

# S<sup>2</sup>YSCODE:

## Stochastic Systems Control and Decision Algorithms Software Laboratory, FORTRAN & MATLAB Versions

Emmanuel Fernández-Gaucherand<sup>1</sup>, Jongsup Choi<sup>2</sup>, Dan Gerhart<sup>3</sup>

Department of Systems and Industrial Engineering  
The University of Arizona  
Tucson, Arizona 85721  
U.S.A.

<sup>1</sup>emmanuel@sie.arizona.edu, <sup>2</sup>choi@sie.arizona.edu, <sup>3</sup>sie\_0099@bigdog.engr.arizona.edu

### Abstract

We present here the first report on a software laboratory for stochastic systems control and decision problems: S<sup>2</sup>YSCODE. Two versions have been developed so far, one in FORTRAN 77, and the other in MATLAB. We expect to distribute, free of charge, beta versions of S<sup>2</sup>YSCODE to interested parties. If you would like to be in our list, please send an email message to the first author. In addition, a more extensive report [3] is also available.

**Keywords:** Stochastic Systems; Control and Decision; Dynamic Programming Algorithms.

## 1 Introduction to S<sup>2</sup>YSCODE

Stochastic control/decision systems (SC/DS) find important applications in many areas, such as: routing and flow control in communication networks, scheduling problems in production and manufacturing, capital investment and management, *etc.*; see [1], [2], [10].

Although the analytical aspects, and to a lesser degree the algorithmic aspects, of SC/DS are at a very high state of development, widely applicable and user-friendly algorithm software implementations and/or software tool boxes are conspicuously scarce. For example, "The Extended List of Control Software (ELCS) 5.0" [4] has *no listings at all* of software especially developed for SC/DS. The only competitive alternative the authors are aware

of is the PC-DOS-based software package MDPS [5], which although useful, has a limited scope since it is aimed mostly at operations research applications, and is not fully transportable.

We present here the first report on a software laboratory for computer-aided control system design (CACSD) and experimentation in SC/DS: S<sup>2</sup>YSCODE. Our principal aims in developing S<sup>2</sup>YSCODE have been to develop a user-friendly, interactive and transportable software laboratory for CACSD in SC/DS. The envisioned utility of S<sup>2</sup>YSCODE falls mainly in the realms of: computer experimentation for analytical validation and algorithm development; CACSD and SC/DS validation; and computer-aided instruction. Currently, there are two versions of S<sup>2</sup>YSCODE: one in FORTRAN 77 and the other in the form of a MATLAB toolbox. Both are very user-friendly and interactive. The MATLAB version of S<sup>2</sup>YSCODE has been implemented in the form of MATLAB commands using M-Files [7], [8]. On-line information for S<sup>2</sup>YSCODE commands is available via standard MATLAB help facilities.

The development environment for S<sup>2</sup>YSCODE has been the cluster of Sun SPARC stations in the Laboratory for Algorithmic Research, at the Systems & Industrial Engineering (SIE) Department, The University of Arizona. However, the FORTRAN 77 version is transportable to any UNIX-based machine running FORTRAN, and the MATLAB version is fully transportable to any MATLAB environment, including the Macintosh or DOS student editions of MATLAB [7], [8].

The organization of this report is as follows. In section 2 we summarize the SC/DS amenable to

computational study using **S<sup>2</sup>YSCODE**. Section 3 briefly summarizes the optimal control/decision algorithms implemented by **S<sup>2</sup>YSCODE**. The format for the input data files to **S<sup>2</sup>YSCODE**, interactive data input, error messages, *etc.*, is discussed in section 4. Section 5 presents some examples of the use of **S<sup>2</sup>YSCODE** on some benchmark problems. A translation scheme implemented by **S<sup>2</sup>YSCODE** to convert from system equation description (analytically amenable) to transition matrix description (computationally amenable) for stochastic systems, is discussed in section 5.

## 2 Infinite Horizon Stochastic Control/Decision Problems

We consider discrete-time, finite and infinite horizon optimal stochastic control/decision problems, where the dynamic (nonlinear) stochastic system is modeled as a controlled Markov process; we follow the notation and work within the framework of [2]; so also and [6]. We refer to these books for details, and only give the essentials in the sequel. The *system equation* description of these systems is as follows: the dynamics are described by

$$x_{k+1} = f(x_k, u_k, w_k), \quad k = 0, 1, 2, \dots,$$

where

- $x_k \in \mathbf{S}$  : state,  $u_k \in \mathbf{C}$  : control,
- $w_k \in \mathbf{D}$  : disturbance, having a specified probability distribution  $P(\cdot | x_k, u_k)$ ,
- $\mu_k : \mathbf{S} \rightarrow \mathbf{C}$ ; with  $\mu_k(x_k) \in U(x_k)$  action constraints for admissible policies.

Equivalently, for the case of finitely many possible states, the controlled dynamics of the system can be given in terms of a collection of state transition matrices  $\{P(u) = [p_{ij}(u)]; u \in \mathbf{C}\}$ . Here  $p_{ij}(u)$  is the probability of making a transition from state  $X_t = i$  to  $X_{t+1} = j$ , under the action  $U_t = u$ . **S<sup>2</sup>YSCODE** has the capabilities of working with either form for describing the system dynamics. Both finite and infinite horizon cost criteria stochastic optimal control/decision problems can be solved. In particular, the following cost criteria can be used.

1. The finite horizon cost (FH).

$$J_N^\pi(x_0) := \mathbf{E}_{w_k} \left\{ \sum_{k=0}^{N-1} g[x_k, \mu_k(x_k), w_k] + g_N[x_N] \right\}.$$

where we have that

- $x_0$  is the initial state,
- $\pi = (\mu_0, \dots, \mu_{N-1})$  is the policy for finite horizon,
- $g(x, u, w)$  is the cost per stage,
- $g_N(x)$  is the (final) cost at stage N, and
- $U(x)$  is the set of feasible actions in state  $X_t = x$ .

2. The expected discounted cost (DC).

In this case, the cost functional takes the form

$$J_\alpha^\pi(x_0) := \lim_{N \rightarrow \infty} \mathbf{E}_{w_k} \left\{ \sum_{k=0}^{N-1} \alpha^k g[x_k, \mu_k(x_k), w_k] \right\}.$$

where we have that

- $x_0$  is the initial state,
- $\pi = (\mu_0, \mu_1, \dots)$  is the policy being used,
- $0 < \alpha < 1$  is the discount rate, and
- $g(x, u, w)$  is the cost per stage.

3. The long-run expected average cost (AC):

$$J^\pi(x_0) := \lim_{N \rightarrow \infty} \frac{1}{N} \mathbf{E}_{w_k} \left\{ \sum_{k=0}^{N-1} g[x_k, \mu_k(x_k), w_k] \right\}.$$

## 3 Computational Methods

For problems with FH, the dynamic programming (DP) algorithm is implemented by **S<sup>2</sup>YSCODE**; (see [2], [6]). **S<sup>2</sup>YSCODE** also implements the two main general-purpose algorithms for computing both optimal values and policies in problems with infinite horizon, namely: policy iteration(PI) and value iteration(VI). Starting from arbitrary initial guesses, both algorithms iteratively seek to improve policies or values, respectively, using the optimality equations and conditions of dynamic programming mentioned above. Both algorithms terminate in a finite number of iterations when the spaces involved are finite sets, as necessarily required in computations. However, when the number of states is large, these methods become impractical because of the large overhead per iteration. In this situation, special purpose algorithms that exploit specific problem structure may need to be developed. We briefly comment on the (general) PI and VI algorithms next. For more details, see [2], [6], [9], [10].

### 3.1 Policy Iteration

This algorithm operates as follows. An initial stationary policy  $\pi^0 = \{\mu^0, \mu^0, \dots\}$  is adopted, and the corresponding cost function  $J^{\mu^0}$  is calculated. Then an improved policy  $\pi^1 = \{\mu^1, \mu^1, \dots\}$  is computed as the minimizer in the DP equation corresponding to  $J^{\mu^0}$ , and the process is repeated.

### 3.2 Value Iteration

This method yields an optimal policy after a finite number of iteration. Furthermore, this method can be substantially improved thanks to the availability of certain error bounds. Starting from an arbitrary initial guess, the optimal cost is iteratively approximated. At each iteration, a policy is obtained also.

## 4 S<sup>2</sup>YSCODE Examples and Tests

A FORTRAN subroutine environment and a MATLAB toolbox, built using M-Files, were written to implement the FH, policy iteration and value iteration algorithm. The FORTRAN programs have been found to be able to handle problems with up to 500 states. In addition, the programs can generate input data during execution, which is an important feature given that the input data requires a large memory space allocation. We have tested our FORTRAN programs and MATLAB toolbox on the SPARC LX's and SPARC 2's Sun workstations in the Laboratory for Algorithmic Research, in the Systems & Industrial Engineering Department at the University of Arizona.

Next, we will present some computational experiments using benchmark problems found in [2] and [5], illustrating the input data file formats, the interactive capabilities, and output of S<sup>2</sup>YSCODE.

### 4.1 FORTRAN Version

The FORTRAN version consists of a main program and several subroutines. The user can use the execution program interactively, and change options to execute different experiments. The output can be seen on the terminal screen, and it is saved in the output file which the user specifies during the interactive process.

#### 1. Subroutines.

- **Trans:** transforms a given system equation description to a transition probability matrix description for the system dynamics.

- **SOLFin:** computes the optimal policy and value for the finite horizon problem.
- **SOLPDis:** computes the optimal policy and value for the discounted cost using policy iteration.
- **SOLPAve:** computes the optimal policy and value for the average cost using policy iteration.
- **SOLVDis:** computes the optimal policy and value for the discounted cost using value iteration.
- **SOLVAve:** computes the optimal policy and value for the average cost using value iteration.

#### 2. Input Parameters.

ProbTyp : MAXimization or MINimization  
CostTyp : FH, Discounted or Average  
Method : Policy Iteration or Value Iteration  
NS : the number of states  
MC : the number of actions(decisions)  
Nstage : the number of stages for FH  
Alpha : the discount rate  
eps : the error bound - epsilon

#### 3. Execution Procedures.

In order to execute the FORTRAN version, the user has to type "dyn" for infinite horizon problems, and "fin" for finite horizon problems. Then the user will be prompted for:

- **Input Data File Name.**  
This file includes all parameters, cost/reward value, transition probability matrix, and initial policies.
- **Output Data File Name.**  
If the output file already exist, then the new output will append to it. Otherwise it will create a new output file.
- **Cost Type.** Select one option among FH, discounted cost, and average cost.
- **Method Type.** Select one option among FH, policy iteration and value iteration.

Now we present several examples illustrating the use of S<sup>2</sup>YSCODE.

#### 4.1.1 Example F-1: Queueing Control Problem.

The next simple example is taken from [2], pp. 5-7. The problem is to optimally control the service rate (either "Fast" or "Slow") in a finite capacity queue (max. buffer capacity = 3). We used the following

cost function.

C(i,u)	u = Fast	u = Slow
i = 0	16	1
i = 1	7	7
i = 2	8	8
i = 3	1	9

1. Input Data File: queue.dat.

```

The queue problem input data
ProbTyp
Min
NS, MC, Nstage
4 2 20
G(i,k)
16 1
7 7
8 8
1 9
P(i,j,k), k=1
0.20 0.50 0.30 0.00
0.16 0.44 0.34 0.06
0.00 0.16 0.44 0.40
0.00 0.00 0.16 0.84
P(i,j,k), k=2
0.20 0.50 0.30 0.00
0.06 0.29 0.44 0.21
0.00 0.06 0.29 0.65
0.00 0.00 0.06 0.94
final stage GN
0 0 0 0

```

2. Sample Execution.

```

coyote ] fin
Type the input file : queue.dat
Type the output file : queue.out
Type the discounted rate
- Discounted rate : 0 < Alpha < 1
- No discounted rate : Alpha = 1
- Quit : Alpha = 0
====> Alpha = ? 1
====> You've got the optimal cost!!!
*****
The optimal solution
*****
the problem type : Minimization
# of states : 4
# of actions : 2
# of stage : 10
discounted rate : 1.000

```

=== Stage 0 ===		
1	2	31.89958
2	2	35.13547
3	2	31.28959
4	1	22.36349
=== Stage 1 ===		
1	2	29.45694
2	2	32.70020
3	2	28.86031
4	1	19.93553
=== Stage 2 ===		
1	2	26.99761
2	2	30.25687
3	2	26.42993
4	1	17.50802
=== Stage 3 ===		
1	2	24.50215
2	2	27.79604
3	2	23.99720
4	1	15.08151
=== Stage 4 ===		
1	2	21.92849
2	2	25.29730
3	2	21.55934
4	1	12.65716
=== Stage 5 ===		
1	2	19.18625
2	2	22.71635
3	2	19.11023
4	1	10.23753
=== Stage 6 ===		
1	2	16.08479
2	2	19.95710
3	2	16.63580
4	1	7.82834
=== Stage 7 ===		
1	2	12.24700
2	2	16.81060
3	2	14.10030
4	1	5.44320
=== Stage 8 ===		
1	2	7.10000
2	2	12.82000
3	2	11.39000
4	1	3.12000
=== Stage 9 ===		
1	2	1.00000
2	1	7.00000
3	1	8.00000
4	1	1.00000

STATE CONTROL COST

#### 4.1.2 Example F-2: Mail Fast-delivery Problem.

If a mail item fails to be delivered by the time requested by a customer, this is regarded as a bad performance (i.e., delayed mail) [5]. It is assumed that the number of incoming mail items next week follows a Poisson distribution, which depends on the state of the previous week. The state at a given week is the number of the delayed mail items in that week. This problem has 5 states, which are the amounts of delayed mail items, and 3 actions which are regular, extra part time, and extra full time mail delivery workforce. The transition probabilities are calculated by

$$P_{ij}^k = \sum_{l=j_1(i)}^{j_2(i)} e^{\lambda_i p_k} \frac{(\lambda_i p_k)^l}{l!},$$

where  $j_1(i)$  is lower bound of state  $i$ ,  $j_2(i)$  is upper bound of state  $i$ , and  $p_k$  is the probability that a mail will be delayed if action  $k$  is taken, (see [5] p. 9). The expected immediate reward per stage  $g(i, k)$ , when in state  $i$  and action  $k$  is taken, can be obtained from the relation.

$$g(i, k) = -C_{a,k} + \lambda_i C_{profit} - \sum_{j=1}^5 P_{ij}^k C_{p,j},$$

where  $C_{a,k}$  is the cost under action  $k$ ,  $C_{profit}$  is the profit per unit delivery, and  $C_{p,j}$  is the penalty cost in state  $j$ .

The transition probability matrix  $P_{ij}^k$  and the expected immediate reward per stage  $g(i, k)$  are shown below. The number of all possible stationary policies is  $3^5 = 243$ , and the execution time is less than a second. The initial policy is: actions at all states are set to 1. From our results, it is seen that just a few iterations, i.e. 3, are needed to compute optimal values and policies, within the stated accuracy.

##### 1. Input Data File: fast.dat.

```
The mail fast delivery company problem
ProbTyp
Max
NS, MC, alpha, eps
5 3 .95 1.e-6
G(i,k)
872.51 872.01 868.91
862.64 862.07 858.94
852.78 852.13 848.98
823.16 822.31 819.07
773.78 772.62 769.21
P(i,j,k), k=1
```

```
.0670 .5160 .3682 .0472 .0016
.0710 .5246 .3591 .0439 .0014
.0750 .5330 .3498 .0409 .0013
.0885 .5568 .3212 .0326 .0009
.1157 .5903 .2720 .0216 .0004
P(i,j,k), k=2
.4457 .5117 .0421 .0005 .0
.4554 .5044 .0397 .0005 .0
.4651 .4970 .0374 .0005 .0
.4950 .4736 .0311 .0003 .0
.5461 .4314 .0224 .0001 .0
P(i,j,k), k=3
.7851 .2120 .0028 .0001 .0
.7914 .2060 .0026 .0 .0
.7975 .2001 .0024 .0 .0
.8156 .1825 .0019 .0 .0
.8441 .1546 .0013 .0 .0
initial U
1 1 1 1 1
```

##### 2. Sample Execution.

```
coyote ] dyn
Type the input file : fast.dat
Type the output file : fast.out
Type the cost type
- Discounted cost: D
- Average cost : A
- Quit : Q (1 char): d
Type the method
- Policy Iteration : P
- Value Iteration : V (1 char): p
==> ITERATION # : 1
==> ITERATION # : 2
==> ITERATION # : 3
==> You've got the optimal cost!!!
*****
The optimal solution
*****
the problem type : Maximization
the cost type : Discounted Cost
the method : Policy Iteration
# of iteration : 3
# of states : 5
# of actions : 3
discount rate : 0.950
error bound : 0.1E-5

i k J-uk T(J-uk)
error
1 3 17337.15 17337.15
0.00D+00 2 3 17327.24
17327.24 0.00D+00 3 3
17317.34 17317.34 0.00D+00 4 3
```

```
17287.61 17287.61 0.00D+00 5 2
17238.43 17238.43 0.72D-11
```

```
Type the cost type
- Discounted cost : D
- Average cost : A
- Quit : Q (1 char): a
Type the method
- Policy Iteration : P
- Value Iteration : V (1 char): p
==> ITERATION # : 1
==> ITERATION # : 2
==> ITERATION # : 3
==> You've got the optimal cost!!!
*****
The optimal solution
*****
the problem type : Maximization
the cost type : Average Cost
the method : Policy Iteration
# of iteration : 3
# of states : 5
# of actions : 3
error bound : 0.1E-5
optimal average cost per stage :
Lambda = 0.8667505E+03
```

state	control	H
1	3	98.85049
2	3	88.94882
3	3	79.05120
4	3	49.32537
5	2	0.00000

## 4.2 MATLAB Version

The MATLAB version consists of three programs which are "fin", "val" and "pol". Each one requires four parameters for execution.

- **fin(NS,NA,STG,Alpha):** Finite Horizon function.  
Computes the optimal policy based on the given state transition matrices and the costs(or rewards) at any stage under any action.  
NS : the number of states  
NA : the number of actions  
STG : the number of stages  
Alpha : the discount rate
- **val(NS,NA,Alpha,Eps):** VALUE ITERATION function.  
Computes the optimal policy based on the given state transition matrix and the costs (or

rewards) at any stage under any action.

```
NS : the number of states
NA : the number of actions
Alpha : the discount rate
Eps : the error bound
```

- **pol(NS,NA,Alpha,Eps):** POLICY ITERATION function.  
Computes the optimal policy based on the given state transition matrices and the costs (or rewards) at any stage under any action.  
NS : the number of states  
NA : the number of actions  
Alpha : the discount rate  
Eps : the error bound

When the functions are invoked with all four arguments the user will be prompted for:

1. Problem Title.
2. Problem Type:  
one word "max", "Max", "min", "Min" must be used to denote a maximization or minimization type problem.
3. Input File Format.  
If the input is of the ASCII format choose 'a' or 'A'. If the input is of the MATLAB format choose 'm' or 'M'.
4. Cost (or reward) filename.  
This file MUST have a three character suffix. It should contain a NS x NA matrix where the (i,j)th element is the cost (or reward) of being in state i and implementing action j. If using MATLAB input, this matrix must be named 'G'.
5. Initial conditions filename.  
This file MUST have a three character suffix. It should contain a row vector of length NS. The i'th element denotes the initial action to use when in state i. This vector should contain integers only. If using MATLAB input this vector must be named 'U'.
6. Final stage cost filename ("fin" function only):  
This file MUST have a three character suffix. It should contain a row vector of length NS. The i'th element denotes the final cost of each state to achieve at final stage.
7. Transition probability matrix filename.  
This file MUST have a three character suffix. It should contain as many matrices as there are actions (k). Each matrix should be of size NS x

NS where the (i,j)th element denotes the probability of going from state i to state j while under action k. If using ASCII input, the matrices must be placed one on top of the other (spaces in between are ok). For example, the transition matrix for action 1 should be placed at the beginning with the matrix for action 2 below it and so forth. If using MATLAB input, these matrices must be named 'p1', 'p2', 'p3'...'p(k)' but the relative placement of the matrices is irrelevant.

8. Output filename.

This can be any legal filename. The function's output will be sent here once an optimal solution has been found. Existing filenames can be used since the output will concatenate to the end.

9. Discounted or Average cost analysis.

Use "D" for Discounted cost analysis or "A" for average cost analysis.

The following examples have the same structure as those given for the FORTRAN version. We show the input and output data for comparison purposes.

4.2.1 Example M-1: Queuing Control Problem.

1. Input Data File: The MATLAB input consists of three separate files:

```

**Contents of cost/reward filename
'queue.ggg'
16 1
7 7
8 8
1 9
**Contents of final stage cost filename
'queue.nnn'
0 0 0 0
**Contents of transition probability filename
'fast.ppp'
0.20 0.50 0.30 0.00
0.16 0.44 0.34 0.06
0.00 0.16 0.44 0.40
0.00 0.00 0.16 0.84

0.20 0.50 0.30 0.00
0.06 0.29 0.44 0.21
0.00 0.06 0.29 0.65
0.00 0.00 0.06 0.94

```

2. Sample Execution.

```

>> fin(4,2,10,1)
Problem Title >> Queuing Control
Problem
Problem Type (Max or Min) >> min
(A)scii or (M)atlab input files >> a
Cost/Reward matrix filename >>
queue.ggg
Final Stage Cost filename >> queue.nnn
Transition Probability matrix filename
>> queue.ppp
loading...

```

```

File where you would like the output
to be sent >> queue.out
.....
====> Output sent to: queue.out

```

```

Queuing Control Problem
*****
                The Optimal Solution
*****
the problem type: minimization
# of states : 4
# of stages : 10
# of actions : 2
discount rate : 1.000000
STATE CONTROL    COST
--stage 0
 1      2      31.899581
 2      2      35.135471
 3      2      31.289594
 4      1      22.363492
--stage 1
 1      2      29.456936
 2      2      32.700203
 3      2      28.860307
 4      1      19.935527
--stage 2
 1      2      26.997611
 2      2      30.256867
 3      2      26.429933
 4      1      17.508022
--stage 3
 1      2      24.502150
 2      2      27.796040
 3      2      23.997202
 4      1      15.081511
--stage 4
 1      2      21.928493
 2      2      25.297298
 3      2      21.559342
 4      1      12.657162
--stage 5
 1      2      19.186248

```

2	2	22.716350
3	2	19.110227
4	1	10.237531
--stage 6		
1	2	16.084790
2	2	19.957098
3	2	16.635803
4	1	7.828336
--stage 7		
1	2	12.247000
2	2	16.810600
3	2	14.100300
4	1	5.443200
--stage 8		
1	2	7.100000
2	2	12.820000
3	2	11.390000
4	1	3.120000
--stage 9		
1	2	1.000000
2	1	7.000000
3	1	8.000000
4	1	1.000000

## 5 Translation Modules

**S<sup>2</sup>YSCODE** has the capability of handling either a system equation (e.g., inventory control) or a transition probability matrix (e.g., queueing control) description for the stochastic controlled system of interest. The computational algorithms are naturally formulated in terms of vector values and policies, and transition probability matrices. Therefore, a system equation description has to be *translated* first to that form. This is accomplished by **S<sup>2</sup>YSCODE** via a *Translation Module*. For example, in the FORTRAN version, the user modifies a function statement, thus defining the system equation. With this, and the (conditional) disturbance probability distribution, **S<sup>2</sup>YSCODE** obtains an equivalent probability transition matrix description, in an automated fashion, which is then inputted to the computational modules.

In addition, oftentimes a model of an stochastic system of interest has a continuous variable, and the dynamics are expressed via a system equation. However, to perform computations one needs to find an equivalent (and approximate) model with a finite number of states, and a state transition matrix description of the dynamics. Thus, a *translation scheme* is needed, if the above is to be accomplished in an automated fashion. Although at this point we do not have a fairly general translation scheme of

this nature, we refer to [3] for detail on next one such scheme, applicable to partially observable controlled Markov chains.

## Acknowledgements

This research was partially supported by The Engineering Foundation under grant RI-A-93-10, by a grant from the AT&T Foundation, and by CIMD under grant F93UR031.

## References

- [1] K.J. Aström, *Introduction to Stochastic Control Theory*, Academic Press, New York, 1970.
- [2] D.P. Bertsekas, *Dynamic Programming: Deterministic and Stochastic Models*, Prentice Hall, 1987.
- [3] E. Fernández-Gaucherand, J. Choi and D. Gerhart, "S<sup>2</sup>YSCODE: Stochastic Systems Control and Decision Algorithms Software Laboratory, FORTRAN & MATLAB Versions," *SIE Report*, Systems & Industrial Engineering Department The University of Arizona, 1993.
- [4] D.K. Frederick, R. Kool, M. Rinvall, and A. van den Boom, *ELCS: The Extended List of Control Software 5.0*, monograph, 1992.
- [5] L. Horng and B.F. Lamond, "MDPS-Markov Process System, An Interactive Software Package for the IBM personal computer," *Working Paper no. 89-007*, University of Arizona, 1989.
- [6] P.R. Kumar and P. Varaiya, *Stochastic Systems: Estimation, Identification, and Adaptive Control*, Prentice-Hall, 1986.
- [7] J. Little and C. Moler, *A Preview of MATLAB*, The MathWorks, Inc., 1992.
- [8] *The Student Edition of MATLAB*, The MathWorks, Inc., 1992.
- [9] M.D. Puterman, "Dynamic Programming," *Encyclopedia of Physical Science and Technology, Vol.4*, Academic Press, 1987.
- [10] H.C. Tijms, *Stochastic Modelling and Analysis: A computational approach*, John Wiley and Sons, 1986.