MAQDP
A software package to compute and simulate the optimal control policies for a tandem two station system with batch production and unreliable machines

José António Alves Moreira
Instituto Superior Técnico

October 25, 1998

Contents

Chapter 1

Introduction

The software package presented in this report was developed for studying a system of two tandem machines with batch production and unreliable machines (figure 1.1). This report is the technical companion for the report [1] which describes the modeling and results obtained for this system.

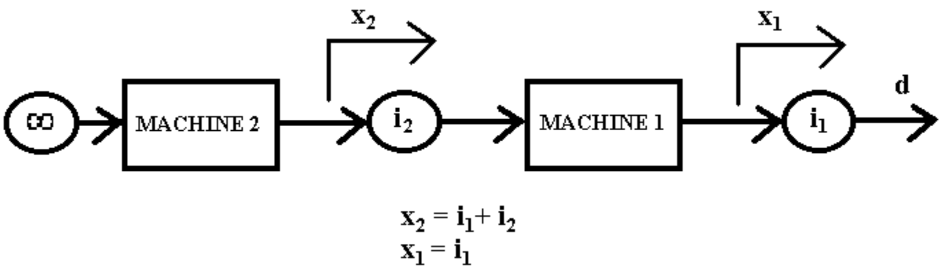

Figure 1.1: Graphic description of the system.

The software contained in this package was developed in **C** [2] and **C++** [3], and was designed to run on any **Unix** System. The software package can use the **X-Window** system for graphic display. Although the program was developed in a **Linux** based platform, it was tested on **Digital** and **Sun** Unix platforms.
The output of the programs contained in this package must be handled by the **Matlab** or **Scilab**[1] programs and by the **Vis5d**[2] program.

The package is composed of the following programs:

Optimo - Computes the optimal control policy.
Gera - Generates groups of optimal control policies.
Simula - Simulates the performance of a control policy.
Suboptimo - Generates a sub-optimal control policy.
Converte - Converts a file generated by **Gera** to the **Vis5d** format.

[1]The Scilab program is available at http://www-rocq.inria/scilab/scilab.html
[2]The Vis5d program is avaliable at http://www.ssec.wisc.edu/~billh/vis5d.html

The package also contains a group of script files written in the **Matlab** and **Scilab** languages to handle the visualization of the output files.
The two most important parts of the software package are the stochastic dynamic programming algorithm and the discrete event simulator. Although the package was designed to solve this concrete problem, the core can be used to solve any problem that requires discrete event simulation and stochastic dynamic programming.
In this report it is described each of the programs contained in the package. More information can be gained by reading the source code of the programs which is fully documented in [4].

Chapter 2

Optimo

The **Optimo** program is responsible for computing the optimal control policy for a specific instance of the problem. It is able to use several algorithms to compute the optimal control policy.

2.1 Description of the program

The **optimo** program receives the necessary data from the command line and from an input file that is created by the user. The command line arguments will have the following form:

optimo option1 option2 (input file) (output data) (output simula)

The command line arguments have the following meaning:

- **option1** - indicates which algorithm to be used to compute the optimal control policy (1 - value iteration, 2 - value iteration with Gauss-Sidel, 3 - policy iteration, 4 - policy iteration with Gauss-Sidel, 5- hybrid algorithm)[1];
- **option2** - Indicates if the user wants to use the X-window display (y - yes, n - no);
- **(input file)** - name of the input file that contains the necessary parameters for the algorithm;
- **(output data)** - name of the output file for the **Matlab** or **Scilab** programs that contains the results of the computations;
- **(output simula)** - name of the output file for the **Simula** program that contains the optimal control policy that was calculated.

The input file must have a standard format that is described in figure 2.1, in which the user inserts the pretended values between the square brackets.
The options presented in the input file have the following meaning:

[1]This algorithms are described in [5] and [6]

```
PARAMETERS USED BY THE OPTIMO PROGRAM
--------------------------------------------------------
VALUE ITERATION ALGORITHM

Error limit                                   [ 1e-5   ]
Number of initial iterations                  [ 100    ]
Number of intermediate iterations             [ 50     ]

HYBRID ALGORITHM

Limit of initial error                        [ 1000   ]
Number of iterations between comparisons      [ 20     ]

MODIFIED POLICY ITERATION ALGORITHM

Error limit                                   [ 0.001  ]
Number of initial iterations                  [ 100    ]
Number of intermediate iterations             [ 50     ]
--------------------------------------------------------
V. cost fun. on the ex. of the state space    [ 1e5    ]
Discount parameter:                           [ 0.01   ]
--------------------------------------------------------
dimension of state space                      [ 2      ]
number of control variables                   [ 2      ]

  >> X1  X2 <<
dim0, superior limit                          [ 50     ]
dim0, inferior limit                          [ -50    ]
dim1, superior limit                          [ 50     ]
dim1, inferior limit                          [ -50    ]

  >> A1 A2 <<
dim2, superior limit                          [ 0      ]
dim2, inferior limit                          [ 0      ]
dim3, superior limit                          [ 0      ]
dim3, inferior limit                          [ 0      ]

  >> L1 L2 <<
dim4, superior limit                          [ 0      ]
dim4, inferior limit                          [ 0      ]
dim5, superior limit                          [ 0      ]
dim5, inferior limit                          [ 0      ]
--------------------------------------------------------
maximum batch of machine 1                    [ 1.0    ]
maximum batch of machine 2                    [ 1.0    ]
demand rate                                   [ 0.5    ]
production rate of machine 1                  [ 1      ]
production rate of machine 2                  [ 1      ]
breakdown rate of machine 1:                  [ 0      ]
breakdown rate of machine 2:                  [ 0      ]
repair rate of machine 1                      [ 0      ]
repair rate of machine 2                      [ 0      ]
--------------------------------------------------------
Holding cost for x1                           [ 0.5    ]
Penalty cost for x1                           [ 2.5    ]
Holding cost for x2                           [ 0.5    ]
Penalty cost for x2                           [ -0.5   ]
--------------------------------------------------------
sim. with unreliable machines (1-yes|0-no)    [ 0      ]
sim. with batch production    (1-yes|0-no)    [ 0      ]
--------------------------------------------------------
OUTPUT
text (1-matlab / 2-scilab)                    [ 1      ]

( 1-yes / 0-no )
control matrix    (modified notation)         [ 1      ]
control matrix    (classic notation)          [ 1      ]
cost matrix       (modified notation)         [ 1      ]
cost matrix       (classic notation)          [ 1      ]
--------------------------------------------------------
```

Figure 2.1: Example of a **Optimo** input file.

Value iteration algorithm

- *Error Limit* - Error limit of the value iteration algorithm;
- *Number of initial iterations* - Number of initial iterations performed before the algorithm starts to compute the error of the cost function value being calculated;
- *Number of intermediate iterations* - Number of iterations between calculations of the error associated to the value of the cost function.

Hybrid algorithm

- *Limit of initial error* - Value of the cost function error value that starts the use of the hybrid algorithm;
- *Number of iterations between comparisons* - Number of iterations between two sucessive comparisons of the control matrix.

Modified policy iteration algorithm

- *Error Limit* - Value of the error limit for the value iteration part of the policy iteration algorithm;
- *Number of initial iterations* - Number of initial iterations performed before the algorithm starts to compute the error of the value of the cost function being calculated for the value iteration part of the policy iteration algorithm;
- *Number of intermediate iterations* - number of iterations between calculations of the error associated to the value of the cost function for the value iteration part of the policy iteration algorithm;
- **V. cost fun. on the ex. of the state space** - Value of the limit that surrounds the state space (frontier value).

Discount parameter - Value of the discount parameter being used.

Dimension of state space - number of variables necessary for the computations (2 - without batch production and with reliable machines, 4- without batch production and unreliable machine, 5- with batch production and unreliable or reliable machines (model II), 4- with batch production and unreliable machines (model I).

Number of control variables - number of control variables necessary to solve the problem (always two).

X1,X2
dim0, superior limit - superior limit for the variable x_1;
dim0, inferior limit - inferior limit for the variable x_1;
dim1, superior limit - superior limit for the variable x_2;
dim1, inferior limit - inferior limit for the variable x_1.

A1,A2
dim2, superior limit - must be one for unreliable machines;
dim2, inferior limit - must be zero for unreliable machines;
dim3, superior limit - must be one for unreliable machines;

dim3, inferior limit - must be zero for unreliable machines.

R1,R2
dim4, superior limit - maximum number of pieces in one batch (model II);
dim4, inferior limit - the value must be zero;
dim5, superior limit - the value must be zero in current version;
dim5, inferior limit - the value must be zero in current version.

sim. with unreliable machines - (1 - simulation with unreliable machines, 0 - simulation with reliable machines).

sim. with batch production - (1 - simulation with batch production, 2 - simulation without batch production).

model - model to be used for the batch production case (1,2).

text - Format for the output file (1 - Matlab, 2 - Scilab).

Control matrix (echelon inventory) - includes in the output file the control matrix in the echelon inventory variables form.

Control matrix (real inventory) - includes in the output file the control matrix in the real inventory variables form.

Cost matrix (echelon inventory) - includes in the output file the cost matrix in the echelon inventory variables form.

Cost matrix (real inventory) - includes in the output file the cost matrix in the real inventory variables form.

In figure 2.2 there is an example of a **Optimo** run, with the option of not using the X-Window display. If the user uses the X-window system, the program will display a window representing the state of the program (figure 2.3). After the required calculations the program will display a graphic output of the resulting optimal control policy (figure 2.4). At the end, the user can close the program by pressing the **exit** box on the program state window.

```
[jose@localhost prog]$ optimo 2 n sim1.o sim1.m sim1.s


                         *** OPTIMO ***
  A PROGRAM TO COMPUTE THE OPTIMAL CONTROL POLICY FOR THE PROBLEM
     OF A TANDEM TWO MACHINE SYSTEM WITH UNRELIABLE MACHINES
                     AND BATCH PRODUCTION

              1997/98 JOSE ANTONIO ALVES MOREIRA
                 INSTITUTO SUPERIOR TECNICO
                     VER 1.0 03/07/1998

 *** STARTING COMPUTATION OF THE OPTIMAL POLICY ***

 Tandem two machine system

PARAMETERS:

Discount parameter:         0.01
Demand rate:                0.05
Production rate of machine 1: 1
Production rate of machine 2: 1

ALGORITHM: Value Iteration using Gauss-Sidel

 *** COMPUTATION OF THE OPTIMAL POLICY TERMINATED ***

[jose@localhost prog]$
```

Figure 2.2: Example of a **Optimo** run.

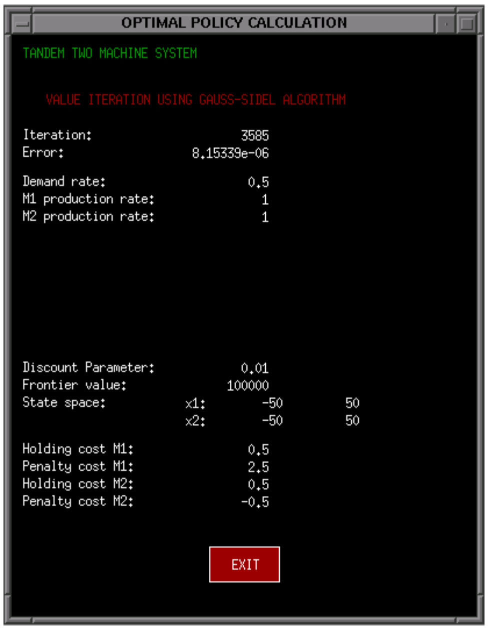

Figure 2.3: Program state display window.

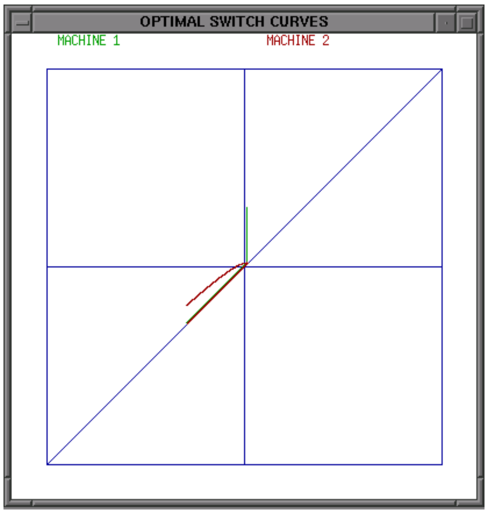

Figure 2.4: Program output graphics display window.

2.2 Algorithms

The algorithms used in this program (except the Hybrid algorithm) were based in [6], which describes in detail the algorithms.

2.2.1 Value iteration

In the value iteration algorithm the user must specify three parameters: the number of initial iterations, the number of intermediate iterations, and the error limit that makes the algorithm stop. Figure 2.5 displays a flow chart that represents the value iteration algorithm. The user can also chose between the normal value iteration algorithm and the Gauss-Sidel variant.

2.2.2 Modified policy iteration

In the modified policy iteration algorithm, the only parameters that the algorithm requires are the ones needed for the value iteration part of the algorithm that were described in the previous section. The user can also chose between using the normal value iteration algorithm or the Gauss-Sidel variant.

2.2.3 Hybrid algorithm

The hybrid algorithm is an extension of the value iteration algorithm in which the decision to stop is not on the value of the error limit, but on the observation that if the control policy doesn't change in a large number of iterations, then there is a large probability that we have obtained the optimal control policy. The flow chart for this algorithm is displayed in figure 2.6.

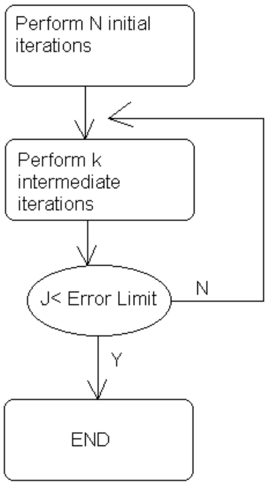

Figure 2.5: Value iteration algorithm flow chart.

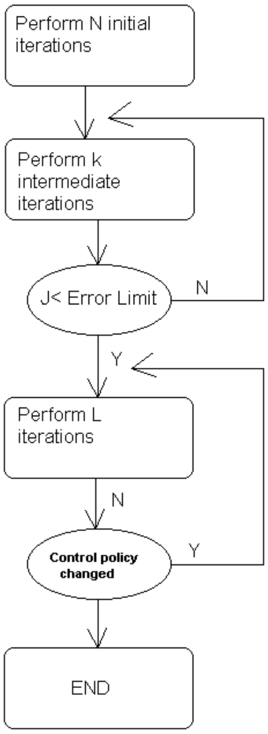

Figure 2.6: Hybrid algorithm flow chart.

2.3 Numerical problems

2.3.1 Value iteration error limit

In the value iteration algorithm, the choice of the error limit is crucial to its performance. A high value of the error limit would imply that there is a large possibility that the control policy obtained is not the optimal control policy. A low value to the error limit would make the program perform more computations than would be needed to find the optimal control policy.
The choice of the error limit will depend of the problem, and a method of trial and error is necessary in the first stages of the algorithm. Table 2.1 and figure 2.7 presents one example that refers to a simulation in which the usual values for the error limit supplied a non-optimal control policy. The optimal policy was obtained by using the policy iteration algorithm.

μ_1	μ_2	λ	β	h'_1	p'_1	h'_2	p'_2	Error limit
1	1	0.9	0.1	0.5	5.5	1	0	1e-5

Table 2.1: Error limit example parameters.

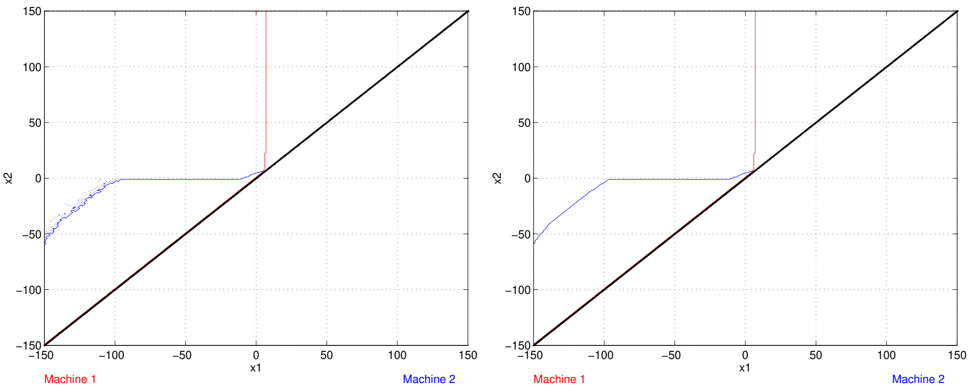

Figure 2.7: Numerical problems with value iteration (left: value iteration , right: policy iteration).

2.3.2 Frontier value

The frontier value arises from the need of using a finite state space for a infinite state space problem. To solve this problem we surround the finite state space by a frontier with a defined value. The problem arises when chossing this value, because it will depend on the size of the finite state space being used and on the parameters of the problem.
A trial and error procedure can be used to find the correct value for the frontier, but an estimate can be found by maximizing the value of the cost function at the extreme points of the finite state space.
Table 2.2 presents the parameters used for computing the optimal control policy of a problem using three different values for the frontier: a correct value, a low and a high value; and the corresponding figures that display the consequences on the switch curves.

μ_1	μ_2	λ	β	h'_1	p'_1	h'_2	p'_2	Front. value	Figure
1	1	0.8	0.01	0.5	2.5	0.5	-0.5	1e4	2.8
1	1	0.8	0.01	0.5	2.5	0.5	-0.5	1e6	2.9 right
1	1	0.8	0.01	0.5	2.5	0.5	-0.5	1e3	2.9 left

Table 2.2: Frontier value example parameters.

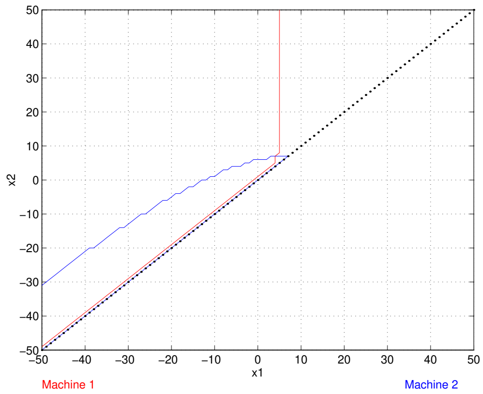

Figure 2.8: Frontier value numerical problems (correct optimal control policy).

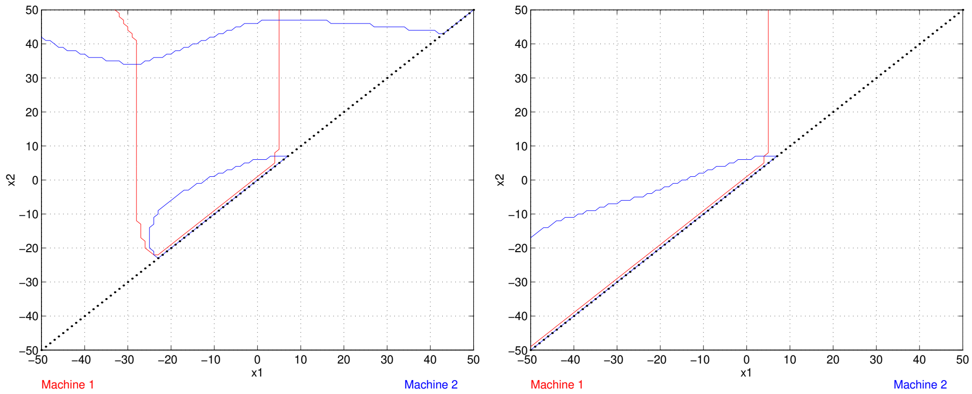

Figure 2.9: Frontier value numerical problems (incorrect optimal control policies) (left: low error limit value, right: high error limit value).

2.3.3 Discount parameter

The discount parameter which is described in [6], represents how much in the future should the algorithm "see" to find the optimal control policy, being the limit value zero. In table 2.3 there is a description of the parameters used on the value iteration algorithm on a specific problem but with decreasing values for the discount parameter. In the corresponding figures, it is possible to observe that when we decrease the discount parameter, the optimal policy will change, but it will start aproaching a limit optimal control policy, which means that it is not necessary to use an infinitesimal value for the discount parameter.

μ_1	μ_2	λ	β	h_1'	p_1'	h_2'	p_2'	Figure
1	1	0.8	0.1	0.5	5.5	0.5	-0.5	2.11 (left)
1	1	0.8	0.01	0.5	5.5	0.5	-0.5	2.10
1	1	0.8	0.001	0.5	5.5	0.5	-0.5	2.11 (right)

Table 2.3: Discount value parameters example.

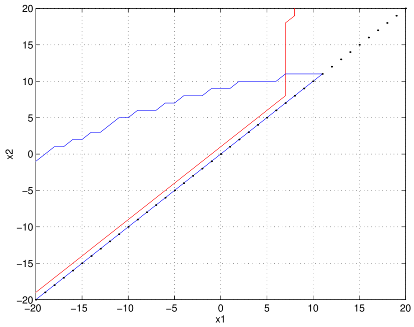

Figure 2.10: Control curves for $\beta = 0.01$.

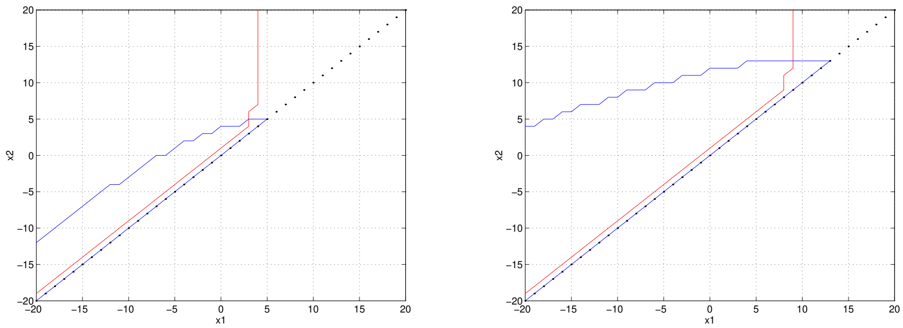

Figure 2.11: Control curves for $\beta = 0.1$ (left) and $\beta = 0.001$ (right).

2.4 Matlab and Scilab interface

The output file generated by the **Optimo** program can be read by the **Matlab** or **Scilab** programs, depending on the option used. The file contains the necessary code to initialize all the variables and includes information on the system parameters and of the algorithms used to calculate the optimal control policy. An example of an output file created with the option of using the **Matlab** program is displayed in figure 2.12. To read the file the user must perform the following operations:

- *Matlab*: The output file must have the extension "filename.m", the user to execute the file inside the **Matlab** environment has only to execute the command **" filename "**;
- *Scilab*: The user can save the file with any extension, and must give the following command inside the **Scilab** environment to execute the output file: **" exec("filename"); "**;

In each of the programs the output file will create some of the following variables depending on the system for which the optimal control policy is being computed:

- **m1** - (size x_1,size x_2) control matrix in the echelon inventory variables form for the machine 1;
- **m1c** - (size i_1,size i_2) control matrix in the real inventory variables form for the machine 1;
- **m2** - (size x_2,size x_2) control matrix in the echelon inventory variables form for the machine 2;
- **m2c** - (size i_1,size i_2) control matrix in the real inventory variables form for the machine 2;
- **cost** - (size x_1,size x_2) cost matrix in the echelon inventory variables form;
- **costc** - (size x_1,size x_2) cost matrix in the real inventory variables form;
- **m1ff** - (size x_1,size x_2) control matrix in the echelon inventory variables form for machine 1 when both machines are up;
- **m1fa** - (size x_1,size x_2) control matrix in the echelon inventory variables form for machine 1 when machine 2 is down;
- **m1ffc** - (size i_1,size i_2) control matrix in the real inventory variables form for machine 1 when both machines are up;
- **m1fac** - (size i_1,size i_2) control matrix in the real inventory variables form for machine 1 when machine 2 is down;
- **m2ff** - (size x_1,size x_2) control matrix in the echelon inventory variables form for machine 2 when both machines are up;
- **m2fa** - (size x_1,size x_2) control matrix in the echelon inventory variables form for machine 2 when machine 1 is down;
- **m2ffc** - (size i_1,size i_2) control matrix in the real inventory variables form for machine 2 when both machines are up;
- **m2fac** - (size i_1,size i_2) control matrix in the real inventory variables form for machine 2 when machine 1 is down;
- **costff** - (size x_1,size x_2) cost matrix in the echelon inventory variables form when both machines are up;
- **costaa** - (size x_1,size x_2) cost matrix in the echelon inventory variables form when both machines are down;
- **costaf** - (size x_1,size x_2) control matrix in the echelon inventory variables form when machine 1 is down;

- **costfa** - (size x_1,size x_2) control matrix in the echelon inventory variables form when machine 2 is down;
- **costffc** - (size i_1,size i_2) cost matrix in the real inventory variables form when both machines are up;
- **costaac** - (size i_1,size i_2) cost matrix in the real inventory variables form when both machines are down;
- **costafc** - (size i_1,size i_2) control matrix in the real inventory variables form when machine 1 is down;
- **costfac** - (size i_1,size i_2) control matrix in the real inventory variables form when machine 2 is down;
- **m2(b)** - (size x_1,size x_2) control matrix in the echelon inventory variables form for machine 2, when machine 1 is producing a batch of b pieces (ex: *m23* 3 pieces);
- **m2(b)c** - (size x_1,size x_2) control matrix in the real inventory variables form for machine 2, when machine 1 is producing a batch of b pieces (ex: *m23c* 3 pieces);
- **cost(b)** - (size x_1,size x_2) cost matrix in the echelon inventory variables form when machine 1 is producing a batch of b pieces (ex: *m23* 3 pieces);
- **cost(b)c** - (size x_1,size x_2) cost matrix in the real inventory variables form when machine 1 is producing a batch of b pieces (ex: *m23* 3 pieces).

The user can then display the data or perform computations over it. The **MAQDP** package includes a collection of files designed to facilitate the visualization of the data obtained from the output file. This collection of files are described in chapter 7.

2.5 Technical details

The output of the program will be given in terms of the real inventory and echelon inventory variables, but the computations are made by using the echelon variables. So it is necessary to transform from the echelon variables state space to the real inventory state space. Figure 2.13 (left) It's gives a graphic description of the echelon state space in terms of the real inventory variables. Since we are only interested on displaying rectangular portions of the state space, it is necessary to discard some of the echelon state space to obtain a rectangular state space for the real inventory variables. This procedure is described in figure 2.13 (right). This means that the output in terms of the real inventory variables will not give all the control state space that was calculated by the algorithm.

```
% TWO TANDEM MACHINES
%
% SPACE DIMENSION X1: -100 100
% SPACE DIMENSION X2: -100 100
%
% DISCOUNT PARAMETER: 0.01
% DEMAND RATE: 0.6
% PRODUCTION RATE OF MACHINE 1: 1
% PRODUCTION RATE OF MACHINE 2: 1
%
% HOLDING COST DE X1: 0.5
% PENALTY COST DE X1: 2.5
% HOLDING COST DE X2: 0.5
% PENALTY COST DE X2: -0.5
%
% m1 - CONTROL MATRIX OF MACHINE 1 (modified notation)
% m2 - CONTROL MATRIX OF MACHINE 2 (modified notation)
% custo - COST FUNCTION VALUE MATRIX (modified notation)
% m1c - CONTROL MATRIX OF MACHINE 1 (classic notation)
% m2c - CONTROL MATRIX OF MACHINE 2 (classic notation)
% custoc - COST FUNCTION VALUE MATRIX (classic notation)

% control matrix for machines 1 and 2 (modified notation)

m1=zeros(201,201);
m2=zeros(201,201);

m1(1,1)=0;
m2(1,1)=1;
m1(2,1)=1;
m2(2,1)=1;
m1(3,1)=1;
m2(3,1)=1;
m1(4,1)=1;
m2(4,1)=1;
m1(5,1)=1;
m2(5,1)=1;
m1(6,1)=1;
m2(6,1)=1;
m1(7,1)=1;
m2(7,1)=1;
m1(8,1)=1;
m2(8,1)=1;
m1(9,1)=1;
m2(9,1)=1;
```

Figure 2.12: Example of a **Optimo** output file for the **Matlab** program.

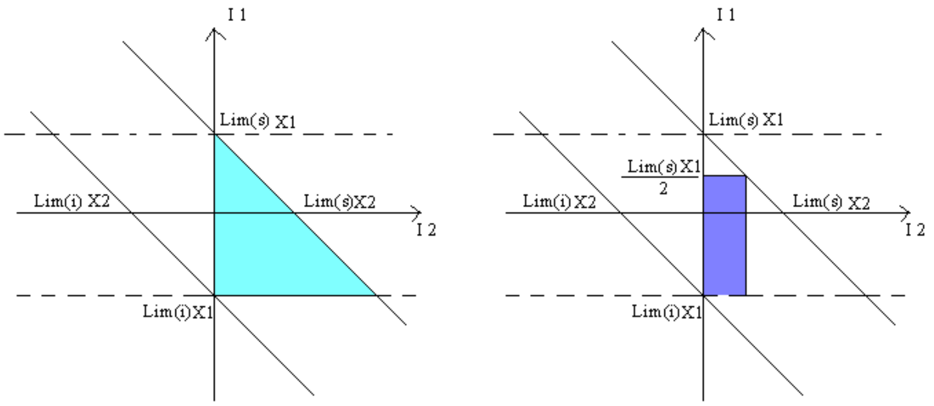

Figure 2.13: Graphic display of the relation betwen the echelon variables state space (x_1, x_2) and the real inventory variables (i_1, i_2).

Chapter 3

Suboptimo

The **Suboptimo** program was created to generate sub-optimal base stock control policies for the problem of a two tandem machines system with reliable machines and without batch production. The sub-optimal control policies are described in figure 3.1, and are defined by two parameters (Z_1, Z_2).

The output file generated by this program is designed to be used by the **Simula** program, and the objective of the development of this program was to validate the results obtained by the **Simula** program (see Sect 4.4). Another possibility is to use this program for calculating the optimal base stock control policy. For that, this program must be used in conjunction with **Simula** for implementing a gradient based search algorithm. This procedure was not implemented by the author.

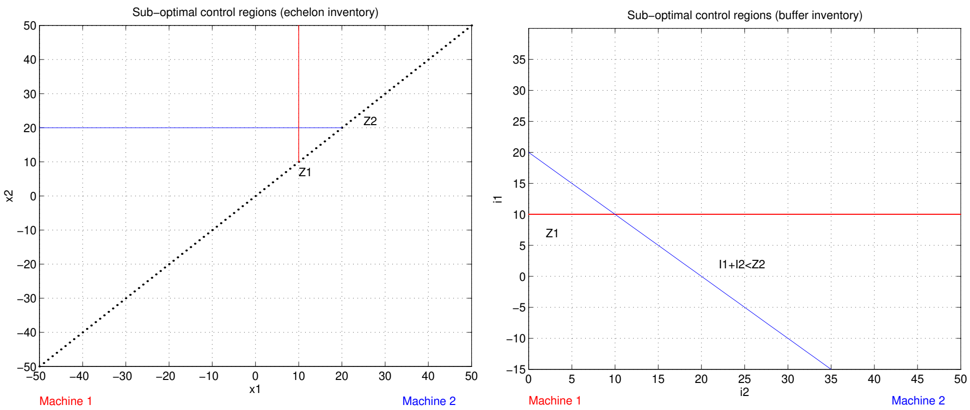

Figure 3.1: Base stock switch curves.

3.1 Description of the program

The **Suboptimo** program can be executed by the command **suboptimo** and it will ask the following questions to the user:

- **Dimension of the state space -** Value for the dimension of the echelon variables state space ($-value \leq x_1 \leq value$; $-value \leq x_2 \leq value$);

- **Name of the output file -** Name of the output file that contains the data to be used in the **Simula** program;
- **Demand rate -** Value of the demand rate;
- **Production rate of machine 1 -** Value of the production rate for machine 1;
- **Production rate of machine 2 -** Value of the production rate for machine 2;
- **Value of Z1 -** Value of the echelon variable x_1 below which machine 1 starts producing parts;
- **Value of Z2 -** Value of the echelon variable x_2 below which machine 2 starts producing parts.

In figure 3.2 there is an example of a **Suboptimo** run.

```
              *** SUBOPTIMO ***
   SUB-OPTIMAL POLICY GENERATION PROGRAM FOR A
           TANDEM TWO MACHINE SYSTEM

        1997/98 JOSE ANTONIO ALVES MOREIRA
           INSTITUTO SUPERIOR TECNICO
               VER 1.0 03/07/1998

Dimension of the state space: 50
Name of the output file: sim.o
Demand rate: 0.8
Production rate of machine 1: 1
Production rate of machine 2: 1
Value of Z1: 3
Value of Z2: 6
```

Figure 3.2: Example of a **Suboptimo** run.

Chapter 4

Simula

The program **Simula** was developed to compute the probability distribution of the echelon and real inventory variables for a specific system and corresponding control policy. The program will also compute the regeneration statistics for the system.
To perform this task the **Simula** program consists of a discrete event system simulator and an exponential distribution generator. With these tools and the optimal control policy that was calculated by the **Optimo** program, it is possible to simulate the performance of the system.

4.1 Description of the program

The **simula** program receives the necessary data from the command line, from an input file created by the user and from an output file of the **Optimo** program. The command line arguments will have the following form:

simula option (input control file) (input parameters file) (output data)

The command line arguments have the following meaning:

- **option -** indicates which type of simulation algorithm to be used (**n** - normal algorithm **t** - use the fast algorithm (no errors or regeneration statistics , **x** - use the X-Window display);
- **input control file -** name of the input file that contains the control policy that was generated by the **Optimo** program.;
- **input parameters file -** name of the input file that contains the necessary parameters for the simulation algorithm;
- **output data -** name of the output file for the **Matlab** or **Scilab** program that contains the results of the computations.

The parameters input file must have a standard format that is exemplified in figure 4.1, in which the user inserts the desired values between the square brackets.
The options presented in the parameters input file have the following meaning:

```
PARAMETERS FILE FOR THE SIMULA PROGRAM

---------------------------------------------------------------

SEEDS OF THE EXPONENTIAL DISTRIBUTION GENERATOR

Demand rate                                         [ 0.1234 ]
Production rate of machine 1                        [ 0.1258 ]
Production rate of machine 2                        [ 0.2186 ]
Failure rate of machine 1                           [ 0.1566 ]
Failure rate of machine 2                           [ 0.2867 ]
Repair rate of machine 1                            [ 0.3545 ]
Repair rate of machine 2                            [ 0.2978 ]

Variation of the demand rate                        [ 0.11   ]
Variation of the production rate of machine 1       [ 0.12   ]
Variation of the production rate of machine 2       [ 0.13   ]
Variation of the failure rate of machine 1          [ 0.14   ]
Variation of the failure rate of machine 2          [ 0.15   ]
Variation of the repair rate of machine 1           [ 0.16   ]
Variation of the repair rate of machine 2           [ 0.17   ]

---------------------------------------------------------------

Simulation time                                     [ 1e6    ]
Number of simulations                               [ 10     ]

---------------------------------------------------------------

Log file ?  (0-no,1-yes)                            [ 0      ]

---------------------------------------------------------------

randomize number                                    [ 1000   ]

---------------------------------------------------------------

OUTPUT

text (1-matlab / 2-scilab)                          [ 2      ]

( 1-yes / 0-no )

Density (modified notation)                         [ 1      ]
Marginal density (modified notation)                [ 1      ]
Marginal distribution (modified notation)           [ 1      ]
Density (classic notation)                          [ 1      ]
Marginal density (classic notation)                 [ 1      ]
Marginal distribution (classic notation)            [ 1      ]

Simulation errors statistics                        [ 1      ]

regenerations statistics                            [ 0      ]

---------------------------------------------------------------
```

Figure 4.1: Example of a **Simula** input file.

- **Seeds of the exponential distribution generator -** The user must insert the value of the seed for the exponential distribution generator for each of the events necessary to the simulation and also the variation applied to the seeds for each simulation (note: $0 \leq seed \leq 1$ and seed(n+1)=seed(n)+variation);
- **Simulation time -** Value of the simulation time;
- **Number of simulations -** Number of simulations to be performed;
- **Log file -** produces an output file with a complete log of the events during the simulation (0 - no, 1 - yes);
- **Randomize number -** Number of iterations performed on the seeds before they are used;
- **Output -** Which variables should be included in the output file.

In figure 4.3 there is an example of a simulation run with the option (**n**). If the option (**x**) is used, the program will display a output window describing the state of the simulation that is displayed in figure 4.4. In figure 4.2 there is an example of part of an output log file.

4.2 Matlab and Scilab interface

The output file generated by the **Simula** program can be read by the **Matlab** or **Scilab** programs, depending on the option used. The file contains the necessary code to initialize all the variables and includes information on the system that was simulated and on the performance of the discrete event simulator. An example of an output file created with the option of using the **Matlab** program is displayed in figure 4.5. To read the file the user must perform the following operations:

- *Matlab*: The output file must have the extension "filename.m", to run the file inside the **Matlab** environment has only to execute the command **" filename "**;
- *Scilab*: The user can save the file with any extension, and must give the following command inside the **Scilab** environment to execute the output file: **" exec("filename"); "**.

In each of the programs the output file will create the following variables:

- **X -** (size x_1 , size x_2) contains the bi-dimensional probability density of the echelon inventory variables;
- **X1 -** (size x_1 , 2) contains the marginal probability density of the variable x_1;
- **X2 -** (size x_2 , 2) contains the marginal probability density of the variable x_2;
- **XI1 -** (size x_1 , 2) contains the marginal probability distribution of the variable x_1;
- **XI2 -** (size x_2 , 2) contains the marginal probability distribution of the variable x_2;
- **I -** (size 2_1 , size i_2) contains the bi-dimensional probability density of the real inventory variables;
- **I1 -** (size i_1 , 2) contains the marginal probability density of the variable i_1;
- **I2 -** (size i_2 , 2) contains the marginal probability density of the variable i_2;

```
event
ITER:   0 TIME:      1.21745 (* deman event         *) estado:   -1   -1    1    1
ITER:   0 TIME:      1.21745 (* next demand event   *) state:   -1   -1    1    1 end: 4.58904
ITER:   0 TIME:      1.21745 (* Mac. 1 stop         *) state:   -1   -1    1    1
ITER:   0 TIME:      1.21745 (* start mac. 2 prod.  *) state:   -1   -1    1    1 end: 1.61158
event
ITER:   0 TIME:      1.61158 (* end. prod. mac. 2   *) state:   -1    0    1    1 end: 1.61158
ITER:   0 TIME:      1.61158 (* start prod. mac. 1  *) state:   -1    0    1    1 end: 2.02232
ITER:   0 TIME:      1.61158 (* start prod. mac. 2  *) state:   -1    0    1    1 end: 2.61577
event
ITER:   0 TIME:      2.02232 (* end prod mac. 1     *) state:    0    0    1    1 end: 2.61577
ITER:   0 TIME:      2.02232 (* mac. 1 stop         *) state:    0    0    1    1
event
ITER:   0 TIME:      2.61577 (* end. prod. mac. 2   *) state:    0    1    1    1 end: 2.61577
ITER:   0 TIME:      2.61577 (* start prod. mac. 1  *) state:    0    1    1    1 end: 2.97326
ITER:   0 TIME:      2.61577 (* start prod. mac. 2  *) state:    0    1    1    1 end: 4.67058
event
ITER:   0 TIME:      2.97326 (* end prod mac. 1     *) state:    1    1    1    1 end: 4.67058
ITER:   0 TIME:      2.97326 (* mac. 1 stop         *) state:    1    1    1    1
event
ITER:   0 TIME:      4.58904 (* deman event         *) state:    0    0    1    1
ITER:   0 TIME:      4.58904 (* next demand event   *) state:    0    0    1    1 end: 5.20189
ITER:   0 TIME:      4.58904 (* Mac. 1 stop         *) state:    0    0    1    1
event
ITER:   0 TIME:      4.67058 (* end. prod. mac. 2   *) state:    0    1    1    1 end: 5.20189
ITER:   0 TIME:      4.67058 (* start prod. mac. 1  *) state:    0    1    1    1 end: 5.77281
ITER:   0 TIME:      4.67058 (* start prod. mac. 2  *) state:    0    1    1    1 end: 6.03953
event
ITER:   0 TIME:      5.20189 (* deman event         *) state:   -1    0    1    1
ITER:   0 TIME:      5.20189 (* next demand event   *) state:   -1    0    1    1 end: 5.99534
event
ITER:   0 TIME:      5.77281 (* end prod mac. 1     *) state:    0    0    1    1 end: 5.99534
ITER:   0 TIME:      5.77281 (* mac. 1 stop         *) state:    0    0    1    1
event
ITER:   0 TIME:      5.99534 (* deman event         *) state:   -1   -1    1    1
ITER:   0 TIME:      5.99534 (* next demand event   *) state:   -1   -1    1    1 end: 6.36076
ITER:   0 TIME:      5.99534 (* Mac. 1 stop         *) state:   -1   -1    1    1
event
ITER:   0 TIME:      6.03953 (* end. prod. mac. 2   *) state:   -1    0    1    1 end: 6.36076
ITER:   0 TIME:      6.03953 (* start prod. mac. 1  *) state:   -1    0    1    1 end: 8.02605
ITER:   0 TIME:      6.03953 (* start prod. mac. 2  *) state:   -1    0    1    1 end: 7.37143
event
ITER:   0 TIME:      6.36076 (* deman event         *) state:   -2   -1    1    1
ITER:   0 TIME:      6.36076 (* next demand event   *) state:   -2   -1    1    1 end: 6.73603
event
ITER:   0 TIME:      6.73603 (* deman event         *) state:   -3   -2    1    1
ITER:   0 TIME:      6.73603 (* next demand event   *) state:   -3   -2    1    1 end: 9.03087
event
ITER:   0 TIME:      7.37143 (* end. prod. mac. 2   *) state:   -3   -1    1    1 end: 9.03087
ITER:   0 TIME:      7.37143 (* start prod. mac. 2  *) state:   -3   -1    1    1 end: 7.83716
event
ITER:   0 TIME:      7.83716 (* end. prod. mac. 2   *) state:   -3    0    1    1 end: 7.83716
ITER:   0 TIME:      7.83716 (* start prod. mac. 2  *) state:   -3    0    1    1 end: 8.78212
event
ITER:   0 TIME:      8.02605 (* end prod mac. 1     *) state:   -2    0    1    1 end: 8.78212
ITER:   0 TIME:      8.02605 (* start prod mac. 1   *) state:   -2    0    1    1 end: 8.81779
event
ITER:   0 TIME:      8.78212 (* end. prod. mac. 2   *) state:   -2    1    1    1 end: 8.81779
ITER:   0 TIME:      8.78212 (* start prod. mac. 2  *) state:   -2    1    1    1 end: 9.01591
event
ITER:   0 TIME:      8.81779 (* end prod mac. 1     *) state:   -1    1    1    1 end: 9.01591
ITER:   0 TIME:      8.81779 (* start prod mac. 1   *) state:   -1    1    1    1 end: 9.49919
event
ITER:   0 TIME:      9.01591 (* end. prod. mac. 2   *) state:   -1    2    1    1 end: 9.49919
ITER:   0 TIME:      9.01591 (* start prod. mac. 2  *) state:   -1    2    1    1 end: 10.441
event
ITER:   0 TIME:      9.03087 (* deman event         *) state:   -2    1    1    1
ITER:   0 TIME:      9.03087 (* next demand event   *) state:   -2    1    1    1 end: 10.5632
event
ITER:   0 TIME:      9.49919 (* end prod mac. 1     *) state:   -1    1    1    1 end: 10.5632
ITER:   0 TIME:      9.49919 (* start prod mac. 1   *) state:   -1    1    1    1 end: 9.67462
```

Figure 4.2: Example of a **Simula** log output file.

```
                    *** SIMULA **
      SIMULATION OF THE OPTIMAL POLICY FOR A TANDEM TWO
   MACHINE SYSTEM WITH UNRELIABLE MACHINES AND BATCH PRODUCTION

              1997/98 JOSE ANTONIO ALVES MOREIRA
                 INSTITUTO SUPERIOR TECNICO
                     VER 1.0 03/07/1998

 *** STARTING SIMULATION ***

 >> Parameters of the control policy <<

Demand rate:                    0.05
Production rate of machine 1:   1
Production rate of machine 2:   1
Discount parameter:             0.01
State space
  X1                            50  -50
  X2                            50  -50
Holding cost M1:                0.5
Penalty cost M1:                4.5
Holding cost M2:                0.5
Penalty cost M2:                -0.5

 >> SIMULATION PARAMETERS <<

Number of simulations:          3
Simulation time:                100000

Iteracao: 0 1 2

 *** SIMULATION COMPLETE ***

 >> SIMULATION RESULTS <<

 << Number of events in simulation >>

  Simulation: 0 : 5051
  Simulation: 1 : 4855
  Simulation: 2 : 5039

Cost average:                 0.445271
Cost variance:                0.000873234
Confidence interval at 95%:   0.0539173

 << Number of regenerations in simulation >>

  Simulation: 0 : 4533
  Simulation: 1 : 4416
  Simulation: 2 : 4545

Regenerations average:        22.2318
Rgenerations variance:        0.128901
Confidence interval at 95%:   0.655075
```

Figure 4.3: Example of a **Simula** run.

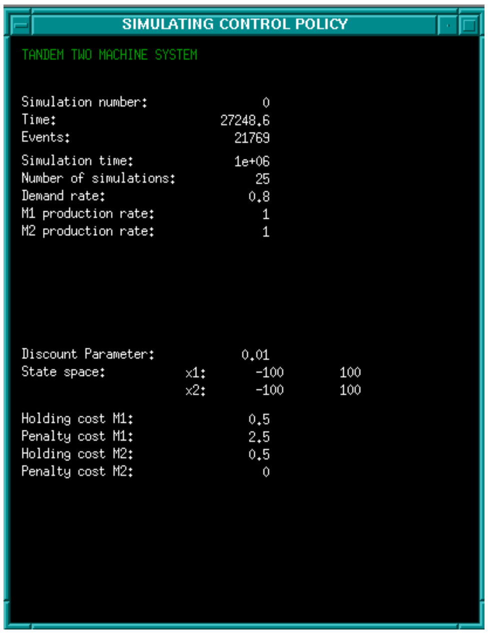

Figure 4.4: **Simula** output window.

- **II1 -** (size i_1 , 2) contains the marginal probability distribution of the variable i_1;
- **II2 -** (size i_2 , 2) contains the marginal probability distribution of the variable i_2;
- **cost_average -** (number of simulations , 2) cost average and the number of events of each of the performed simulations;
- **t_cost_average -** total cost average of all the simulations;
- **cost_variance -** value of the cost variance;
- **cost_confidence_interval -** value of the 95% confidence interval for the average cost value;
- **reg_x1 -** value of the regeneration point for the variable x_1;
- **reg_x2 -** value of the regeneration point for the variable x_2;
- **regenerations_average -** (number of simulations , 2) regeneration time average and number of regenerations of each of the performed simulations;
- **t_regenerations_average -** total regeneration time average of all the simulations;
- **regenerations_variance -** value of the regeneration time average variance;
- **regenerations_confidence_interval -** value of the 95% confidence for the average regeneration time.

The user can then display the data or perform computations over it. The **MAQDP** package includes a collection of files designed to facilitate the visualization of the data obtained from the output file. This collection of files is described in chapter 7.

```
% SIMULATION RESULTS
%
% OPTIMAL CONTROL POLICY PARAMETERS
% DISCOUNT PARAMETER: 0.01
% DEMAND RATE: 0.8
% PRODUCTION RATE OF MACHINE 1: 1
% PRODUCTION RATE OF MACHINE 2: 1
%
% SIMULATION PARAMETERS
% NUMBER OF SIMULATIONS: 25
% SIMULATION TIME:   1e+06


% density (modified notation)

X=zeros(201,201);

X(1,1)=0;
X(2,1)=0;
X(3,1)=0;
X(4,1)=0;
X(5,1)=0;
X(6,1)=0;
X(7,1)=0;
X(8,1)=0;
X(9,1)=0;
X(10,1)=0;
X(11,1)=0;
X(12,1)=0;
X(13,1)=0;
X(14,1)=0;
X(15,1)=0;
X(16,1)=0;
X(17,1)=0;
X(18,1)=0;
X(19,1)=0;
X(20,1)=0;
X(21,1)=0;
X(22,1)=0;
X(23,1)=0;
X(24,1)=0;
```

Figure 4.5: Example of a **Simula** output file for **Matlab**.

4.3 Validation of the results

When dealing with the data generated by the simulation it is necessary to have a measure of the results accuracy . To obtain that measure the program computes the 95% confidence interval to each of the estimators that the programs computes (average cost value and regeneration time average). The method used is called the method of independent replications. The simulation is repeated a total of R times, each run using a different random number generator seed.

First it is necessary to calculate the estimator by the following equation:

$$\hat{\theta} = \frac{1}{R}\sum_{i=0}^{R} \theta_i, \tag{4.1}$$

being R the number of simulations and θ_i the cost or regeneration time average for each simulation. The variance for the estimator will be given by the expression

$$\hat{\sigma^2}(\hat{\theta}) = \frac{1}{(R-1)R}\sum_{i=0}^{R} (\hat{\theta} - \theta_i)^2 \tag{4.2}$$

The 95% confidence interval for the estimator $\hat{\theta}$ is given by the expression

$$\hat{\theta} - t_{0.005,R}.\hat{\sigma}(\hat{\theta}) \leq \hat{\theta} \leq \hat{\theta} + t_{0.005,R}.\hat{\sigma}(\hat{\theta}), \tag{4.3}$$

Where $t_{0.005,R}$ represents the value of the t-student distribution for the value 0.05 with R degrees of freedom. The program will output the value $t_{0.005,R}.\hat{\sigma}(\hat{\theta})$ has the 95% confidence interval for the estimator $\hat{\theta}$.

When using this approach there are some assumptions being made regarding the distribution of the output data. Although there is no evidence regarding that, the simplicity of this test makes it ideal to use for a rough estimate of the estimator error. A detailed description of the procedure is available in [7].

Chapter 5

Gera

The **Gera** program was created to address the need of observing the influence of a set of parameters on the switch curves of the optimal control for a given system. Since for that it is necessary to obtain a set of switch curves to observe the influence of the parameters, the **Gera** program gives the user the possibility of generating a set of switch curves in which a group of parameters is being changed for each curve.
The **Gera** program will generate an output file that can be converted by a special program (**Converte**) to a format that can be read by the **Vis5d** program, which will be briefly presented in this chapter.
The evolution of the switch curves will be possible in two dimensions. The first dimension corresponds to sequencing a group of switch curves to form a three dimensional graph, the second dimension corresponds to creating a sequence of three dimensional graphs along time. The procedure can be illustrated by the following example:

$\longleftarrow$ **3D Graph** $\longrightarrow$
$curve(x)$ $curve(x+\beta)$ $curve(x+2\times\beta)$ $curve(x+3\times\beta)$ - **time unit 1**
$curve(x+\mu)$ $curve(x+\beta+\mu)$ $curve(x+2\times\beta+\mu)$ $curve(x+3\times\beta+\mu)$ - **time unit 2**
$curve(x+2\times\mu)$ $curve(x+\beta+2\times\mu)$ $curve(x+2\times\beta+2\times\mu)$ $curve(x+3\times\beta+2\times\mu)$ - **time unit 3**
β - **variation applied to the parameters in space**
μ - **variation applied to the parameters in time**

5.1 Description of the program

The **Gera** program receives the necessary data from two input parameter files created by the user.
The command line arguments will have the following form:

gera (input optimo) (input gera) (output converte)

The command line arguments have the following meaning:

- **input optimo -** name of the input file in the format used by the **Optimo** program and corresponds to the parameters of the dynamic programming algorithm and the parameters of the initial switch curve (this file is called the **Optimo** parameters file);

- **input gera -** name of the input file with the parameters necessary to the gera program (this file is called the **Gera** parameters file);
- **output converte -** name of the output file to be used by the **Converte** program.

An example of a **Gera** parameters file is displayed in figure 5.1 and can be divided in the following four sections:

- *first section:* corresponds to the parameter evolution of the curves that will be displayed sequencly in one 3D graph. The user must specify the number of curves that will be used;
- *second section:* corresponds to the parameters evolution of the curves that will be displayed for each unit of time. The user must specify the number of time units that will be used;
- *third section:* in this section the user must chose the dynamic programming algorithm to be used for the optimal policy computation of each switch curve;
- *fourth section:* in this section the user must chose the type of output data that will be displayed in the **Vis5d** program.

5.2 Vis5d Interface

The output file generated by the **Gera** program consists of a collection of switch curves corresponding to the computation of several optimal control policies. To visualize this curve in a unified way the output file must be converted to the **Vis5d** format. For that, the user must use the **Converte** program that is described in the next chapter. An example of usage of the **Gera** and **Converte** program to visualize a collection of switch curves would be:

gera sim1.o sim1.gera sim1.converte
converte sim1.converte sim1.v5d
vis5d sim1.v5d -box 2 2 2

This section presents some instructions on how to use the **VIS5d** program to visualize the output of the **Gera** program. Complete instructions can be found in [8]. In figure 5.3 there is an example of the **VIS5D** program. To call the program the user is advised to use the following command line instruction:

vis5d filename -box 2 2 2

After starting the program the user can rotate the visualization cube in the output window using the mouse. After setting the pretended view angle the user can view the entire collection of the generated switch curves. For that he must press the adequate button under the bar named *VOLUME*. To view one special switch curve the user must press the adequate button under the bar *CONTOUR HORIZ*.

```
PARAMETERS USED BY THE GERA PROGRAM
------------------------------------------------------------

Variation of the parameters on the coordinate Z

Variation of the holding cost of X1 (h1)        [ 0.1      ]
Variation of the penalty cost of X1 (p1)        [ 4        ]
Variation of the holding cost of X2 (h2)        [ 0.1      ]
Variation of the holding cost of X2 (p2)        [ 4        ]
Variation of the demand rate                    [ 0        ]
Variation of the production rate of machine 1   [ 0        ]
Variation of the production rate of machine 2   [ 0        ]
Variation of the failure rate of machine 1      [ 0        ]
Variation of the failure rate of machine 2      [ 0        ]
Variation of the repair rate of machine 1       [ 0        ]
Variation of the repair rate of machine 2       [ 0        ]

number of iterations in the coordinate Z        [ 1        ]

------------------------------------------------------------

Variation of the parameters in time

Variation of the holding cost of X1 (h1)        [ 0.1      ]
Variation of the penalty cost of X1 (p1)        [ 4        ]
Variation of the holding cost of X2 (h2)        [ 0.1      ]
Variation of the holding cost of X2 (p2)        [ 4        ]
Variation of the demand rate                    [ 0        ]
Variation of the production rate of machine 1   [ 0        ]
Variation of the production rate of machine 2   [ 0        ]
Variation of the failure rate of machine 1      [ 0        ]
Variation of the failure rate of machine 2      [ 0        ]
Variation of the repair rate of machine 1       [ 0        ]
Variation of the repair rate of machine 2       [ 0        ]

Number of iterations in time                    [ 1        ]

------------------------------------------------------------

 1 - Value Iteration
 2 - Value Iteration using Gauss-Sidel
 3 - Modified Policy Iteration
 4 - Modified Policy Iteration using Gauss-Sidel
 5 - Hybrid algorithm


Algorithm to use                                [ 1        ]

------------------------------------------------------------

OUTPUT

 1 - control matrix (echelon inventory)
 2 - control matrix (echelon inventory)
 3 - cost matrix (real inventory)
 4 - cost matrix (real inventory)

option                                          [ 1        ]
```

Figure 5.1: Example of a **Gera** input file.

*** GERA ***

A PROGRAM TO GENERATE OPTIMAL CONTROL POLICIES FOR THE
TANDEM TWO MACHINE SYSTEM WITH UNRELIABLE MACHINES AND BATCH PRODUCTION

1997/98 JOSE ANTONIO ALVES MOREIRA
INSTITUTO SUPERIOR TECNICO
VER 1.0 03/07/1998

*** STARTING GENERATION ***

OUTPUT: Control matrix (modified notation)

Time	It. Z	Cost h1	Cost p1	Cost h2	Cost p2	Dem. R.	M1 p.	M2 p.	M1 f.	M2 f.	M1 r.	M2 r.
0	0	0.5	1.5	0.5	-0.5	0.8	1	1	0	0	0	0
0	1	0.5	2.97	0.5	-0.5	0.8	1	1	0	0	0	0
0	2	0.5	4.44	0.5	-0.5	0.8	1	1	0	0	0	0
0	3	0.5	5.91	0.5	-0.5	0.8	1	1	0	0	0	0
0	4	0.5	7.38	0.5	-0.5	0.8	1	1	0	0	0	0
0	5	0.5	8.85	0.5	-0.5	0.8	1	1	0	0	0	0
0	6	0.5	10.32	0.5	-0.5	0.8	1	1	0	0	0	0
0	7	0.5	11.79	0.5	-0.5	0.8	1	1	0	0	0	0
0	8	0.5	13.26	0.5	-0.5	0.8	1	1	0	0	0	0
0	9	0.5	14.73	0.5	-0.5	0.8	1	1	0	0	0	0
0	10	0.5	16.2	0.5	-0.5	0.8	1	1	0	0	0	0
0	11	0.5	17.67	0.5	-0.5	0.8	1	1	0	0	0	0
0	12	0.5	19.14	0.5	-0.5	0.8	1	1	0	0	0	0
0	13	0.5	20.61	0.5	-0.5	0.8	1	1	0	0	0	0
0	14	0.5	22.08	0.5	-0.5	0.8	1	1	0	0	0	0
0	15	0.5	23.55	0.5	-0.5	0.8	1	1	0	0	0	0
0	16	0.5	25.02	0.5	-0.5	0.8	1	1	0	0	0	0
0	17	0.5	26.49	0.5	-0.5	0.8	1	1	0	0	0	0
0	18	0.5	27.96	0.5	-0.5	0.8	1	1	0	0	0	0
0	19	0.5	29.43	0.5	-0.5	0.8	1	1	0	0	0	0
0	20	0.5	30.9	0.5	-0.5	0.8	1	1	0	0	0	0
0	21	0.5	32.37	0.5	-0.5	0.8	1	1	0	0	0	0
0	22	0.5	33.84	0.5	-0.5	0.8	1	1	0	0	0	0
0	23	0.5	35.31	0.5	-0.5	0.8	1	1	0	0	0	0
0	24	0.5	36.78	0.5	-0.5	0.8	1	1	0	0	0	0
0	25	0.5	38.25	0.5	-0.5	0.8	1	1	0	0	0	0
0	26	0.5	39.72	0.5	-0.5	0.8	1	1	0	0	0	0
0	27	0.5	41.19	0.5	-0.5	0.8	1	1	0	0	0	0
0	28	0.5	42.66	0.5	-0.5	0.8	1	1	0	0	0	0
0	29	0.5	44.13	0.5	-0.5	0.8	1	1	0	0	0	0
0	30	0.5	45.6	0.5	-0.5	0.8	1	1	0	0	0	0
0	31	0.5	47.07	0.5	-0.5	0.8	1	1	0	0	0	0
0	32	0.5	48.54	0.5	-0.5	0.8	1	1	0	0	0	0
0	33	0.5	50.01	0.5	-0.5	0.8	1	1	0	0	0	0

*** GENERATION COMPLETE ***

Figure 5.2: Example of a **Gera** run.

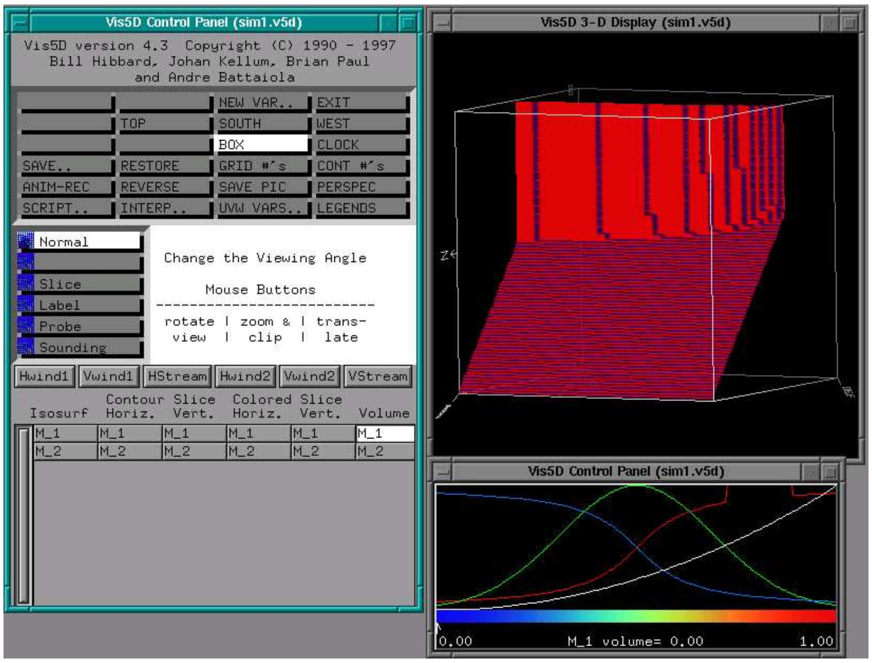

Figure 5.3: Vis5d display window.

Chapter 6

Converte

The **converte** program was designed to converte files generated by the **Gera** program to the **Vis5d** format. The program was written in **C** and most of the code is based on the conversion source code supplied in the **vis5d** package [8].

6.1 Description of the program

The program was designed by using the source code of an exemplified conversion program in the **Vis5d** package and adding the necessary code for the program to handle the **Gera** output files. In the source code you will find a complete description of the process, since the **Vis5d** library offers a very powerful collection of conversion functions to simplify the process. To use the program it is necessary to have an output file from the **Gera** program, and then give the following command line instruction.

converte (input file from gera) (output file to vis5d)

The command line arguments have the following meaning:

- **input file from gera -** name of the output file from the **Gera** program that contains the collection of switch curves;
- **output file to vis5d -** name of the output file to be processed by the **Vis5d** program.

One instance of using this program is exemplified in the figure 6.1.
The program will verify if the input file is a valid file generated by the **Gera** program. Because the program uses libraries from the **Vis5d** package there are the following limitations:

- There is a maximum number for the grid space, which depends on the **Vis5d** configuration you are using when you compile the **Converte** source code;
- If your grid space is larger than allowed, then the program will exit and inform the user of the problem.

In [8] there is a description of a series of utility programs that can be used to process **Vis5d** files. For example, it is possible to merge several **Vis5d** in to a single file.

```
[jose@localhost prog]$ converte sim1.gera sim1.v5d

                        *** CONVERTE ***
    PROGRAM TO CONVERTE OUTPUT FILES FROM THE GERA PROGRAM
                  TO A VIS5D INPUT FILE

               1997/98 JOSE ANTONIO ALVES MOREIRA
                 INSTITUTO SUPERIOR TECNICO
                    VER 1.0 03/07/1998

>> Generation parameters <<

 Number of colums:             101
 Number of lines:              101
 Number of 2D curves:          34
 Number of time steps:         1

[jose@localhost prog]$
```

Figure 6.1: Example of a run of the **Converte** program.

Chapter 7

Matlab programs

This chapter presents a collection of functions created in the **Matlab** script language to facilitate the visualization of the results obtained from the **Optimo** and **Simula** programs. These functions were saved in corresponding files so that the user must first load the output file and then call the corresponding function to visualize the data contained in the output file. The complete source code for each of the functions presented in this chapter is described in [4].

7.1 Visualization of results from the Optimo program

7.1.1 slsa

The **slsa** function was designed to visualize the results from the **optimo** program when we are treating a system consisting of a tandem two reliable machine system without batch production. The function **slsa** must be called after loading the output file obtained from the **optimo** program. The function has the following structure:

slsa(size,m1,m2,m1c,m2c)

For example, if we have an output file from the **Optimo** program in which we used a state space with the following dimensions for each variable

$$(-50 \leq x_1 \leq 50) \; ; \; (-50 \leq x_2 \leq 50)$$

and if the output file has the filename *"simul.m"* then the following set of command line instructions can be used to visualize the results that are displayed in figure 7.1.

simul
slsa(50,m1,m2,m1c,m2c)

7.1.2 slca

The **slca** function was designed to visualize the results from the **Optimo** program when we are treating a system consisting of a tandem two machine system without batch production and unreliable machines. The function **slca** must be called after loading the output file obtained from the **Optimo** program.
The function has the following structure:

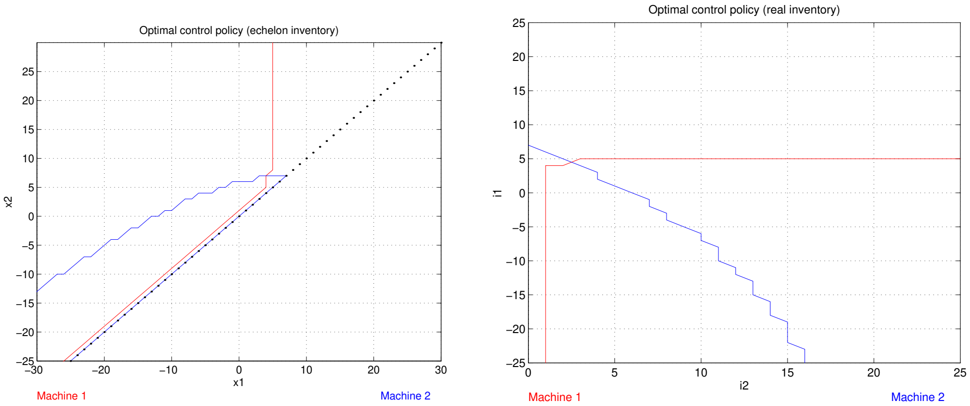

Figure 7.1: **Matlab** output for the file simul.m using slsa.

$$slca(size,m1ff,m1fa,m2ff,m2af,m1ffc,m1fac,m2ffc,m2afc)$$

For example, if we have an output file from the **Optimo** with the following state space

$$(-50 \le x_1 \le 50) \; ; \; (-50 \le x_2 \le 50) \; ; \; (0 \le x_3 \le 1) \; ; \; (0 \le x_4 \le 1)$$

and if the output file has the filename *"simul.m"*, then the following set of command line instructions can be used to visualize the results that are displayed in figure 7.2:

simul
slca(50,m1ff,m1fa,m2ff,m2af,m1ffc,m1fac,m2ffc,m2afc)

7.1.3 clcam1

The **clcam1** function was designed to visualize the results from the **Optimo** program when we are treating a system consisting of two machine in tandem with batch production on machine 1 and unreliable machines, which was modeled by the denominated model I for this system (see [1] and [9]). The function **clcam1** must be called after loading the output file obtained from the **Optimo** program.
The function has the following structure

$$slca(size,batch\ size,m1,m1c,m1c,m2c)$$

For example, if we have an output file from the **Optimo** with the following state space

$$(-50 \le x_1 \le 50) \; ; \; (-50 \le x_2 \le 50) \; ; \; b_{max} = 3$$

and if the output file has the filename *"simul.m"*, then the following set of command line instructions can be used to visualize the results that are displayed in figure 7.3.

simul
clcam1(50,3,m1,m1c,m2,m2c)

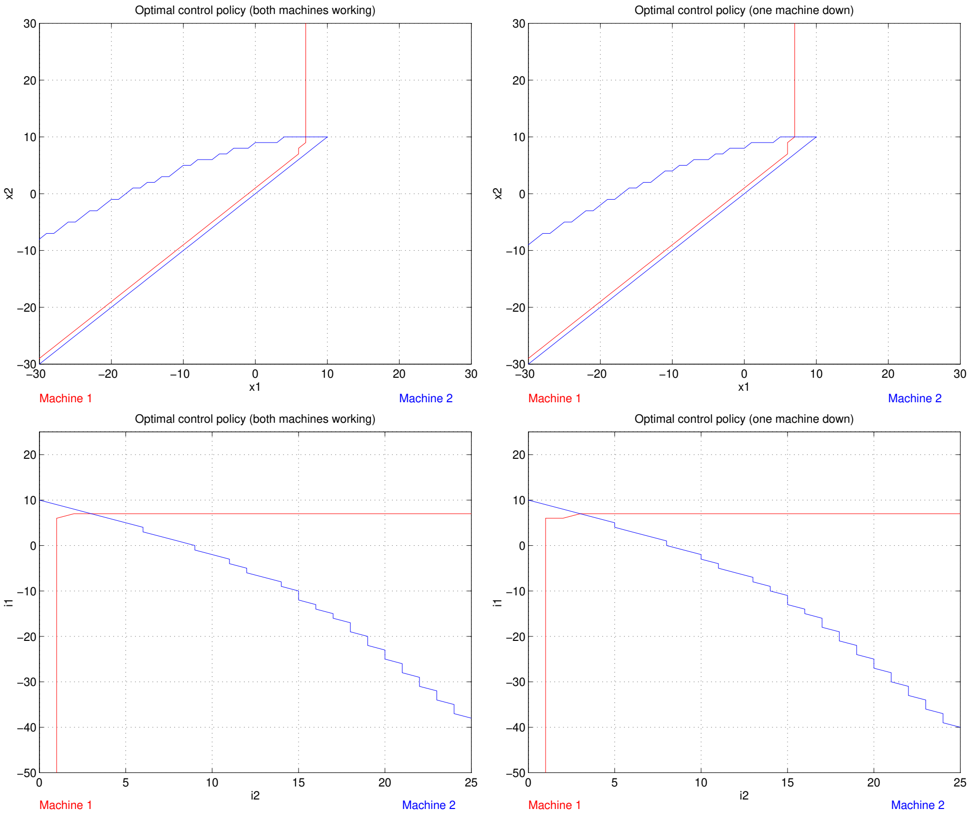

Figure 7.2: **Matlab** output for the file simul.m using slca.

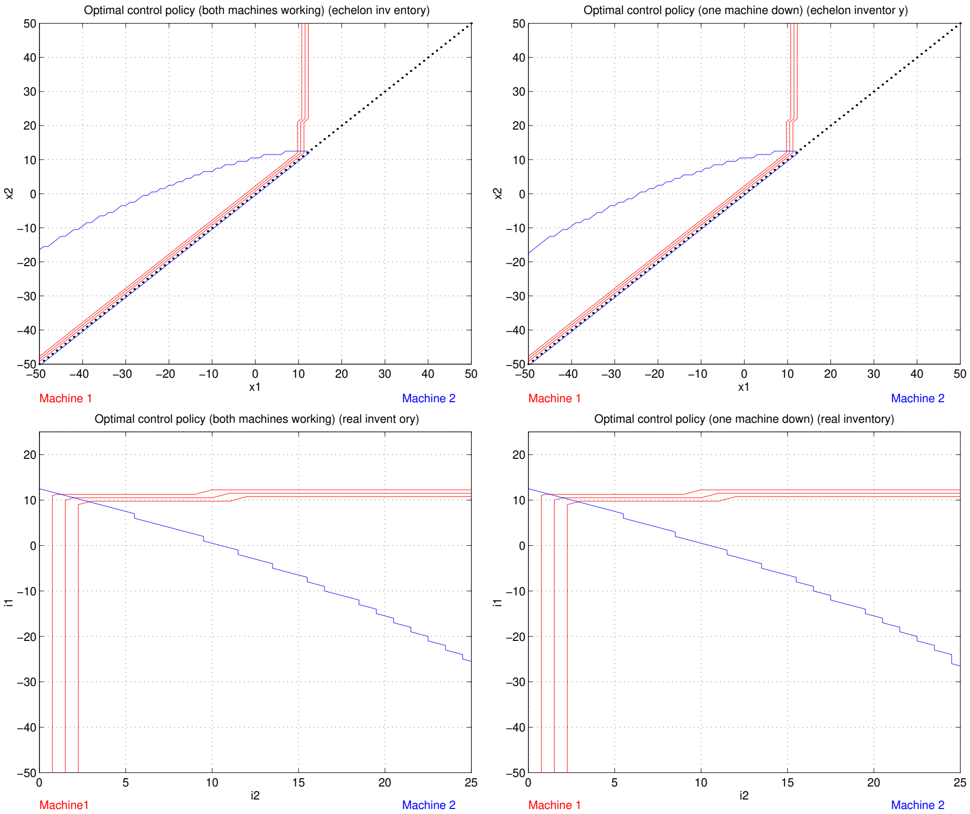

Figure 7.3: **Matlab** output for the file simul.m using clcam1.

7.1.4 clsa(b)

The **clsa(b)** module is a collection of functions designed to visualize the results from the **Optimo** program when we are treating a system consisting of a tandem two-machine system with batch production on machine 1 and reliable machines.
Depending on the maximum number of pieces that can be processed in a batch (b_{max}) in machine 1, the user must chose the appropriate function (ex. for $b_{max} = 3$ the user must call the function clsa3).
The function must be called after loading the output file from the optimo program, and its structure will depend on b_{max}. The user should use the command *help (function)* to see what is the structure of the function (ex. *help clsa4* for $b_{max} = 4$).
For example, if we have an output file from the **Optimo** with the following state space

$$(-50 \leq x_1 \leq 50) \ ; \ (-50 \leq x_2 \leq 50) \ ; \ b_{max} = 3$$

and if the output file has the filename *"simul.m"* then the following set of command line instructions can be used to visualize the results that are displayed in figure 7.4 and 7.5:

simul
clsa3(50,m1,m20,m21,m22,m23,m1c,m2c,m2c1,m2c2,m2c3)

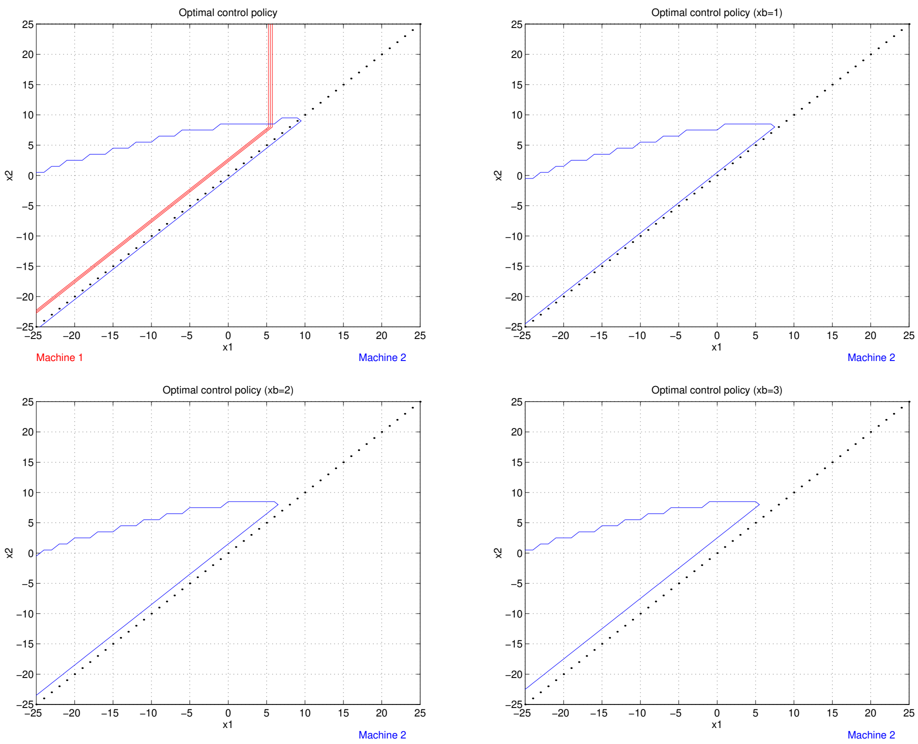

Figure 7.4: **Matlab** output for the file simul.o using clsa3.

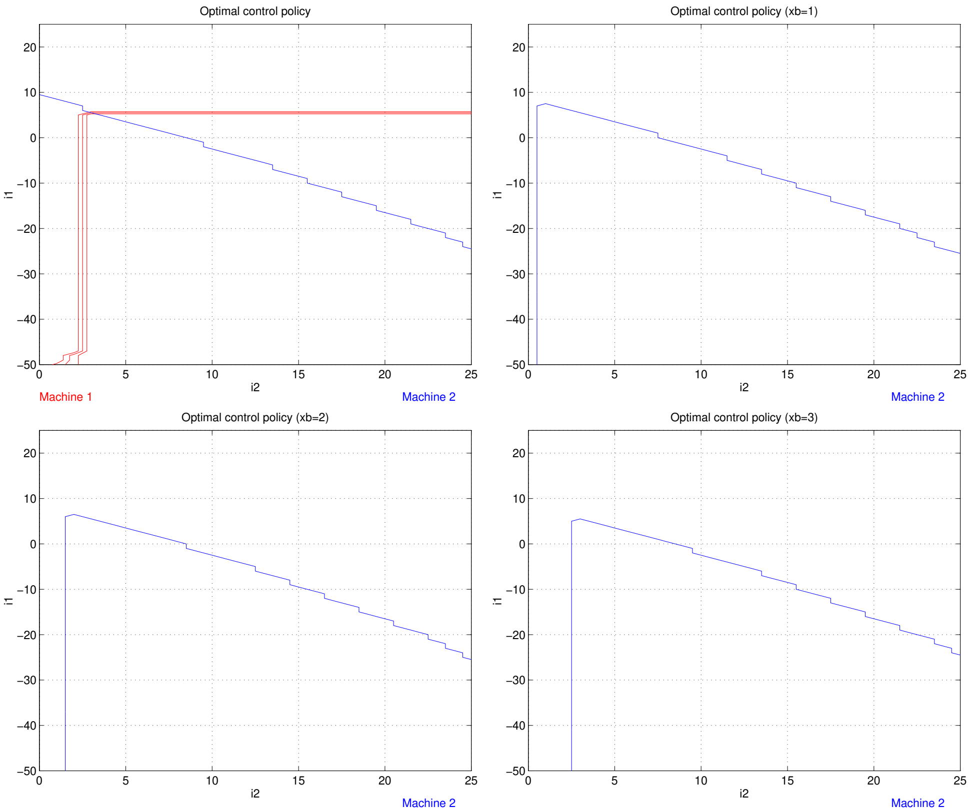

Figure 7.5: **Matlab** output for the file simul.o using clsa3.

7.2 Visualization of results from the Simula program

7.2.1 sim

The function **simslsa** was designed to visualize the results from the **Simula** program when we are treating a system consisting of a two machines in tandem without batch production and with reliable machines.
The function **simslsa** must be called after loading the output file from the **Simula** program. The function has the following structure

sim(size,X,X1,X2,XI1,XI2,I,I1,I2,II1,II2)

For example, if we have an output file from the **Simula** program with the following state space

$$(-100 \leq x_1 \leq 100)\ ;\ (-100 \leq x_2 \leq 100)$$

and if the output file has the filename *"simul.m"*, then the following set of command line instructions can be used to visualize the results that are displayed in figure 7.6 and 7.7.

simul
sim(100,X,X1,X2,XI1,XI2,I,I1,I2,II1,II2)

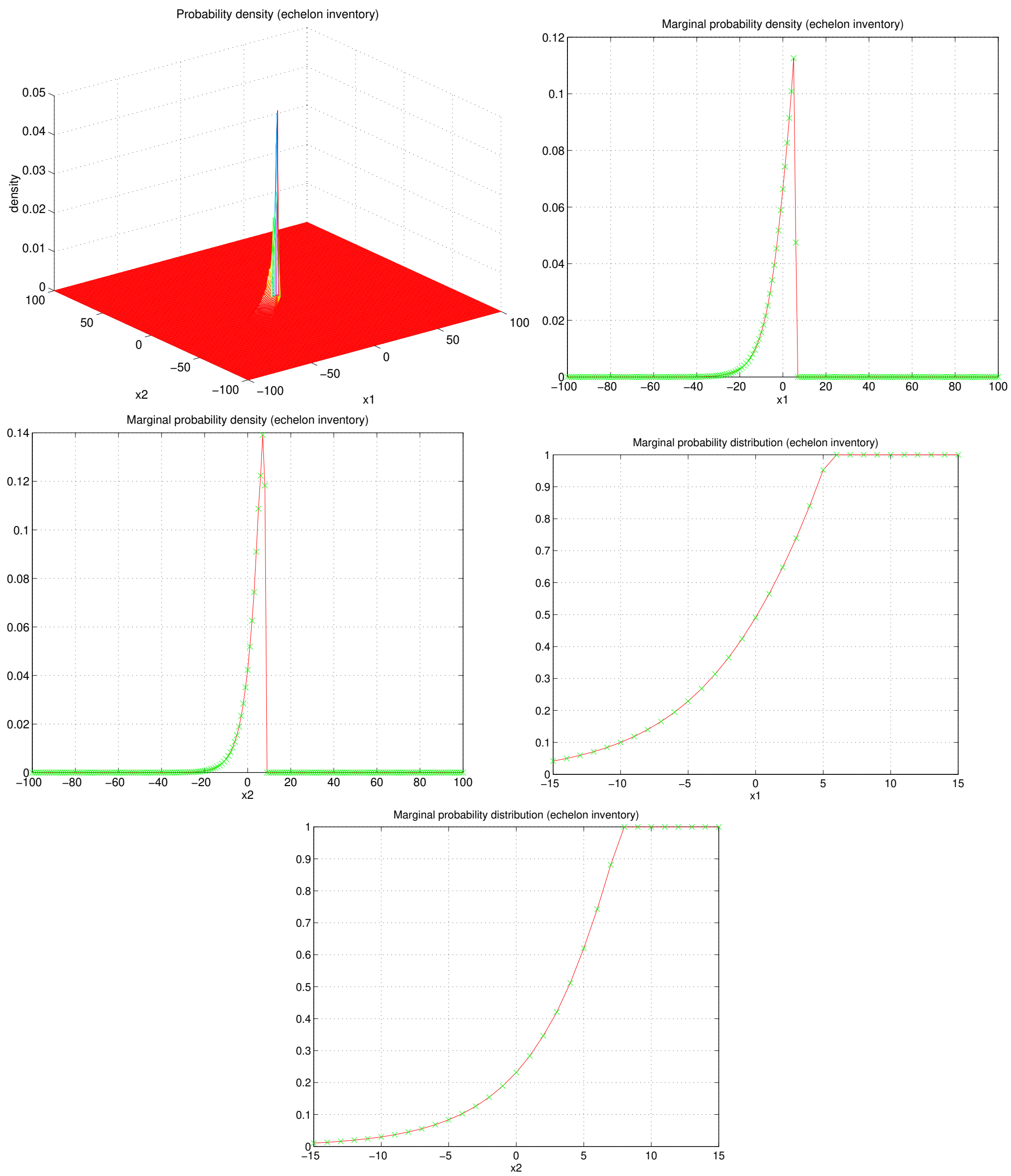

Figure 7.6: **Matlab** output for the file simul.m using clsa3.

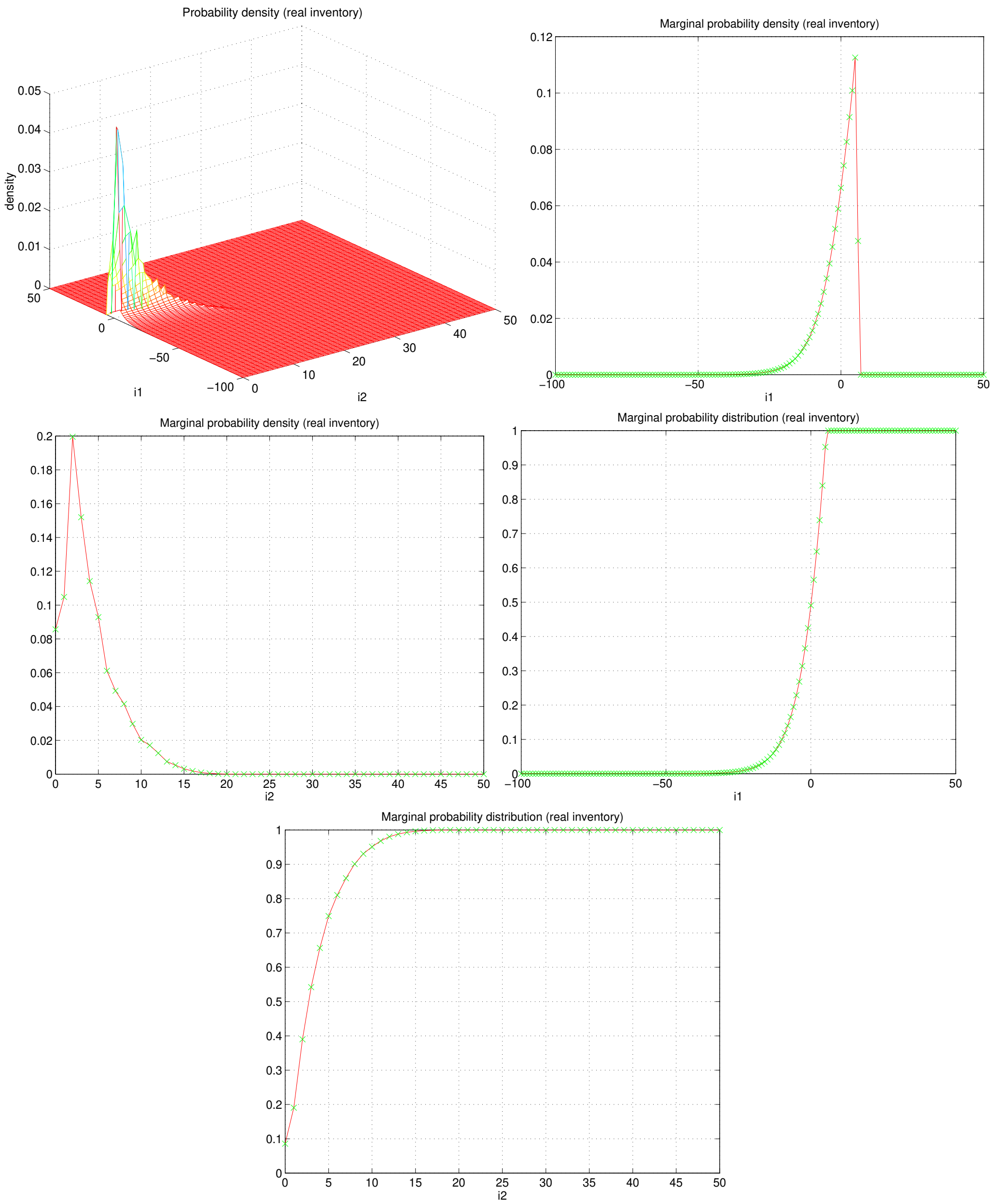

Figure 7.7: **Matlab** output for the file simul.m using clsa3.

Bibliography

[1] J. A. A. Moreira, *Modelação e Controlo de um Sistema de duas Máquinas em Série Incluindo Avarias e Produção de Lotes.* Instituto Superior Técnico, 1998.

[2] B. W. Kernighan and D. M. Ritchie, *The C Programming Language.* Englewood Cliffs,New Jersey: Prentice-Hall, 1988.

[3] B. Stroustrup and M. A. Ellis, *The Annotated C++ Reference Manual.* Reading,Massachusetts: Addison-Wesley, 1990.

[4] J. A. A. Moreira, *Complete Source Code for the MAQDP Software Package.* Instituto Superior Técnico, 1998.

[5] D. P. Bertsekas, *Dynamic Programming: deterministic and stochastic models.* Englewood Cliffs,New Jersey: Prentice-Hall, 1987.

[6] D. P. Bertsekas, *Dynamic Programming and Optimal Control Vol.2.* Belmont, Massachusetts: Athena Publishing, 1995.

[7] J. Banks and J. S. C. II, *Discrete Event Simulation.* Englewood Cliffs, New Jersey: Prentice-Hall, 1984.

[8] B. Hibbard and D. Santek, *Complete Guide for using Vis5d.* Space Science and Engineering Center, University of Wiscounsin.

[9] C. F. G. Bispo, "Modeling a two-station tandem system to acommodate batch and machine failures," *Working paper*, 1998.