The Nsolve Program

Peter Buchholz
Informatik IV, University of Dortmund
D-44227 Dortmund, Germany
Email: peter.buchholz@udo.edu

1 Short Introduction to Nsolve

`Nsolve` is a program to determine the stationary solution of hierarchical Markovian models exploiting a hierarchical Kronecker structure of the generator matrix. The model class contains stochastic automata networks (SANs) without functional transition rates and superposed generalized stochastic Petri nets (SGSPNs) which are both described by a hierarchical model with one state in the upper level. A hierarchical model consists of a number of components (LLMs) and one high level model (HLM). Detailed descriptions of the required matrix structure and some solution methods can be found in [4, 7, 5, 2]. The different file formats which are used to describe hierarchical models are explained in a companion report [8].

`Nsolve` includes iterative solution techniques using the hierarchical Kronecker representation of the matrix and it includes as well iterative techniques using a sparse matrix representation. In the latter case, the flat matrix is first generated out of the hierarchical Kronecker representation. Observe that the implementation of the generation is not very efficient such that solution techniques for sparse matrices should only be used for models with a modest state space size ($\leq 10^5$).

`Nsolve` requires the following files:

- ⟨ `name` ⟩`.conf` configuration file (Description in 2)
- ⟨ `name` ⟩`0.mat` HLM matrix (Description in [8])
- ⟨ `name` ⟩`.spa` HLM state space (Description in [8])
- ⟨ `name` ⟩j`.mat` LLM j matrix (Description in [8])

  A file for each LLM has to exist. The number of LLMs can be found in the configuration file.

All files may include comments. Comments start with # as the first character in a line, then the rest of the line is interpreted as a comment!

The program is called with

```
Nsolve  name
```

2 Configuration File

The configuration file includes the following values (plus possibly some comments):

```
Number of LLMs (integer ≥ 0)
Solution method (integer ≥ 0)
Maximum number of iterations (integer ≥ 0)
ε1 (real ≥ 0.0)
ε2 (real ≥ 0.0)
ε3 (real ≥ 0.0)
Maximum CPU-time (real ≥ 0.0)
Relaxation-parameter (real ≥ 0.0)
Output overall vector (Boolean)
Output HLM vector (Boolean)

for all LLMs 1,..., Number of LLMs
Output LLM vector (Boolean)
```

`Initial state HLM (integer` ≥ -2`)`

`if initial state HLM` ≥ 0 `then`
`for all LLMs` $1, \ldots,$`Number of LLMs`
`Initial state LLM (integer` ≥ 0`)`

2.1 Interpretation of the parameters

Values in the configuration file are interpreted as follows.

`Number of LLMs`
The number of components which form the model.

`Solution method`
The currently available solution methods are summarized in table 1. The number in the first column of the table has to be used in the configuration file.

`Maximal number of iterations`
The solution stops, if the number of iterations has been reached

ϵ_1
The solution stops, if the absolute value if the largest element in the residual vector becomes smaller than ϵ_1 . The residual vector is currently checked every 10 iterations in most methods. Observe that in some methods like the projection methods TFQMR and BICGSTAB the residual can only be estimated during computation. Thus, these methods may stop too early or too late. In any case, the true residual is computed after termination of the method.

ϵ_2
The value is used for several purposes. In A/D-methods, an A/D step for the HLM is performed if the residual norm projected onto the HLM state space exceeds ϵ_2. For methods with ILUTH preconditioner, the value is interpreted as the threshold for keeping non-zero elements during the incomplete factorization.

ϵ_3
The value is only used for A/D-methods where an A/D step for a LLM is performed, if the residual norm projected onto the LLM state exceeds ϵ_3.

`CPU time limit`
The method stops, if the CPU time of the solution (excluding the time for matrix generation) exceeds the limit.

`Relaxation-parameter`
Relaxation-parameter for the methods JOR, SJOR, SOR and BSOR. The value should be taken from $(0, 2)$.

`Output parameters for solution vectors`
Parameter values can be 0 (= no output) or 1 (= output). The first parameter is for the complete solution vector, the second for the solution vector of the HLM and then follow the values for all LLMs. Solution vectors are written in files. File `name_overall.vec` contains the complete vector, `name0.vec` HLM vector and `name`j`.vec` the vector for LLM j.

`The initial distribution`
Different methods exist to define the initial distribution. The first value determines the type of the initial distribution. If the first value is larger or equal to zero, then it is assumed that a single initial state is defined. In this case the following values include the initial states of the LLMs. Observe that not every combination of states defines a correct state. If inconsistencies are detected, the program tries to repair the state and prints a warning. The distribution is initialized with 1.0 at the initial state and 0.0 elsewhere. Usually a uniform distribution is recommended as initial distribution because in a hierarchical model all states are reachable if the state space is generated appropriately (see [5]). A uniform distribution is generated if the first state value is set to -2. In this case the following values are not interpreted. The last possibility to define the initial distribution is to read it from a file. In this case the value in the configuration file has to be set to -1. The name of the file has to be `name_overall.reach`. The vector length has to be equal to the number of states in the state space of the hierarchical model. The vector is normalized to 1 before the iteration starts. With this feature result vectors of one method can be used as initial vectors of another method.

Apart from the parameters in the configuration file, some parameters exist in the source files of the different methods. However, modifications of these parameters requires a modification of the source code and recompilation of the program. We tried to set the parameters to reasonable values, but the choices might not be optimal for specific examples.

The methods NO_SOLUTION (0) and MAT_GEN (1) do not analyze the system. With NO_SOLUTION only the matrix structures are built to prove consistency of the model. MAT_GEN generates a flat sparse matrix which is written in the file `name_overall.mat`. The file format for the flat matrix can be found in [9].

References

[1] P. Buchholz. Numerical solution methods based on structured descriptions of Markovian models. In G. Balbo and G. Serazzi, editors, *Computer Performance Evaluation - Modelling Techniques and Tools*, pages 251–267. Elsevier, 1992.

[2] P. Buchholz. A framework for the hierarchical analysis of discrete event dynamic systems. Habilitationsschrift, Fachbereich Informatik, Universität Dortmund, 1996. available upon request.

[3] P. Buchholz. An aggregation/disaggregation algorithm for stochastic automata networks. *Probability in the Engineering and Informational Sciences*, 11(2):229–253, 1997.

[4] P. Buchholz. An adaptive aggregation/disaggregation algorithm for hierarchical Markovian models. *European Journal of Operational Research*, 116(3):85–104, 1999.

[5] P. Buchholz. Hierarchical structuring of superposed GSPNs. *IEEE Transactions on Software Engineering*, 25(2):166–181, 1999.

[6] P. Buchholz. Projection methods for the analysis of stochastic automata networks. In B. Plateau, W. J. Stewart, and M. Silva, editors, *Numerical Solution of Markov Chains (NSMC'99)*, pages 149–168. Prensas Universitarias de Zaragoza, 1999.

[7] P. Buchholz. Structured analysis approaches for large Markov chains. *Applied Numerical Mathematics*, 31(4):375–404, 1999.

[8] P. Buchholz. A matrix format for structured markovian models. working paper, 10 2010.

[9] P. Buchholz. The sparse matrix file format. working paper, 10 2010.

[10] P. Buchholz and T. Dayar. Block SOR for Kronecker structured representations. *Linear Algebra and its Applications*, 386:83–109, 2004.

[11] P. Buchholz and T. Dayar. Block SOR preconditioned projection methods for Kronecker structured Markovian representations. *SIAM Journal on Scientific Computing*, 26:1289–1313, 2005.

[12] P. Buchholz, J. Dunkel, B. Müller-Clostermann, M. Sczittnick, and S. Zäske. *Quantitative Systemanalyse mit Markovschen Ketten*. Teubner-Texte zur Informatik Band 8. Teubner, 1994.

[13] R. W. Freund and M. Hochbruck. On the use of two QMR for solving singular systems and applications in Markov chain modelling. *Numerical Linear Algebra and Applications*, 1:403–420, 1994.

[14] G. Horton and S. Leutenegger. A multi-level solution algorithm for steady state Markov-chains. *ACM Performance Evaluation Review*, 22:191–200, 1994.

[15] Y. Saad and K. Wu. DQGMRES: a direct quasi-minimal rsidual algorithm based on incomplete orthogonalization. *Numerical Linear Algabra and Applications*, 3:329–343, 1996.

[16] W. J. Stewart. *Introduction to the numerical solution of Markov chains*. Princeton University Press, 1994.

[17] E. Uysal and T. Dayar. Iterative methods based on splittings for stochastic automata networks. *European Journal of Operational Research*, 110(1):166–186, 1998.

[18] H. A. van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific Computing*, 13:631–644, 1992.

Nr.	Method	Explanation
		Matrix structure test
0	NO SOLUTION	Matrix generation without solution
1	MAT_GEN	Flat matrix generation
		Solution techniques for sparse matrices
2	LU	LU-Factorization ([16] pp. 66ff, [12] pp. 30ff)
11	POWER	Power-Method ([16] pp. 121ff)
12	JOR	JOR ([16] pp. 127f)
13	SOR	SOR ([16] pp. 128f)
21	POWER A/D	Power-Method with Aggregation/Disaggregation ([12], pp. 51ff)
22	JOR A/D	JOR with Aggregation/Disaggregation ([12], pp. 51ff)
23	SOR A/D	SOR with Aggregation/Disaggregation ([12], pp. 51ff)
24	KMS	KMS ([16], pp. 308)
25	ML	Multi-level method ([14])
31	GMRES	GMRES ([16], pp. 198ff)
32	BICGSTAB	BiCGSTAB ([18]))
33	TFQMR	Transpose free variant of QMR ([13]))
61	POWER ILU0	Power-Method with ILU0 Preconditioning ([16], pp. 145)
62	POWER ILUTH	Power-Method with ILUTH Preconditioning ([16], pp. 145)
63	GMRES ILU0	GMRES with ILU0 Preconditioning ([16],pp. 205ff, 145)
64	GMRES ILUTH	GMRES with ILUTH Preconditioning ([16],pp. 205ff, 146)
65	BICGSTAB ILU0	BICGSTAB with ILU0 Preconditioning
66	BICGSTAB ILUTH	BICGSTAB with ILUTH Preconditioning
67	TFQMR ILU0	TFQMR with ILU0 Preconditioning
68	TFQMR ILUTH	TFQMR with ILUTH Preconditioning
		Solution techniques for Kronecker representations
111	POWER	Power-Method ([12] pp. 203)
112	JOR	JOR ([1])
113	SJOR	SJOR ([12] pp. 204f)
114	SOR	Real SOR ([17])
121	POWER A/D	Power-Method with Aggregation/Disaggregation ([3, 4])
122	JOR A/D	JOR with Aggregation/Disaggregation ([3, 4])
123	SJOR A/D	SJOR with Aggregation/Disaggregation ([3, 4])
131	GMRES	GMRES ([16], Spp 198ff)
132	DQGMRES	DQGMRES ([15])
133	ARNOLDI	Arnoldi-Method ([16], pp. 191)
134	CGS	CGS ([16], pp. 222)
135	BICGSTAB	BiCGSTAB ([18]))
136	TFQMR	TFQMR ([13]))
150	PREPOWER	Power with Neumann preconditioning ([7])
151	PREGMRES	GMRES with Neumann preconditioning ([7])
152	PREARNOLDI	Arnoldi with Neumann preconditioning ([7])
154	PRECGS	CGS with Neumann preconditioning ([6])
155	PREBICGSTAB	BICGSTAB with Neumann preconditioning ([6])
156	PRETFQMR	TFQMR with Neumann preconditioning([6])
157	BSOR_BICGSTAB	BICGSTAB with Block SOR preconditioning ([11])
157	BSOR_GMRES	GMRES with Block SOR preconditioning ([11])
157	BSOR_TFQMR	TFQMR with Block SOR preconditioning ([11])
172	STR_BSOR_T	Two level BSOR ([10])

Table 1: Overview of available solution methods