**Report about Optimal Control Design for The Reparable System**

**Name: Fangyao Liu**

**UID: 11783980**

**Abstract**

In this optimal control design for the reparable system project, the final goal is to make the probability of a device in good mode approach 90%, the probability of the same device in failed mode approach 10% and the sum of control input’s abstract value should be as small as possible. The problem has first been transformed to a standard quadratic problem. Then analytical solution and Lagrange algorithm have been applied to solve this problem. At last, with the help of MATLAB, these two methods return the same optimized answer.

1. **Problem Description**

A multi-state device with outside control input has two states: good mode and failed mode. The probability in good mode at time t is . The Probability in failed mode at time t is . The outside control input is . The mathematical model for this multi-state decide is introduced in Chung (1981) as below:

Here,

. Constant failure rate of the device for the failure mode 1;

0.65. Constant repair rate when the device is in state 1;

Here set the time interval as . In this time period, the desired situation is:



Therefore, we have a quadratic optimization problem:

1. **Methodology**

Using the backward Euler’s method for discretizing the linear system of ordinary differential equations, we can reduce the controlled system to the following

With the discretized-form constraints, the question can be formed as a standard quadratic problem. In order to reach the standard form of quadratic problem, a replacement of variable is necessary:

And with some algebra transformation, it is easy to get the standard quadratic form of this problem:

Here:

* **Analytical Method**

First, solve the problem using analytical method. The analytical solution to this problem is:

Construct the coefficient matrix in MATLAB, plug all the matrixes into the formula and MATLAB will return the result

* **Numerical Method**

Lagrange algorithm is applied here as the numerical method. The Lagrange function can be derived as:

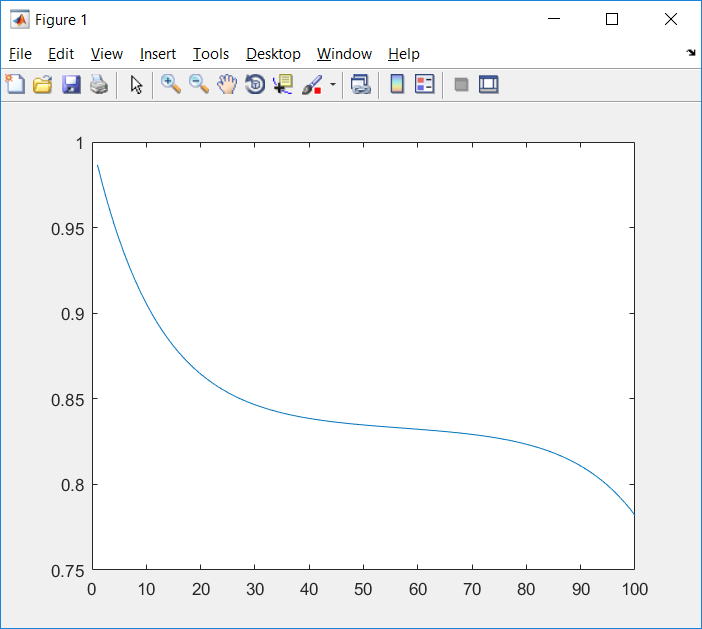
Here, are the Lagrange multipliers. Then, do the derivative of respect to and , and we can get Lagrange conditions:

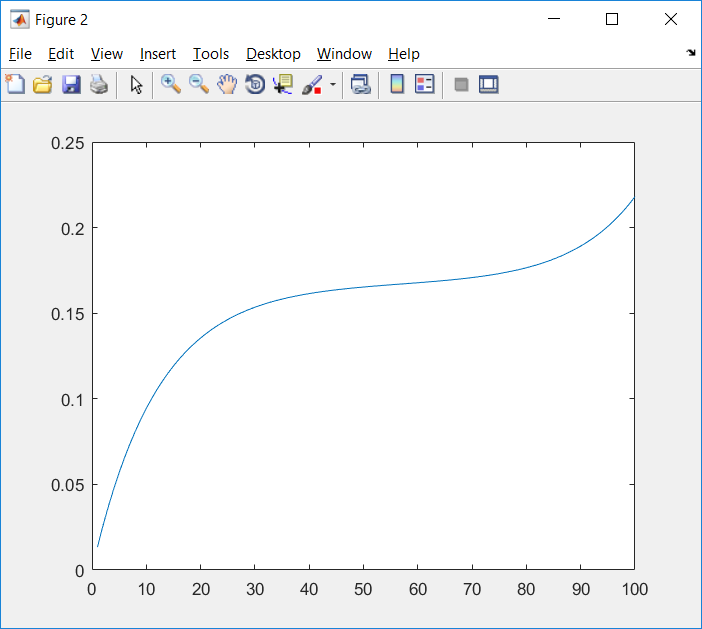
Iteration equations can therefore be derived as:

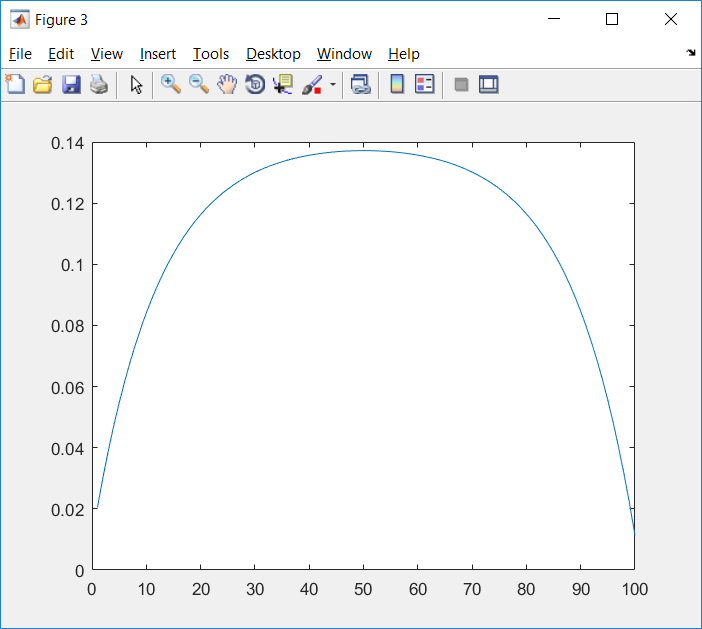
The stopping criteria are:

1. **Results**

* **Analytical Solution**







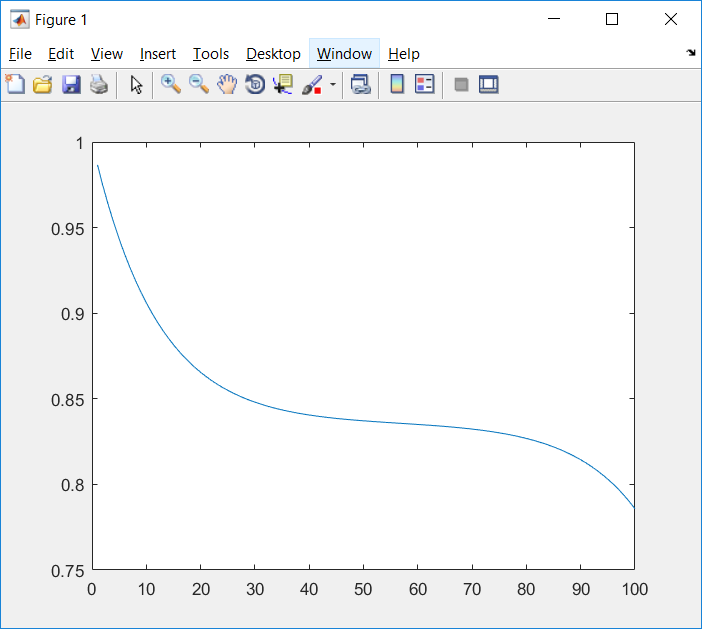
The goal function is:

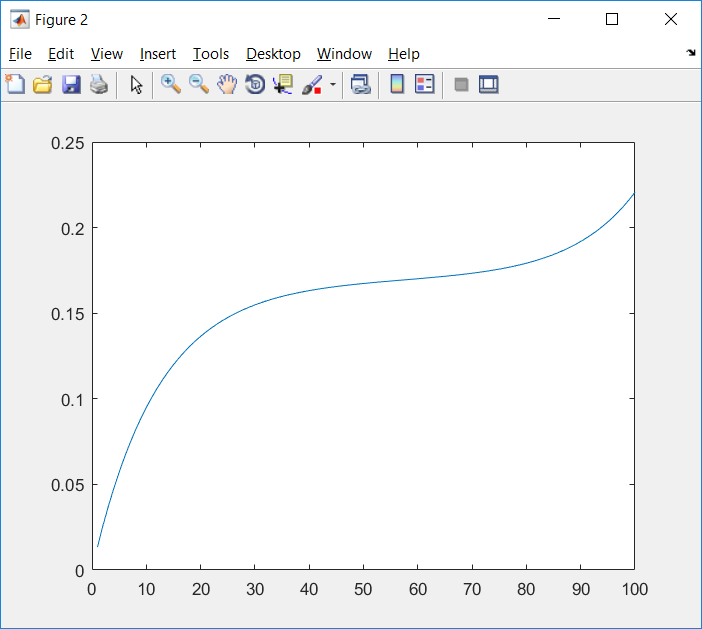
* **The numerical method**

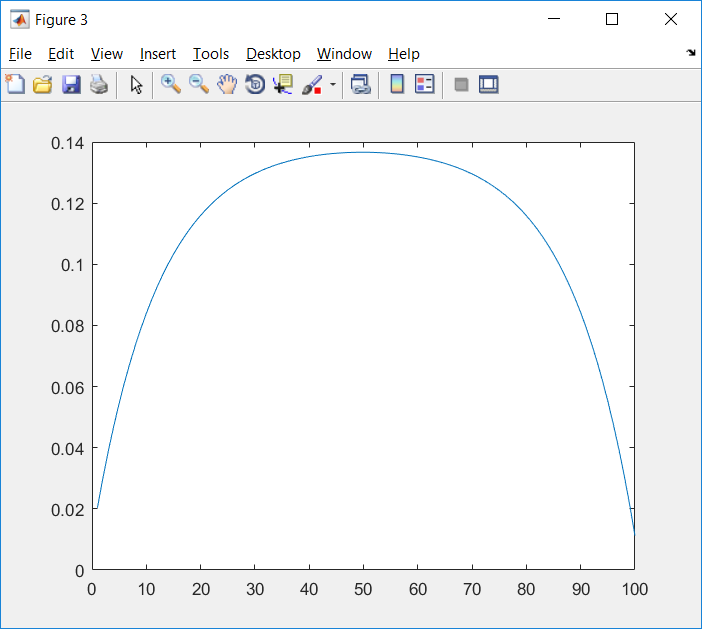
In the program, and .

Initial condition is set as:

MATLAB returns the results through 73489 iterations:







The goal function is:

1. **Observation and Conclusions**

Through analytical and numerical method, same results are achieved. In order to make sure this result is correct, a built-in MATLAB function***fmincon*** is also applied to this project and it returns the exactly the same results, which confirms those are the correct optimal solutions.

However, the results do seem surprising. Our goal is to make approach to 0.9 and approach to 0.1 as close as possible. So the expectation should be like oscillates around 0.9 and oscillates around 0.1. But the results are far from the expectation. Both and have a very short period of stable time (approximately from time 1.5s – 4s) and their stable values are not 90% or 10%. After some reflections, I realized that our goal is not only to make approach to 0.9 and approach to 0.1 as close as possible, but also minimize the abstract sum of control input . Therefore, I believe that the introduction of into the goal function is the reason of why plots are not the same as the expectations.

**Reference**

W.K. Chung. A Reparable Multi-State Device with Arbitrarily Distributed Repair Times Micro.

Reliab., volume 21, number 2, pages. 255-256, 1981.

**Appendix**

* **Analytical Method**

A\_leftup = -20.3\*eye(100)+[zeros(1,100);[20\*eye(99),zeros(99,1)]];

A\_leftdown = 0.3\*eye(100);

A\_midup= 0.65\*eye(100);

A\_middown = -20.65\*eye(100)+[zeros(1,100);[20\*eye(99),zeros(99,1)]];

A\_rightup = eye(100);

A\_rightdown = -1\*eye(100);

A = [[A\_leftup;A\_leftdown],[A\_midup;A\_middown],[A\_rightup;A\_rightdown]];

b = [-1.795;0.205\*ones(99,1);1.795;-0.205\*ones(99,1)];

Q = 2\*eye(300);

Xa = inv(Q)\*A'\*inv(A\*inv(Q)\*A')\*b;

figure(1)

plot(Xa(1:100)+0.9\*ones(100,1));

figure(2)

plot(Xa(101:200)+0.1\*ones(100,1));

figure(3)

plot(Xa(201:300));

1/2\*Xa'\*Q\*Xa

* **Numerical Method**

A\_leftup = -20.3\*eye(100)+[zeros(1,100);[20\*eye(99),zeros(99,1)]];

A\_leftdown = 0.3\*eye(100);

A\_midup= 0.65\*eye(100);

A\_middown = -20.65\*eye(100)+[zeros(1,100);[20\*eye(99),zeros(99,1)]];

A\_rightup = eye(100);

A\_rightdown = -1\*eye(100);

A = [[A\_leftup;A\_leftdown],[A\_midup;A\_middown],[A\_rightup;A\_rightdown]];

b = [-1.795;0.205\*ones(99,1);1.795;-0.205\*ones(99,1)];

Q = 2\*eye(300);

X1 = 0.5\*ones(300,1);

lamda1 = 0.5\*ones(200,1);

X0 = zeros(300,1);

lamda0 = zeros(200,1);

n=0;

while(norm(lamda1-lamda0)>0.00001||norm(X1-X0)>0.00001)

X0 = X1;

lamda0 = lamda1;

X1 = X0 - 0.001\*(Q\*X0+A'\*lamda0);

lamda1 = lamda0 + 0.001\*(A\*X0-b);

n = n+1;

end

figure(1)

plot(X1(1:100)+0.9\*ones(100,1));

figure(2)

plot(X1(101:200)+0.1\*ones(100,1));

figure(3)

plot(X1(201:300));

1/2\*X1'\*Q\*X1