

Autonomous Solution Methods for Large-Scale Markov Chains

W. S. Barge, II
Ph.D. Candidate

W. J. Stewart
Professor

Operations Research Program
North Carolina State Univ.
Raleigh, NC 27695

Department of Computer Science
North Carolina State Univ.
Raleigh, NC 27695

1 Introduction

Since their introduction in the early 1900s, Markov chains have been proposed as a means of modeling a variety of stochastic processes, including weather forecasting, voting patterns, and demographic trends. Markov chains have even been suggested as a model to predict and guide usage of the World Wide Web [13].

In his book, *Modeling and Analysis of Stochastic Systems* [10], Kulkarni provides some details on how applications in genetics, sociology, manpower planning, and telecommunications could be modeled as Markov chains. In most scenarios, the models are kept very small, usually less than several hundred states, in order to be kept tractable. Only in the areas of telecommunications and computer systems performance modeling have larger models been used successfully, and this because of the detailed knowledge of numerical solution algorithms possessed by the researchers working in these areas.

We are interested in autonomously choosing a method for computing the stationary distribution of finite, irreducible Markov chains. Essentially, this means solving the linear system

$$\pi Q = 0, \quad \sum_i \pi_i = 1, \quad \pi > 0 \quad (1)$$

where Q is the matrix of transition rates (or infinitesimal generator) of the Markov chain [15]. The rows of Q sum to zero and q_{ij} is an element of Q . The element q_{ij} is the rate of transition from state i to j . The diagonal elements are defined by

$$q_{ii} = - \sum_{i \neq j} q_{ij}.$$

1.1 Possible Roadblocks

The successful use of a Markov chain model involves at least three (global) steps:

1. A detailed knowledge and understanding of the application domain such as manufacturing, queuing systems, or telecommunications.
2. A working knowledge of how to construct a Markov chain model.
3. The knowledge and ability to obtain useful information from the Markov chain after it is constructed.

We contend that step 3 represents a real and often unnecessary roadblock for potential users of Markov chain modeling.

The ability to model a system or procedure using a Markov chain can be distinct from the ability to analyze the chain's behavior. Modeling ability is derived primarily from a *basic* knowledge about the meaning of a Markov chain coupled with a *detailed* knowledge of a specific application area. One of the roadblocks to greater application of Markov chains is that potential but non-numerically sophisticated users possess the detailed domain knowledge needed to construct a large Markov chain but may have a difficult time deciding which numerical method might be best suited to their applications. A realistic Markov chain model can easily contain thousands or hundreds of thousands of states. The transition matrix can be quite sparse, which means that compact storage schemes are often required to make the matrix manageable for the computer. The choice of a particular solution method also has a great effect on the amount of time required to obtain a solution. Even after deciding upon a certain method, implementation details imposed by compact storage schemes and the nature of the methods themselves pose additional barriers. As a consequence, users may severely restrict their models to keep them small enough so that software packages such as MATLAB can be used.

The difficulty with a self-imposed model size restriction is that even small, simple models can generate very large state spaces. One example we have used in our research is a manufacturing model proposed by Krieg and Kuhn [9] that describes a kanban controlled production system where products are processed on a single manufacturing facility. With only five different products and five kanbans for each product the resulting Markov chain has 71,280 states. Clearly, an *a priori* size limitation is not a realistic approach for removing roadblocks to the usefulness of Markov chain models.

Greenbaum [6, p. 94] observes that "*it remains an open problem to characterize the classes of problems for which one [solution] method outperforms the others.*" Indeed, she reports that examples can be constructed for which a particular solution method performs the best, or the worst, depending on the researcher's goal. Because the modeling nuances are limitless we hold out little hope for a far reaching theoretical classification scheme that encompasses all potential Markov chain models and the currently known solution techniques.

1.2 Roadblock Removal

Where does this leave the non-numerically sophisticated user? This person is in need of an *intelligent software tool* to guide the analysis. We envision an experimentally-validated solution approach embedded into a software application. The application should have a suitable user interface to guide the analysis of a Markov chain. We propose a battery of diagnostic tests to be performed at the time the Markov chain matrix is obtained by the software. The results of these tests are interpreted by an *expert system* and then a suitable solution method is chosen and executed. In addition to selecting an appropriate solution technique any needed algorithm parameters will be supplied for the novice user.

2 Technology and Literature Review

The purpose for modeling a system, be it with Markov chains, or any other technique, is to gain valuable information about the system. In terms of Markov chains, this implies the ability to compute various functionals, or metrics, that are characteristic to the system (as described by the Markov chain model.) The functional that is the focus of this research is the *stationary distribution* of the Markov chain. Some functionals can be obtained by more than one method and this is true for the stationary distribution. If the state space is large, the novice user of Markov chains may not be equipped to efficiently obtain the stationary distribution. The choice of a solution technique can have a large impact on the

speed, efficiency, and accuracy of the computed solution. A person with experience solving Markov chains might employ a variety of tests or judgments to reach a conclusion about which solution technique to use.

The idea of providing analysis aids is not new. In their book, *Fitting Equations to Data*, authors Daniel and Wood begin with a flow chart that offers a scheme for making sense of raw scientific data. In their case, the goal is the design of a useful predictor equation that will allow greater inference into the behavior of the underlying system [4]. Their general purpose was to help the analyst, scientist, or engineer to select appropriate parameters and make reasoned assumptions about least-squares methods of designing fitted predictor equations for data.

In their 1976 book, *Markov Chains: Theory and Applications*, Isaacson and Madsen devote an entire chapter to the topic of using a computer to help classify and analyze a Markov chain. One of their main goals is that “*no preliminary analysis of the Markov chain by the user of the program is required*” [8, p. 223]. The chapter also contains many flow charts that describe a logical procedure for the analysis of a Markov chain. Many of their methods depend heavily on the calculation of powers of matrices and would not be appropriate for the large matrices we use in our examples. Also, their treatment of finding the stationary distribution provides little insight into when one solution method might be preferred over another.

The book *Templates for the Solution of Linear Systems* by Barrett, *et al*, describes a wide variety of iterative solution methods. The flowchart on the inside jacket gives a suggested rationale for their possible selection and use. However, the flowchart does not consider the characteristics of a Markov chain and is not designed in the context of an intelligent software application [2].

In the 1980’s, the academic and industrial statistical communities had a substantial amount of interest in augmenting statistical analysis software with embedded statistical expertise [12, 11, 7, 3]. One of the chief concerns was that the advent of Personal Computers (PC) had made statistical software widely available but the software lacked intelligence and was not user-friendly [7]. In view of what exists for the casual PC-user today, that charge could apply to almost any software package of the 1980’s; however, the statisticians had specific goals in mind for what embedded expert knowledge could do for data analysis.

By the 1990’s commercial software claimed to be catching up with the desires of statisticians [1]. Today, advertisements for statistical software claim to have expert systems and artificial intelligence designed into the software. In our research we do not evaluate these claims – we comment on them here only to show the absence of similar claims from software that can analyze Markov chain models.

3 Selection of a Solution Method

We specify a battery of six initial tests to recommend a solution approach. The tests are each discussed in detail:

1. *Structure*. Is the Markov chain irreducible? At this time only irreducible chains are accepted in the software; future work could include reducible chains by giving the user the option of choosing a specific block to analyze.
2. *Order*. This is the $n \times n$ size of the matrix and this information is available when the matrix is acquired by the software. This is not a *test* but rather just a fact about the matrix that has bearing on the selection of a solution method.
3. *Time estimate for a direct solution*. The time estimate is computed by first making an estimate of the worst case fill-in that would result if the stationary distribution is obtained using Gaussian Elimination(GE). As the matrix is brought into the software

a check is made to find the location of the element(s) that are furthest off the main diagonal. Knowing this information provides a band of possible fill-in should the matrix be solved using GE. For each element that must be eliminated we allocate two floating-point operations (flops). The software also contains a loop that performs $5 \times$ (one million flops) to get an estimate of the processor speed in seconds/million flops. The number of flops required for a GE solution \times the flop rate give an estimate, in seconds, for a direct solution of the Markov chain. The essential issue here is how long must the user wait for a returned solution. We suggest that 30 seconds is an upper limit for user waiting time; of course, a shorter waiting time is desirable.

4. *A heuristic measure of the sparseness.* Most large Markov chains will be represented by relatively sparse matrices; however, some are more sparse than others. We compute a numerical value, or *matrix score* as the matrix is stored into the software. It attempts to give a quantitative measure of how the matrix elements are dispersed away from the main diagonal. This is an important storage consideration when deciding between a direct or iterative solution method. A matrix with most of its elements near the main diagonal will experience little fill-in; conversely, even a matrix that starts off rather sparse can experience significant fill-in when its elements are widely dispersed off of the main diagonal. An experienced researcher would likely gain this type of insight from seeing a pixel-plot of the matrix elements; however, the novice user might not recognize the meaning of the plot. The *matrix score* is derived from what is basically a histogram of elements and their distance from the diagonal. Let x_i be the number of matrix elements that are a distance i from the diagonal. The area under the histogram is computed and then scaled by n^2 where n is the order of the matrix. The matrix score is

$$matrixScore = \frac{\sum_{i=1}^{n-1} x_i}{n^2}. \quad (2)$$

In testing we observed that matrices with $n \geq 200$ and *matrixScore* > 0.85 , are susceptible to fill-in amounts that might contraindicate the use of a direct solution method. Additionally, through experiments we have found the matrix score to be useful in deciding between an iterative and projection method for finding the stationary distribution.

5. *The degree of decomposability.* In this test we examine the matrix to determine if it is Nearly Completely Decomposable, or NCD. The fact of being NCD will support the use of a block iterative method (such as block SOR) or an iterative aggregation/disaggregation method (IAD). In our current software we have the IAD method available for solving NCD matrices.
6. *The periodicity of the Markov chain.* Both direct and iterative methods can take advantage of periodicity if the matrix is first permuted to Cyclic Normal Form. During initial processing, the matrix is tested to determine its periodicity, p .

4 Design of the Expert System

When the matrix that represents a Markov chain model is acquired by our intelligent software program, many of its characteristics can be quickly examined and quantified (this is the initial battery of tests discussed in the previous section). By observing and combining judgments about this information an *experienced* researcher or practitioner can usually propose a solution technique in a short amount of time. A problem this research seeks to address is how to obtain a proposed solution technique without the services of an expert and with little or no intervention from the novice user. The job of the expert system (ES) is to compare the results of the initial tests against its battery of known *expert* patterns and

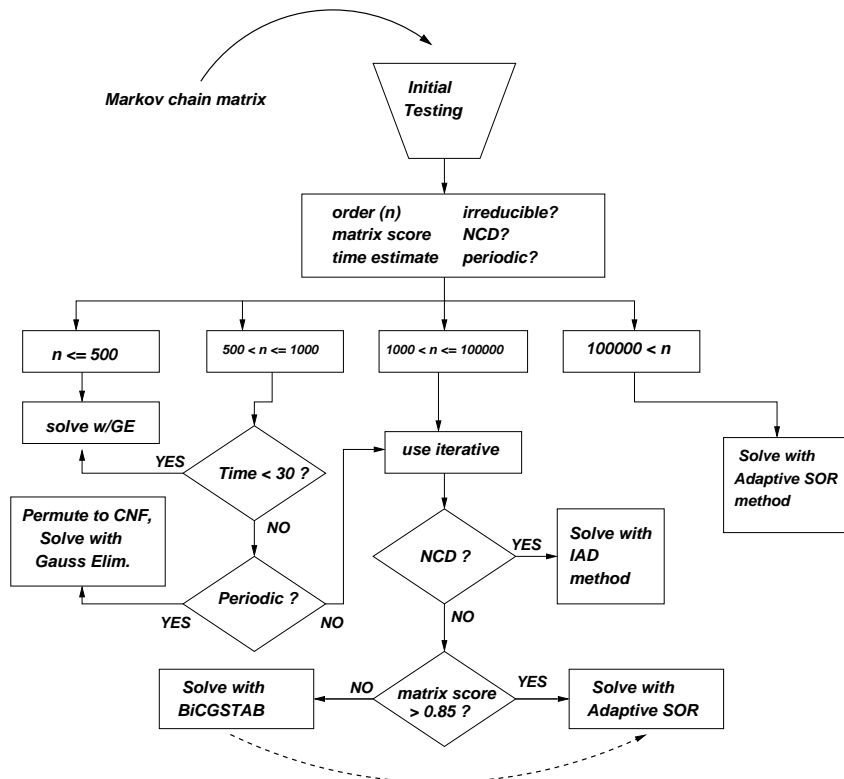


Figure 1: Sieve decision chart

then render a proposed solution technique. The expert patterns we designed are embodied in Figure 1. The dashed line from “BiCGSTAB” to “Adaptive SOR” indicates our experiments to provide a switching mechanism in case the expert system chooses a method that fails or does not otherwise provide the solution. In section 2, we reviewed efforts to embed expert knowledge into statistical software; in a similar vein, we propose that a basic *expert system* can help a novice user select a solution technique for a Markov chain. The interested reader is invited to see an appropriate text [14, 5, 16] for an in-depth study of knowledge representation, expert systems, and their applications. We will represent our knowledge about Markov chains as a collection of *condition-action pairs*, or rules. In this way, our expert system will be a tool in our larger goal of making Markov chains more accessible to novice users.

5 A Proposed Graphical User Interface

Our user interface is graphical in nature; it uses controls and affordances familiar to all users of today’s PCs and workstations. The interface has a plot window that is used to display a pixel plot of the matrix elements and later used to display a plot of the solution vector. Textual feedback is shown in a listbox on the GUI. A screen shot is shown in Figure 2.

In our current approach, the non-zero elements of the Markov chain matrix are stored in a flat text file in i, j, a_{ij} format¹ When the user selects a particular matrix from a dialog box, the software application begins the process of acquiring and testing the matrix to

¹Other input formats and schema are certainly possible. In fact, another important roadblock to the widespread use of Markov chain modeling is the *specification* of a large model; we do not consider this problem in our immediate research.

gain information that can be used to recommend one solution technique above the others. As an initial step, the non-zero elements are stored into Harwell-Boeing(HB) format. All subsequent calculations and tests access the matrix via the HB format and the matrix is never inflated to its full $n \times n$ size. After a battery of initial tests, a solution method is selected by an embedded, forward-chaining expert system. Currently five solution methods are available in the software: Gaussian Elimination, permuted Gaussian Elimination, Adaptive SOR, Iterative Aggregation/Disaggregation (IAD), and BiCGSTAB.

Users can either choose an *Automatic* solution or a *Manual* solution. In Automatic mode the interface reads in the model, obtains a recommendation from the expert system and then uses that recommendation to solve for the stationary distribution. In Manual mode the interface will recommend a solution method but lets the user make the actual choice of which method to use.

The GUI is implemented using MATLAB (version 5.3, R11) and FORTRAN (f77) code, including selected algorithms and subroutines from MARCA². We chose MATLAB as our primary investigation tool because of its graphic, scientific, and user interface capabilities.

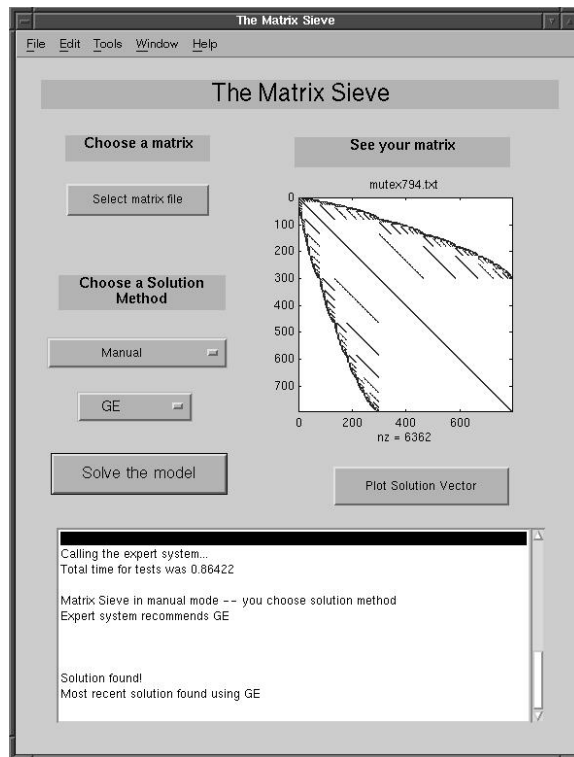


Figure 2: Proposed GUI

6 Numerical Experiments

To investigate the usefulness of our approach we will define four alternative methods for obtaining the stationary solution to a Markov chain:

- *The naive approach.* The Markov chain exists in MATLAB as a full matrix. By *full* we mean that every element, including zero elements, are stored and the matrix exists

²MARCA is the Markov Chain Analyzer, written by Dr. W. J. Stewart.

in its $n \times n$ format. The chain is represented by the infinitesimal generator, Q . The system to be solved is $Q^T \pi^T = 0$ where π is the stationary distribution vector. The last row of system is replaced to reflect the constraint $\sum \pi_i = 1$ and the resulting system is solved using the MATLAB command $A \backslash b$.

- *The advanced approach.* The Markov chain is stored in MATLAB using native MATLAB sparse matrix format. The system is then solved using techniques from the sparse matrix functions available in MATLAB. Additionally, the user must supply any needed parameters.
- *The expert system approach.* This is our proposed approach as described in this paper.
- *The validation approach.* This is a validation step performed without the user interface and the expert system. Through this step we can observe the relative success of the expert system approach. In this step we will apply a variety of solution methods to the model and seek to answer the question “*Did the embedded expert system choose the best solution method?*”

For our numerical experiments we used five Markov chain models from a variety of disciplines and we now present results and discussion for one of those models. All experiments were performed on a Sun UltraSPARC-IIi (Ultra 10) workstation with 256 MB of RAM and a 440 MHz RISC cpu. Unless otherwise stated all timings are given in seconds.

6.1 Description of the Kanban Model

This model is based on a kanban³ manufacturing system and is adapted from the work of Krieg and Kuhn [9]. In this model, a kanban controlled system processes three or more different products on a single machine. The machine is setup for each product, in succession, even if there is not currently a shortage of that product. Setup times are different for each product and are *not* instantaneous. At the end of a setup, if there is still no demand for the product then the machine is setup to produce the next product, according to the setup schedule. Unfulfilled demand is considered lost sales and is not back ordered. Raw materials are always available. Demands and setup times are exponentially distributed which leads to a continuous-time Markov chain model.

As a specific example, consider the kanban model with 5 products and 5 kanbans for each product. This Markov chain model has 71,280 states and 438,480 non-zero elements. In subsequent paragraphs, matrices from this example will be labeled as `kanbanXY` where X describes the number of kanbans and Y describes the number of products.

6.2 Stationary Solution Results

In the case of the `kanban55` model, the initial battery of tests reveal the information in Table 1. The large time estimate for using Gaussian Elimination (209 seconds), along with the size of the state space (71,280), cause the expert system to consider methods other than Gaussian Elimination. The entries in Table 1 constitute the initial working memory of the expert system. Recall that the logic shown in Figure 1 reflects the rules of the expert system. The expert system evaluates the contents of the working memory, adding conclusions to the working memory as appropriate, and reaches the conclusion to recommend the use of the BiCGSTAB method to solve for the stationary distribution. Using the information in Table 1, the reader can trace the logic shown in Figure 1.

The processing time to perform the initial tests are shown in Table 2. These times do not include overhead time required to obtain the matrix from disk storage.

³A *kanban* is a card that indicates the status of a product. Cards are placed on inventory containers or on a set-up scheduling board. The location of a card indicates, for example, if a particular product needs to be produced or if that product already has sufficient inventory.

Table 1: Initial test results for `kanban55` model

Test Performed	Result
order	71,280
periodicity	acyclic
NCD test	<i>not</i> NCD
structure	irreducible
matrix score	0.3815
GE time estimate	209 seconds

Table 2: Initial test times for `kanban55` model

Test Performed	Time
order	0.00
periodicity	0.38
NCD and structure test	2.96
matrix score	0.15
GE time estimate	0.12
ES (inference engine)	0.69
TOTAL Time	4.30

One of our principle goals is to evaluate the efficacy of the expert system approach for obtaining the stationary distribution. In Table 3 we show the results of the four solution approaches discussed earlier. The results for this specific example indicate that the expert

Table 3: Solution times for `kanban55` model

Soln Approach	Method	# Iterations	Time
Naive	A\b (direct)	–	fail
Advanced	GMRES (10)	116	1,308
Advanced	GMRES (20)	32	846
Advanced	GMRES (100)	–	fail
Advanced	QMR	–	fail
Advanced	BiCGSTAB	–	fail
Expert System	BiCGSTAB	127	14.4
Validation	SOR ($\omega = .50$)	–	fail
Validation	SOR ($\omega = .75$)	795	87
Validation	SOR ($\omega = 1.25$)	–	fail
Validation	IAD method	–	fail
Validation	Gauss-Seidel	707	76
Validation	Power Method	297	38

system did choose the fastest solution method. The power method of obtaining π is often very slow, although it gave good results in this example. Generally, it would not be the first method tried by a human expert. Additionally, the expert system approach is far superior to both the *Naive* and the *Advanced* approaches.

We also examined a `kanban33` model consisting of three products and three kanbans per product. This model has 336 states and 1,416 non-zero elements. The solution results are shown in Table 4. The results show that the number of states can be used to help determine

Table 4: Solution times for `kanban33` model

Soln Approach	Method	# Iterations	Time
Naive	A\b (direct)	–	0.75
Advanced	A\b (direct)	–	0.04
Expert System	GE	–	0.15
Validation	SOR ($\omega = .50$)	187	0.06
Validation	SOR ($\omega = .75$)	114	0.04
Validation	SOR ($\omega = 1.25$)	–	fail
Validation	Gauss-Seidel	90	0.03
Validation	Power Method	106	0.04
Validation	BiCGSTAB	47	0.14

which initial tests are performed on the matrix. For this model, the expert system time (2.20 seconds) exceeds every other successful solution time. Most of the 2.20 seconds was consumed with the NCD and periodicity tests and the outcome of those tests is not needed to quickly obtain a solution for such a small model. This leads us to consider forgoing certain aspects of the initial testing for appropriately small models.

For a novice user, the time cost of the initial testing and the time cost for adaptively choosing parameters would likely be offset by overall time saved in obtaining the solution. Another benefit of the expert system approach is the likely reduction of novice user frustration.

6.3 Solution Vector Re-use

We also consider two likely scenarios that will interest users of Markov models who may want to repeatedly solve a particular model. The first situation involves the concept of scalability where a small model is enlarged. The second situation is one where the size of the model is unchanged but one or more model parameters are varied. In this case, the size of the initial starting vector does not change. The modeler, of course, is interested on how each of these changes affect the solution (i.e., stationary distribution). We take the additional perspective of trying to see what implications these changes have for our expert system and intelligent solution tool. In both scenarios, various methods for re-using the previous solution vector are compared against the default starting vector where each element is $\pi_i = 1/n$ where $n =$ the number of states.

In the first scenario we experimented with how to take advantage of a current solution vector to aid the solution of an enlarged model. The existing solution vector is extended to the proper dimension and then used as the starting vector for the larger model. We examined three alternatives:

- *Alt 1.* Do not use the existing π vector at all; instead, for the new, larger, Markov chain model create a starting vector of size n where each element is $1/n$.
- *Alt 2.* Extend the existing π vector by appending enough zero (0) elements onto the vector to make it the correct length for the new, larger model.
- *Alt 3.* Distribute the probability in the smaller vector over the state space of the new, larger starting vector in way that seeks to preserve the shape of the current solution

vector. This tests the assumption that the states with the most probability in smaller models will still have high probabilities in larger models of the same system.

Table 5 shows the results of extending a `kanban54` model to a `kanban55` model.

Table 5: Extend `kanban54` to `kanban55`

	From	To
Name:	<code>kanban54</code>	<code>kanban55</code>
Order:	28,125	71,280
Non-zeros:	168,125	438,480
	Algorithm	
Alternative	Adaptive SOR	BiCGSTAB
Alt 1	34.54	14.47
Alt 2	40.59	15.20
Alt 3	35.40	14.14

For experiments with the second scenario, we compared the previous solution vector to the default solution vector. In both scenarios, our results indicate that re-use of a previous solution vector gives no distinct advantage over the default starting vector. In some instances, the re-use of a previous solution vector adds time to the solution process.

7 Summary and Outlook

In this paper we have presented some results of our current research in developing a graphical user interface with an embedded expert system that will assist the analysis of a Markov chain. We believe that a suitable graphical user interface (GUI) coupled with an expert system can have a positive impact on making Markov chains more accessible. In this research, we focused on the autonomous selection of a method to solve a linear system for the purpose of obtaining the stationary distribution of the Markov chain. Future work will likely encompass other functionals of Markov chain analysis, such as transient analysis.

We are well on the way to having a generic GUI that uses basic Markov chain terminology; however, Markov chains are used in a wide variety of modeling disciplines and it is difficult to imagine one interface that can suit them all. This makes the specification and internal representation of the Markov chain important considerations. In particular, we would like the specification and generation of the matrix to occur in a domain-specific manner without the user having to write computer code.

During the course of this research we experimented with the feasibility of making our software available on the World-Wide-Web through the use of Java applets. A rudimentary Java applet that performs Gaussian Elimination and provides for transient analysis is available at URL: www4.ncsu.edu/~wsbarge.

References

- [1] R. Andrews. New Software for PCs Help Take Anxiety Out of Statistics. *The Scientist*, 6(2), 1992.
- [2] R. Barrett, M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H. van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, Philadelphia, 1994.

- [3] E. Brent. Statistical Expert Systems: An Example. *J. Statist. Comput. Simul.*, 31: 103-110, 1989.
- [4] C. Daniel and F. Wood. *Fitting Equations to Data*. John Wiley & Sons, New York, 1971.
- [5] J. Durkin. *Expert Systems Design and Development*. Macmillian Publishing, New York, 1994.
- [6] A. Greenbaum. *Iterative Methods for Solving Linear Systems*. SIAM, Philadelphia, PA, 1997.
- [7] T. Hietala. How to Assist an Inexperienced User in the Preliminary Analysis of Time Series. *Proceedings of Compstat 1986*, Physica-Verlag, Heidelberg, 295-300.
- [8] D.L. Isaacson and R.W. Madsen. *Markov Chain Theory and Applications*. Krieger Publishing Company, Malabar, Florida, 1985.
- [9] G. Krieg and H. Kuhn. *A Decomposition Method for Multi-Product Kanban Systems with Setup Times and Lost Sales*. not yet published. March 2001.
- [10] V. Kulkarni. *Modeling and Analysis of Stochastic Systems*. Chapman & Hall, New York, 1995.
- [11] K.M. Portier and P.Y. Lai. A Statistical Expert System for Analysis Determination. *Proceedings of the American Statistical Association Section on Statistical Computing*, (1983) pp 309-311.
- [12] D. Pregibon. Incorporating statistical expertise into data analysis software. *In The Future of Statistical Software*, pp 51-62. National Research Council, National Academy Press, 1991.
- [13] R. Sarukkai. "Link Prediction and Path Analysis using Markov Chains", [web page] <http://www9.org/v9cdrom/68/68.html> [Accessed in May 2001].
- [14] P. Sell. *Expert Systems – A Practical Introduction*. Halstead Press, Great Britain, 1985.
- [15] W. J. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, Princeton, NJ, 1994.
- [16] E. Turban. *Decision Support and Expert Systems*. Prentice Hall, Englewood Cliffs, NJ, 1995.