1 & B_2 & & 0 \\ B_0 & A_1 & A_2 & \\ & A_0 & A_1 & \ddots \\ 0 & & \ddots & \ddots \end{bmatrix}$$

- QBD_pi(B0,B1,R,'Verbose',50,'Boundary',[B2; A1+R*A0]);

Using the QBD routines of the SMCSolver (Matlab)

QBD_CR options

- MaxNumIt: Maximum number of iterations (default: 50)
- Verbose: The residual error is printed at each step when set to 1, (default:0)
- Mode: 'Basic' uses the Basic Cyclic Reduction, 'Shift' uses the shift technique to accelarate convergence (default: 'Shift')

Shift Technique (on zero in 1)

- Accelerates the speed of convergence from $\eta = sp(R)$ without shift, to η/ξ with shift, where
  $\xi = \min\{z \| |z| > 1, det(A_2 + A_1 z + A_0 z^2 - zI) = 0\}$

Continuous time QBD

QBD rate matrix Q

- Rate matrix Q has the same form as P
- Automatically transformed into discrete-time QBD by SMCSolver using a uniformization argument
  - Define $\lambda = \max\{\max_j\{-(A_1)_{j,j}\}, \max_j\{-(B_1)_{j,j}\}\}$,
  - $\bar{B}_1 = B_1/\lambda - I$ and $\bar{B}_0 = B_0/\lambda$,
  - $\bar{A}_1 = A_1/\lambda - I$, $\bar{A}_0 = A_0/\lambda$ and $\bar{A}_2 = A_2/\lambda$
- Both chains have the same R and G matrix and the same steady state vector π

Discrete time M/G/1-type Markov chains

M/G/1-type transition matrix P

- QBD is skip-free in both directions, M/G/1-type is skip-free to the left

$$P = \begin{bmatrix} B_0 & B_1 & B_2 & B_3 & \dots \\ A_0 & A_1 & A_2 & A_3 & \dots \\ & A_0 & A_1 & A_2 & \ddots \\ & & A_0 & A_1 & \ddots \\ 0 & & & \ddots & \ddots \end{bmatrix}$$

Discrete time M/G/1-type: Main results

Key Equation

- Smallest nonnegative solution to nonlinear matrix equation

$$G = \sum_{i=0}^{\infty} A_i G^i$$

- G has the same probabilistic interpretation as with the QBD
- Algorithms for G (implemented in SMCSolver in Matlab)
  - Functional iterations (FI), (Neuts, Latouche)
  - Newton Iteration (NI), (Perez, Telek, Van Houdt)
  - Cyclic Reduction (CR), (Bini, Meini)
  - Invariant Subspace (IS), (Akar, Sohraby)
  - Ramaswami Reduction (RR), (Bini, Meini, Ramaswami)

Discrete time M/G/1-type: Main results

Positive recurrence and stationary vector π

- Markov chain is positive recurrent if and only if $\theta \sum_{i=1}^{\infty} iA_i e < 1$ (and $\sum_i iB_i e < \infty$), with $\theta A = \theta$ and $A = \sum_{i=0}^{\infty} A_i$, which implies G is stochastic
- Stationary vector $\pi = (\pi_0, \pi_1, \ldots)$ obeys Ramaswami's formula

$$\pi_i = \left[\pi_0 \bar{B}_i + \sum_{j=1}^{i-1} \pi_j \bar{A}_{i+1-j} \right] (I - \bar{A}_1)^{-1}$$

for $i > 0$, with $\bar{A}_i = \sum_{j \geq i} A_j G^{j-i}$ and $\bar{B}_i = \sum_{j \geq i} B_j G^{j-i}$

- Computing π_0 is somewhat more involved

Using the M/G/1 routines of the SMCSolver (Matlab)

Without options

- Formulate your problem as an M/G/1-type MC, meaning determine A_i, for $i \geq 0$ and the boundary matrices B_i, for $i \geq 0$
- Select your favorite algorithm, say CR and call

$$\text{G=MG1\_CR(A);}$$

where $A = [A0\ A1\ A2\ A3\ \ldots\ AN]$

- To obtain the steady state π use

$$\text{pi=MG1\_pi(B,A,G);}$$

this will compute π such that $\pi e > 1 - 10^{-10}$

RSITÀ DI PISA

versidad os Andes

Universiteit Antwerpen

Using the M/G/1-type routines of the SMCSolver (Matlab)

MG1_pi options

- MaxNumComp: Max. number of comp. of π (default: 500)
- Verbose: The accumulated probability mass is printed every n steps when set to n (default:0)
- Boundary: Allows solving the M/G/1-type MC with a more general boundary

$$P = \begin{bmatrix} B_0 & B_1 & B_2 & B_3 & \dots \\ C_0 & A_1 & A_2 & A_3 & \dots \\ & A_0 & A_1 & A_2 & \ddots \\ & & A_0 & A_1 & \ddots \\ 0 & & & \ddots & \ddots \end{bmatrix}$$

- MG1_pi(B,A,G,'Verbose',50,'Boundary',C0);

Using the M/G/1-type routines of the SMCSolver (Matlab)

MG1_pi options

- Special boundary: You can solve the M/G/1-type MC with $B_i = A_i$ for all $i \geq 0$

$$P = \begin{bmatrix} A_0 & A_1 & A_2 & A_3 & \cdots \\ A_0 & A_1 & A_2 & A_3 & \cdots \\ & A_0 & A_1 & A_2 & \ddots \\ & & A_0 & A_1 & \ddots \\ 0 & & & \ddots & \ddots \end{bmatrix}$$

  by calling MG1_pi([],A,G);

- If π_0 is known (due to the application at hand), you can pass it via the InputPiZero option, such that it is not computed using the general procedure MG1_pi(B,A,G,'InputPiZero',pi0);

Using the M/G/1 routines of the SMCSolver (Matlab)

MG1_CR options

- MaxNumIt, MaxNumRoot, EpsilonValue (stopping criteria) and Verbose
- Mode: 'PWCR' uses the point-wise Cyclic Reduction, 'ShiftPWCR' uses the shift technique to accelarate convergence (default: 'ShiftPWCR')
- ShiftType:
  - one: moves the zero in $z = 1$ (to zero),
  - tau: moves the zero in $\xi = \min\{z||z| > 1, det(A(z) - zI) = 0\}$ (to infinity) for a positive recurrent MC and $\eta = \max\{z||z| < 1, det(A(z) - zI) = 0\}$ (to zero) for a transient chain
  - dbl: moves both these zeros

RSITÀ DI PISA

Universidad de los Andes

Universiteit Antwerpen

Using the M/G/1 routines of the SMCSolver (Matlab)

MG1_FI options

- MaxNumIt, Verbose and ShiftType
- Mode: 6 modes (3 with and 3 without shifting), default U-Based (Latouche algorithm)
- NonZeroBlocks: Useful when only a few A_i blocks are nonzero and m is large

  MG1_FI(A,'NonZeroBlocks',vec);

  A=[Av1 Av2 ... Avk] and vec=[v1 v2 ... vk] (default: vec=[1 ... N])
- StartValue: Use a different matrix as $G[0]$ (default: $G[0] = 0$), e.g., $G[0] = I$

Using the M/G/1 routines of the SMCSolver (Matlab)

MG1_NI options

- MaxNumIt, Verbose, ShiftType
- Mode: 6 possible modes (default: 'RealSchurShift')
  - DirectSum (+ shifting)
  - RealSchur (+ shifting)
  - ComplexSchur (+ shifting)

UNIVERSITÀ DI PISA

Universidad de los Andes

Universiteit Antwerpen

Using the M/G/1 routines of the SMCSolver (Matlab)

MG1_NI_LRA0

- If A0 has rank $r < m$ call

  MG1_NI_LRA0(A, A0hat, Gamma);

  where A0 = A0hat*Gamma and A=[A1 A2 ... AN]

MG1_NI_LRAi

- If [A1 A2 ... AN] has rank $r < m$ call

  MG1_NI_LRAi(A0, Ahat, Gamma);

  where Ai = Gamma*Aihat, for $i > 0$, and Ahat=[A1hat A2hat ... ANhat]

UNIVERSITÀ DI PISA

Universidad de los Andes

Universiteit Antwerpen

Using the M/G/1 routines of the SMCSolver (Matlab)

MG1_RR options

- MaxNumIt, Verbose
- Mode: 3 modes (default: 'DispStruct')
  - 'Direct' does not rely on the displacement structure, Memory $O(m^2N^2)$, Time $O(m^3N^3)$ per iteration
  - 'DispStruct' makes use of the displacement structure, Memory $O(m^2N)$, Time $O(m^3N^2)$ per iteration
  - 'DispStructFFT' uses the displacement structure and FFTs, Memory $O(m^2N)$, Time $O(m^2NlogN + m^3N)$ per iteration

UNIVERSITÀ DI PISA

Universidad de los Andes

Universiteit Antwerpen

Discrete time GI/M/1-type Markov chains

GI/M/1-type transition matrix P

- QBD is skip-free in both directions, GI/M/1-type is skip-free to the right

$$P = \begin{bmatrix} B_1 & A_0 & & & 0 \\ B_2 & A_1 & A_0 & & \\ B_3 & A_2 & A_1 & A_0 & \\ B_4 & A_3 & A_2 & A_1 & \ddots \\ \vdots & \vdots & \ddots & \ddots & \ddots \end{bmatrix}$$

Discrete time GI/M/1-type MC: Main results

Key Equation

- Smallest nonnegative solution to nonlinear matrix equation

$$R = \sum_{i=0}^{\infty} R^i A_i$$

  R has the same probabilistic interpretation as for the QBD
- To compute R we make use of the (Ramaswami or Bright) dual (in SMCSolver) which is an M/G/1-type Markov chain
  - compute G of the dual process (using FI, CR, RR, IS or NI)
  - obtain R directly from G

Using the GI/M/1 routines of the SMCSolver (Matlab)

Without options

- Formulate your problem as a GI/M/1-type MC, meaning determine A_i and the boundary matrices B_i, for $i \geq 0$
- Select your favorite algorithm, say CR and call

$$\text{R=GIM1\_CR(A,'A','CR',options);}$$

where $A = [A0\ A1\ A2\ A3\ \ldots\ AN]$, any options of the M/G/1 routine for G can be passed as well

- To obtain the steady state π use

$$\text{pi=GIM1\_pi(B,R);}$$

this will compute π such that $\pi e > 1 - 10^{-10}$

RSITÀ DI PISA

versidad os Andes

Universiteit Antwerpen

Using the GI/M/1 routines of the SMCSolver (Matlab)

GIM1_pi dual selection

- When executing

  R=GIM1_R(A,'A','CR',options);

  the Bright dual is used for positive recurrent chains and the Ramaswami dual for transient chains
- One can select the Bright (B) or Ramaswami (R) dual by replacing 'A' by 'B' or 'R', e.g.,

  R=GIM1_R(A,'B','CR',options);

  uses the Bright dual even when the chain is transient

RSITÀ DI PISA

Universidad de los Andes

Universiteit Antwerpen

Using the GI/M/1 routines of the SMCSolver (Matlab)

GIM1_pi options

- MaxNumComp: Max. number of comp. of π (default: 500)
- Verbose: The accumulated probability mass is printed every n steps when set to n (default:0)
- Boundary: Allows solving the GI/M/1-type MC with a more general boundary

$$P = \begin{bmatrix} B_1 & B_0 & & & 0 \\ B_2 & A_1 & A_0 & & \\ B_3 & A_2 & A_1 & A_0 & \\ B_4 & A_3 & A_2 & A_1 & \ddots \end{bmatrix}$$

- GIM1_pi(B,R,'Boundary',[B0; A1; A2; ...]);

Discrete time Non-skip-free (NSF)-type Markov chains

NSF-type transition matrix P

- QBD is skip-free in both directions, NSF is skip-free in neither

$$P = \begin{bmatrix} B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} & \cdots \\ B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \\ B_{N,0} & B_{N,1} & B_{N,2} & B_{N,3} & \cdots \\ A_0 & A_1 & A_2 & A_3 & \ddots \\ & A_0 & A_1 & A_2 & \ddots \\ 0 & & \ddots & \ddots & \ddots \end{bmatrix}$$

Discrete time Non-skip-free (NSF)-type Markov chains

Algorithms for NSF

- Reblock to M/G/1-type Markov chain
- Solution
  - Could use any M/G/1-type algorithm to compute G matrix of size mN
  - Often faster: Gail, Hantler, Taylor (GHT) algorithm (1996) that shows that G is determined by its first block row (size m matrices $G_1, \ldots, G_N$) (see also Wuyts, Van Houdt, Boel, Blondia 1999)
  - SMCSolver implements GHT algorithm
- If level increases by at most $M \approx N$ levels, better to reblock to QBD (do not need Ramaswami's formula to compute π)

Using the NSF routines of the SMCSolver (Matlab)

Without options

- Formulate your problem as a NSF-type MC, meaning determine A_i and the boundary matrices $B_{j,i}$, for $i \geq 0$ and $j = 1, \ldots, N$
- Compute the G matrix of the reblocked system

```
G=NSF_GHT(A,N);
```

  where $A = [A0\ A1\ A2\ A3\ \ldots\ Amax]$
- To obtain the steady state π use

```
pi=NSF_pi(B,A,G);
```

  this will compute π such that $\pi e > 1 - 10^{-10}$

RSITÀ DI PISA

versidad os Andes

Universiteit Antwerpen

Using the NSF routines of the SMCSolver (Matlab)

NSF_pi options

- MaxNumComp: Max. number of comp. of π (default: 500)
- Verbose: The accumulated probability mass is printed every n steps when set to n (default:0)
- FirstBlockRow: When set to 1, it suffices to give the first blockrow of G as input (default:0)
- If $B_{j,i} = A_i$ for $i \geq 0$ and $j = 1, \ldots, N$, then π is computed more efficiently by the call

```
pi=NSF_pi([],A,G);
```

Using the NSF routines of the SMCSolver (Matlab)

NSF_GHT options

- MaxNumIt: Maximum number of iterations (default: 10000)
- Verbose: When set to k, the residual error is printed every k steps (default:0)
- FirstBlockRow: When set to one, only the first block row of G is returned, which fully characterizes G (default:0)

Advantage of reblock to QBD (not yet implemented)

- If reblock to QBD, then R is characterized by last block column (M size m matrices $R_1, \ldots, R_M$)
- Steady state component $\pi_i = \pi_{i-1} R_1 + \ldots + \pi_{i-M} R_M$ for $i > M$

Download: `http://www.win.ua.ac.be/~vanhoudt/`

If you use the software, please cite the paper(s)