Study of Real time Dynamic Preventive Maintenance Policy for Deteriorating Production Systems

Chin-Tai Chen¹, Ming-Han Lin², Ching-Ping Wang³ and John Yuan⁴

¹ Associate professor, Department of Industrial Engineering and Management, Ta-Hwa Institute of Technology, Hsinchu, 307, Taiwan, ROC.
² Associate professor, Department of Automation Engineering, Ta-Hwa Institute of Technology, Hsinchu 307, Taiwan, ROC.
³ Department of Industrial Engineering and Engineering Management National Tsing-Hua University, R.O.C.
⁴ Professor, Department of Industrial Engineering and Engineering Management National Tsing-Hua University, R.O.C.

Abstract

The purpose of this article is to propose a both state and time dependent preventive maintenance policy for a deteriorating system. Using health index integrating the parameters inspected by sensors can generate the real-time equipment operating state. In this research, a discrete time Markovian multi-state deteriorating model is adopted to describe the state transition of equipment. The model also includes (1) the deteriorating of operating states and (2) the aging of transition probabilities; (3) Base on historical data, we can estimate the model parameters ( aging factor and initial transition matrix ), and (4) assume that multiple imperfect actions with risk are available. For the practical application of the model, the maintenance time is also taken into consideration. The real time optimal maintenance policy of each operating state can be generated by the rule of the minimum expected total cost per unit time for each cycle during finite time interval. Algorithm and computer code are also listed for industrial application.

Keywords: dynamic preventive maintenance policy, aging factor, health index

1. Introduction

The advanced technology industries require high precision and highly stable equipment to avoid the cost caused by failure occurrence. Equipment automation and preventive maintenance must be executed to keep equipment within proper operating states. It has always been an important topic for study in inspecting the operating states of equipment by sensors and determining the dynamic preventive maintenance policies according to the real-time condition by computer analyzing centers.

Extensive reviews of various maintenance policies on a deteriorating system can be found in [1, 2, 3]. The first
class is to deal with a two state system. This follows from the fact that the action is taken upon failure only under such no inspection equipment case. The age replacement policy (i.e. replacement upon failure or at age) [4] and their extensions by including minimal repair such as [5-10] belong to such a class. The block replacement policy (i.e. replacement at \(kT\) for \(k=1,2,...\) or upon failure) and the failure replacement policy (i.e. replacement upon failure) also belong to such a class [11-13]. The scheduled maintenance policies through predicting failure time statistically also belong to this class [cf.14, 15].

The second class is to deal with a multi-state deteriorating system provided with inspection equipment(s). Each state of the system must be identified through inspection for example at each \(n\). Then the optimal maintenance action can be decided at \(n\) for the system by minimizing the objective function such as the expected total cost per life cycle, the expected total cost per unit time and etc. Hence, it is only the second class on preventive maintenance policies. Lam and Yeh [16] proposed a control-limit replacement policy under continuous inspection so that replacement is taken optimally whenever the threshold state \(j^*\) is identified by inspection or the complete failure \(L\) is observed. Lam and Yeh [12] proposed another control-limit replacement policy under inspection at each \(n\) so that the replacement takes places at \(n\) whenever the system state \(x\) at \(n\) satisfies \(j^* \leq x \leq L\) for a threshold state \(j\), to determine the optimal \((d^*,j^*)\). Chiang and Yuan [17,18] proposed other control-limit preventive maintenance policies under continuous and periodic inspection respectively by determining two threshold states \(i^* < j^*\) (\(1 < i^* < j^* < L\)) so that the optimal action is that repair (resp. replacement, do-nothing) is taken whenever the system state is identified as \(x\) so that \(i^* \leq x < j^*\) (resp. \(j^* \leq x \leq L\): otherwise). Wood [19] proposed a control limit rule that requires restoration of the system whenever the damage exceeds a certain level under continuous inspection. Such methods have to assume that the system before taking any maintenance policy satisfies a continuous-time Markov chain. Also, the threshold state(s) thus obtained and so the optimal maintenance action taken is state dependent only (i.e. not time dependent). Jardine etc. [20] proposed an optimal dynamic (i.e. time-dependent too) replacement policy for condition-based maintenance. Wildeman etc. [22] proposed another dynamic preventive maintenance policy that takes a long-time tentative plan as a basis for a subsequent adaptation according to available information on the short term by a rolling-horizon approach.

Recently, Chen et al [24] proposed a state dependent preventive maintenance policy for a multi-state deteriorating system, which is equipped with inspection equipment(s) connected to a computer center. After the system being identified as state \(x\) at \(nd\) through computation by the computer center after inspection (or measurement) via equipment(s), one maintenance action with the minimum expected total cost since \(nd\) till \(Nd\) (where \(N = n+K\) for a fixed integer \(0 < K < \infty\)) will be chosen from the set \(A_x\) of alternatives also with the help of the computer center. In real case, the expected total costs since \(nd\) till \(Nd\) will be time dependent and so is the maintenance action chosen at \(nd\).

Extended the research of Chen et al. [24], the purpose of this paper is to propose both state and time dependent preventive maintenance policy for a deteriorating system. Such a dynamic preventive maintenance policy can be illustrated in a flow chart in Figure 1. In Chiang and Yuan [17,18], the percentage of defectives among products, which the system produces, is in
fact taken as the system healthy index \( H \), their control-limit preventive maintenance policies cannot prevent the production system from products defectives effectively as the proposed one here.

Figure 1  The diagram of the dynamic preventive maintenance policy

2. System Modeling and Algorithm

2-1 Symbol description :

- \( \beta \) discount rate
- \( K_{PM} \) period for PM policy
- \( c(a_{ik}) \) expected cost for action \( a_{ik} \) and \( c(a_{0i}) = 0 \) \( \forall i \in S_H \)
- \( o(i) \) expected cost during \( d \) due to minimal repairs or scrap given system state \( i \)
- \( R_L \) replacement cost of equipment
- \( T(a_{ik}) \) time for action \( a_{ik} \) ; \( T(a_{0i}) = d \) \( \forall i \in S_H \)
- \( T_L \) replacement time
- \( t_n \) detection time point for \( n \)th decision
- \( g_n \) equipment age for a system at \( t_n \)
- \( \phi^{(k)}_{t_n}(a_{ik}) \) expect cost per unit time for action \( a_{ik} \in W_i \) at \( t_n \) and state
- \( \phi^{(k-m)}_{t_n-k-m}(a_{ik}) \) expect cost per unit time for

action \( a_{ik} \in W_i \) at \( t_{n+m} \) and state

\( a_i^* \) minimum action of \( \phi^{(k)}_{t_n}(a_{ik}) \) ; i.e., \( a_i^* \) is satisfied by \( \phi^{(k)}_{t_n}(a_{ik}) \) for \( a_{ik} \in W_i \)

\( W^{(k)}_{t_n} \) optimal action under minimum cost per unit time at \( t_n \)

\( ( W^{(k)}_{t_n} = (w^{(k)}_{n}(i)|i < i \leq L) \)

\( W^{(m)}_{t_n-k-m} \) optimal action under minimum cost per unit time at \( t_{n+k-m} \)

\( ( W^{(m)}_{t_n-k-m} = (w^{*}_{n+k-m}(i)|i < i \leq L) \)

2-2 Preventive maintenance policy

Expected total cost due to minimum repairs or scrap given system state \( i \) during \((t_n, t_{n+1})\) for action \( a_{ik} \) can be expressed as follows:

International Journal of The Computer, the Internet and Management Vol. 15\#1 (January – April, 2007) pp 1 - 14
Probability at detected system state \( j \) for action \( a_{ik} \) \((i = 1, 2, \cdots, L - 1)\) can be expressed as follows:

\[
P_{i,j}^{s_k}(a_{ik}) = \sum_{h=1}^{L} r_{kh} \cdot P_{h,j}^{s_k} \quad \forall t \geq 0
\]

\[
P_{L,j}^{s_k}(a_{i0}) = P_{1,j}^{s_k} = P_{1,j}^{0} \quad \forall t \geq 0 \quad \text{(when \( i = L \))}
\]

**2.3 Algorithm:**

**Step 0:** determine optimal action

\[
W_{t_{n+K}}^{(0)}(w_{n+1}^{*}(i))_{1 \leq i \leq L} \quad \text{at} \ t_{n+K}, \ i.e.,
\]

\[a_{i}^{*} = \arg \min \{ \phi^{(0)}_{t_{n+K}}(a_{ik}) \} \quad \forall i \in S_{H}
\]

Firstly, calculate the cost and time for \( a_{ik} \) at \( t_{n+K} \):

\[
\tau_{t_{n+K}}^{(0)}(a_{ik}) = \begin{cases} R_{L} & \text{if } i = L \\ o(i) & \text{if } k = 0 \\ c(a_{ik}) + O(a_{ik}) & \text{if } k \neq 0, L \end{cases}
\]

and

\[
\psi_{t_{n+K}}^{(0)}(a_{ik}) = \begin{cases} T_{L} & \text{if } i = L \\ d & \text{if } k = 0 \\ T(a_{ik}) & \text{if } k \neq 0, L \end{cases}
\]

then

\[
\phi_{t_{n+K}}^{(0)}(a_{ik}) = \frac{\tau_{t_{n+K}}^{(0)}(a_{ik})}{\psi_{t_{n+K}}^{(0)}(a_{ik})}
\]

we obtain that,

\[
a_{i}^{*} = \arg \min \{ \phi^{(0)}_{t_{n+K}}(a_{ik}) \} \quad \forall i \in S_{H};
\]

\[
\tau_{t_{n+K}}^{(0)}(a_{i}^{*}), \quad \psi_{t_{n+K}}^{(0)}(a_{i}^{*}), \quad \text{and}
\]

\[
(W_{t_{n+K}}^{(0)}(w_{n+1}^{*}(i)))_{1 \leq i \leq L} = (a_{i}^{*})_{1 \leq i \leq L}
\]

**Step 1** Given the condition

\[
W_{t_{n+K-1}}^{(0)}(w_{n+K-1}^{*}(i))_{1 \leq i \leq L}
\]

Firstly, calculate the cost and expanded time for \( a_{ik} \) at \( t_{n+K-1} \):

\[
\tau_{t_{n+K-1}}^{(1)}(a_{ik}, W_{t_{n+K}}^{(0)}) \text{ and } \psi_{t_{n+K-1}}^{(1)}(a_{ik}, W_{t_{n+K}}^{(0)})
\]

For convenience, let random variable

\[
\tau_{t_{n+K-1}}^{(1)}(a_{ik}, W_{t_{n+K}}^{(0)}) \text{ be:}
\]

when \( X_{t_{n+K}} = j \),
then

\[
\tau_{i_{n+K-1}}^{(1)}(a_{ik}, W_{i_{n+K-1}}^{(0)}) = \tau_{i_{n+K-1}}^{(1)}(a_{ik}, W_{n+K}^{*}(j))
\]

and

\[
\psi_{i_{n+K-1}}^{(1)}(a_{ik}, W_{i_{n+K-1}}^{(0)}) = \psi_{i_{n+K-1}}^{(1)}(a_{ik}, W_{n+K}^{*}(j))
\]

then

\[
\phi_{i_{n+K-1}}^{(1)}(a_{ik}, W_{i_{n+K-1}}^{(0)}) = \sum_{j=1}^{L} \phi_{i_{n+K-1}}^{(1)}(a_{ik}, W_{i_{n+K-1}}^{(0)}) \cdot P_{j_{n+K-1}}(W_{i_{n+K-1}}^{(0)})
\]

and

\[
\phi_{i_{n+K-1}}^{(1)}(a_{ik}, W_{i_{n+K-1}}^{(0)}) = \min_{a_{ik} \in W_{i_{n+K-1}}} \{ \phi_{i_{n+K-1}}^{(1)}(a_{ik}, W_{i_{n+K-1}}^{(0)}) \} \quad \forall i \in S_H ;
\]

\[
\phi_{i_{n+K-1}}^{(1)}(a_{ik}, W_{i_{n+K-1}}^{(0)}) = \sum_{j=1}^{L} \phi_{i_{n+K-1}}^{(1)}(a_{ik}, W_{i_{n+K-1}}^{(0)}) \cdot P_{j_{n+K-1}}(a_{ik}^{*})
\]

(Given \( W_{i_{n+K-1}}^{(0)} = (w_{n+K}^{*}(i))_{i \in S_H} \),

then \( W_{i_{n+K-1}}^{(0)} = (w_{n+K}^{*}(i))_{i \in S_H} = (a_{ik}^{*})_{i \in S_H} \) is obtained)

**Step 2** Given the condition of \( W_{i_{n+K-1}}^{(1)} = (w_{n+K}^{*}(i))_{i \in S_H} \), determine the optimal maintenance action

\[
W_{i_{n+K-2}}^{(2)} = (w_{n+K-2}^{*}(i))_{i \in S_H} \text{ at } t_{n+K-2}.
\]

For convenience, let

\[
\tau_{i_{n+K-2}}^{(1)}(w_{i_{n+K-2}}^{*}(i)) = \tau_{i_{n+K-2}}^{(1)}(w_{n+K-2}^{*}(i), W_{i_{n+K-2}}^{(0)}) \cdot P_{j_{n+K-2}}(w_{n+K-2}^{*}(i))
\]

\[
\psi_{i_{n+K-2}}^{(1)}(w_{i_{n+K-2}}^{*}(i)) = \sum_{j=1}^{L} \psi_{i_{n+K-2}}^{(1)}(w_{n+K-2}^{*}(i), W_{i_{n+K-2}}^{(0)}) \cdot P_{j_{n+K-2}}(w_{n+K-2}^{*}(i))
\]

It means that given the condition \( W_{i_{n+K-1}}^{(0)} \), the expected cost and time due to minimum repair or scrap given system state \( i \) during \( t_{n+K-1} \) are calculated, respectively.

Secondly, the expected cost and time due to minimum repair or scrap given system state \( j \) during \( t_{n+K-2} \) are calculated,

\[
\tau_{i_{n+K-2}}^{(2)}(a_{ik}, W_{i_{n+K-2}}^{(0)}) = \tau_{i_{n+K-2}}^{(2)}(a_{ik}, W_{n+K-2}^{*}(j))
\]

\[
\psi_{i_{n+K-2}}^{(2)}(a_{ik}, W_{i_{n+K-2}}^{(0)}) = \psi_{i_{n+K-2}}^{(2)}(a_{ik}, W_{n+K-2}^{*}(j))
\]

Let random variable \( \tau_{i_{n+K-2}}^{(2)}(a_{ik}, W_{i_{n+K-2}}^{(0)}) \) be:

when \( X_{i_{n+K-1}} = j \),

\[
\tau_{i_{n+K-2}}^{(2)}(a_{ik}, W_{i_{n+K-2}}^{(0)}) = \tau_{i_{n+K-2}}^{(2)}(a_{ik}, W_{n+K-2}^{*}(j))
\]

and

\[
\psi_{i_{n+K-2}}^{(2)}(a_{ik}, W_{i_{n+K-2}}^{(0)}) = \psi_{i_{n+K-2}}^{(2)}(a_{ik}, W_{n+K-2}^{*}(j))
\]

then,

\[
\phi_{i_{n+K-2}}^{(2)}(a_{ik}, W_{i_{n+K-2}}^{(0)}) = \phi_{i_{n+K-2}}^{(2)}(a_{ik}, W_{n+K-2}^{*}(j))
\]
and
\[
\phi^{(2)}_{t_{n+K-2}}(a^*_i, W^{(1)}_{t_{n+K-1}}) = \min_{a_{ik} \in R^1_i} \left\{ \phi^{(2)}_{t_{n+K-2}}(a_{ik}, W^{(1)}_{t_{n+K-1}}) \right\}
\]

We obtain that,
\[
a^*_i = \arg \min_{a_{ik} \in R^1_i} \left\{ \phi^{(2)}_{t_{n+K-2}}(a_{ik}, W^{(1)}_{t_{n+K-1}}) \right\} \quad \forall i \in S_H ;
\]
\[
\phi^{(2)}_{t_{n+K-2}}(a^*_i, W^{(1)}_{t_{n+K-1}}) = \sum_{j=1}^{L} \phi^{(j)}_{t_{n+K-1}}(w^*_i, W^{(0)}_{t_{n+K-1}}) \cdot p_{t_{n+K-1}}(a^*_i)
\]

**Step m (m=3, \ldots, K)** Given the condition \(W^{(m-1)}_{t_{n+K-m+1}}\), determine the optimal maintenance action
\[
W^{(m)}_{t_{n+K-m}} = (w^*_i, (i)) \quad \text{at} \quad t_{n+K-m}.
\]

Let
\[
\tau^{(m)}_{t_{n+K-m}}(a^*_i, w^*_i) = \sum_{j=1}^{j} \tau^{(j)}_{t_{n+K-m}}(w^*_i, w^{(j)}_{t_{n+K-m}}) \cdot p_{t_{n+K-1}}(w^*_i, W^{(0)}_{t_{n+K-1}})
\]
\[
\psi^{(m)}_{t_{n+K-m}}(w^*_i) = \sum_{j=1}^{j} \psi^{(j)}_{t_{n+K-m}}(w^*_i, w^{(j)}_{t_{n+K-m}}) \cdot p_{t_{n+K-1}}(w^*_i, W^{(0)}_{t_{n+K-1}})
\]
\[
\phi^{(m)}_{t_{n+K-m}}(a^*_i, W^{(m-1)}_{t_{n+K-m+1}}) = \min_{a_{ik} \in R^1_i} \left\{ \phi^{(m)}_{t_{n+K-m}}(a_{ik}, W^{(m-1)}_{t_{n+K-m+1}}) \right\}
\]

We obtain that,
\[
\phi^{(m)}_{t_{n+K-m}}(a^*_i, W^{(m-1)}_{t_{n+K-m+1}}) = \sum_{j=1}^{j} \phi^{(j)}_{t_{n+K-m}}(a^*_i, W^{(m-1)}_{t_{n+K-m+1}}) \cdot p_{t_{n+K-1}}(a^*_i, W^{(m-1)}_{t_{n+K-m+1}})
\]

For convenience, let random variable
\[
\tau^{(m)}_{t_{n+K-m}}(a^*_i, W^{(m-1)}_{t_{n+K-m+1}}) \quad \text{and} \quad \psi^{(m)}_{t_{n+K-m}}(a^*_i, W^{(m-1)}_{t_{n+K-m+1}})
\]
be,
\[
\tau^{(m)}_{t_{n+K-m}}(a^*_i, W^{(m-1)}_{t_{n+K-m+1}}) = \tau^{(m)}_{t_{n+K-m}}(a^*_i, W^{(m-1)}_{t_{n+K-m+1}})
\]
and
\[
\psi^{(m)}_{t_{n+K-m}}(a^*_i, W^{(m-1)}_{t_{n+K-m+1}}) = \psi^{(m)}_{t_{n+K-m}}(a^*_i, W^{(m-1)}_{t_{n+K-m+1}})
\]

Then,
\[
\phi^{(m)}_{t_{n+K-m}}(a^*_i, W^{(m-1)}_{t_{n+K-m+1}}) = \phi^{(m)}_{t_{n+K-m}}(a^*_i, W^{(m-1)}_{t_{n+K-m+1}})
\]
and
\[
\phi^{(m)}_{t_{n+K-m}}(a^*_i, W^{(m-1)}_{t_{n+K-m+1}}) = \min_{a_{ik} \in R^1_i} \left\{ \phi^{(m)}_{t_{n+K-m}}(a_{ik}, W^{(m-1)}_{t_{n+K-m+1}}) \right\}
\]

We obtain that,
\[
\phi^{(m)}_{t_{n+K-m}}(a^*_i, W^{(m-1)}_{t_{n+K-m+1}}) = \sum_{j=1}^{j} \phi^{(j)}_{t_{n+K-m}}(a^*_i, W^{(m-1)}_{t_{n+K-m+1}}) \cdot p_{t_{n+K-1}}(a^*_i, W^{(m-1)}_{t_{n+K-m+1}})
\]

For convenience, let
Study of Real time Dynamic Preventive Maintenance Policy for Deteriorating Production Systems

It means that given the condition \( W_{t_{n+K-m}}^{(m)} \), the expected cost and time due to minimum repair or scrap given system state \( i \) during \( t_{n+K-m} \) are calculated, respectively.

**Step \( K+1 \):**

We obtain

\[
W_{t_n}^{(K)} = \left( w^*_n(i) \right)_{1 \leq i \leq L}, \quad \forall i \in S_H
\]

Then the optimal maintenance period under the condition of the expected cost and time due to minimum repair or scrap given system state \( i \) (i.e.,

\[
\phi^{(K)}_{t_n} [W^{(K)}] = \min_{K} \left\{ \phi^{(K)}_{t_n} [W^{(K)}(i)] \right\}
\]

for optimal action \( A^*_n(t_{n}) = W_{t_n}^{(K^*_n)(i)} \) during

\[
(t_{n}, t_{n+K^*_n}(t_{n})) \text{ is obtained,}
\]

\[
K^*_n(t_{n}) = \arg \min_{K} \left\{ \phi^{(K)}_{t_n} [W^{(K)}(i)] \right\}, \quad K = 1, 2, 3, \ldots, K_{PM}
\]

The program code using Matlab software is listed in Appendix for reference.

3. Concluding Remark

Extended the research of Chen et al. [24], the purpose of this paper is to propose both state and time dependent preventive maintenance policy for a deteriorating system. This research is focused on the deteriorating system. Using health index integrating the parameters inspected by sensors can generate the real-time equipment operating state. In this research, a discrete time Markovian multi-state deteriorating model is adopted to describe the state transition of equipment.

The estimation methodologies of the model parameters are also provided in this paper. For the practical application of the model, the maintenance time is also taken into consideration. According to the real-time equipment status, the real time optimal maintenance policy of each operating state can be generated by the rule of the minimum expected total cost per unit time for each cycle during finite time interval.

References


Appendix

Program code using Matlab software

Estimate aging factor :

```
【Command Window】
clear all;
scandata;
MLE;
【scandata.m】
% given initial transition probability matrix
P=[0.9501 0.0192 0.0137 0.0127 0.0032 0.0007 0.0003 0.0001;  
   0.9318 0.0645 0.0015 0.0010 0.0006 0.0005 0.0001;  
   0 0.9169 0.0792 0.0017 0.0010 0.0009 0.0003;  
   0 0 0.9174 0.0757 0.0040 0.0015 0.0014;  
   0 0 0 0.9599 0.0357 0.0029 0.0015;  
   0 0 0 0 0.8600 0.1379 0.0021;  
   0 0 0 0 0 0.7081 0.2919;  
   0 0 0 0 0 0 1.0000];
P1=P;
% given maintenance action cost
C=[ 0 0 0 0 0 0 0 50 80 130 290 400 40 70 100 150 220 35 75 120 200 30 100 150 65 100 45 15000];
% given action risk matrix
R=[0.9817 0.0049 0.0044 0.0035 0.0030 0.0025 0 0;  
   0.8955 0.0878 0.0161 0.0006 0 0 0;  
   0 0.8672 0.0745 0.0517 0.0066 0 0;  
   0 0 0.9214 0.0554 0.0131 0.0101 0;  
   0 0 0 0.8188 0.1763 0.0027 0.0022;  
   0 0 0 0 0.7747 0.2211 0.0042];
% given action time
T=[1 1 1 1 1 1 5 10 20 40 65 85 6 12 22 45 70 8 15 35 60 10 30 55 25 40 30 125];
% given regular state cost
o=[100 100 125 200 300 425 450 15000];
global O
O(1:7)=o(1:7);
O(8:13)=R(1,:)*o;
O(14:18)=R(2,:)*o;
O(19:22)=R(3,:)*o;
O(23:25)=R(4,:)*o;
O(26:27)=R(5,:)*o;
O(28)=R(6,:)*o;
% discount rate
global beta
beta=0.96;
global cost_star
```

---

*International Journal of The Computer, the Internet and Management* Vol. 15#1 (January – April, 2007) pp 1 - 14

9
global period_star
global UC_star
% given aging factor
global aging_parameter
aging_parameter=0.025;

global Kpm
Kpm=10;

【sample.m】
scandata;
P1(:,:,1)=P;
for k=2:90
    P1(:,:,k)=aging(P1(:,:,k-1),aging_parameter);
end
% generate 50 sample paths
for num=1:50
    state=1;
    sample_path(num,state)=1;
    for k=2:90
        sample_path(num,k)=trans(rand,P1(sample_path(num,k-1),:,:k-1));
        if sample_path(num,k)==8
            for t=k:90
                sample_path(num,t)=8;
            end
            break
        end
    end
end
% calculate lengths of 50 sample paths
length = zeros(1,50);
for num = 1 : 50
    for k = 1 : 90
        length(num) = length(num) + 1;
        if sample_path(num,k) == 8
            break
        end
    end
end

【MLE.m】
sample;
for a = 1 : 9999
    sigma = a / 10000;
    P2(:,:,1,a)=P;
    for k = 2 : max(length)
        P2(:,:,k,a) = aging(P2(:,:,k-1,a),sigma);
    end
end
L=zeros(1,9999);
for a = 1 : 9999
    for num = 1 : 50
        for k = 1 : length(num)-1
            L(a) = L(a) + log(P2(sample_path(num,k),sample_path(num,k+1),k,a));
        end
    end
end

[sorted,index]=sort(L);
Est=index(9999)/10000

Example for PM policy :

【Command Window】
clear all;
scandata;
% given t=20d
for i = 1 : 20
    P=aging(P,aging_parameter);
end
dynamic;

【aging.m】
function matrix=aging (P,sigma);
    matrix=P;
    for i = 1 : 7
        m = find( P(i,:) == max( P(i,:)));
        if m == 8
            m = 7;
        end
        Lsum = 0 ;Rsum = 0;
        for j = 1 : m
            Lsum = Lsum+P(i,j);
        end
        for j = m+1 : 8
            Rsum = Rsum+P(i,j);
        end
        for j = 1 : m
            if Lsum>0
                matrix(i,j) = P(i,j)*(1 - (Rsum/Lsum)*sigma);
            end
            if matrix(i,j) < 0
                matrix(i,j) = 0;
            break
        end
    end
end
for j = m+1 : 8
    matrix(i,j) = P(i,j)*(1 + sigma);
    if matrix(i,j) > 1
        matrix(i,j) = 1;
        break
    end
end
end

【dynamic.m】
M(:,:,1)=P;
for i = 1 : Kpm+1
    Q(:,:,i)=R*M(:,:,i);
    M(:,:,i+1)=aging(M(:,:,i),aging_parameter);
end
for K = 0 : Kpm
    cost_star=zeros(1,8);
    period_star=zeros(1,8);
    UC_star=zeros(1,8);
    label=[1 0 0 0 0 0 0 29];
    m=K+1;
    while m>=1
        dynamic_iteration;
        m=m-1;
    end
    A(K+1,:)=UC_star;B(K+1,:)=label;D(K+1,:)=UC;
end
for i =1 : 8
    K_star(i)=min(find(A(1:Kpm+1,i)==min(A(1:Kpm+1,i))));
    action_star(i)=B(K_star(i),i);
end
K_star=K_star-1
action_star

【dynamic_iteration.m】
temp1=zeros(1,29);
temp2=zeros(1,29);
for i = 1 : 7
    for j = 1 : 8
        temp1(i)=temp1(i)+(M(i,j,m)*cost_star(j));
        temp2(i)=temp2(i)+(M(i,j,m)*period_star(j));
    end
    ECost(i)=C(i)+O(i)+beta*temp1(i)*T(i);
    EPeriod(i)=T(i)+temp2(i);
end
for i = 8 : 13
    for j = 1 : 8
        temp1(i)=temp1(i)+(Q(1,j,m)*cost_star(j));
    end
temp2(i)=temp2(i)+(Q(1,j,m)*period_star(j));
end
ECost(i)=C(i)+O(i)+beta*temp1(i)*T(i);
EPeriod(i)=T(i)+temp2(i);
end
for i = 14 : 18
    for j = 1 : 8
        temp1(i)=temp1(i)+(Q(2,j,m)*cost_star(j));
        temp2(i)=temp2(i)+(Q(2,j,m)*period_star(j));
        end
        ECost(i)=C(i)+O(i)+beta*temp1(i)*T(i);
        EPeriod(i)=T(i)+temp2(i);
end
for i = 19 : 22
    for j = 1 : 8
        temp1(i)=temp1(i)+(Q(3,j,m)*cost_star(j));
        temp2(i)=temp2(i)+(Q(3,j,m)*period_star(j));
        end
        ECost(i)=C(i)+O(i)+beta*temp1(i)*T(i);
        EPeriod(i)=T(i)+temp2(i);
end
for i = 23 : 25
    for j = 1 : 8
        temp1(i)=temp1(i)+(Q(4,j,m)*cost_star(j));
        temp2(i)=temp2(i)+(Q(4,j,m)*period_star(j));
        end
        ECost(i)=C(i)+O(i)+beta*temp1(i)*T(i);
        EPeriod(i)=T(i)+temp2(i);
end
for i = 26 : 27
    for j = 1 : 8
        temp1(i)=temp1(i)+(Q(5,j,m)*cost_star(j));
        temp2(i)=temp2(i)+(Q(5,j,m)*period_star(j));
        end
        ECost(i)=C(i)+O(i)+beta*temp1(i)*T(i);
        EPeriod(i)=T(i)+temp2(i);
end
for j = 1 : 8
    temp1(28)=temp1(28)+(Q(6,j,m)*cost_star(j));
    temp2(28)=temp2(28)+(Q(6,j,m)*period_star(j));
    end
    ECost(28)=C(28)+O(28)+beta*temp1(28)*T(28);
    EPeriod(28)=T(28)+temp2(28);
    for j = 1 : 8
        temp1(29)=temp1(29)+(P1(1,j)*cost_star(j));
        temp2(29)=temp2(29)+(P1(1,j)*period_star(j));
        end
        ECost(29)=C(29)+O(29)+beta*temp1(29)*T(29);
        EPeriod(29)=T(29)+temp2(29);
temp2(29)=temp2(29)+(P1(1,j)*period_star(j));
end
ECost(29)=C(29)+o(1)+beta*temp1(29)*T(29);
EPeriod(29)=T(29)+temp2(29);
UC=ECost./EPeriod;
UC_star(1)=UC(1);
UC_star(2)=min([UC(2) UC(8)]);
UC_star(3)=min([UC(3) UC(9) UC(14)]);
UC_star(4)=min([UC(4) UC(10) UC(15) UC(19)]);
UC_star(5)=min([UC(5) UC(11) UC(16) UC(20) UC(23)]);
UC_star(6)=min([UC(6) UC(12) UC(17) UC(21) UC(24) UC(26)]);
UC_star(7)=min([UC(7) UC(13) UC(18) UC(22) UC(25) UC(27) UC(28)]);
UC_star(8)=UC(29);
for k = 2 : 7
    for i = 1 : 28
        if UC_star(k)==UC(i)
            label(k)=i;
        end
    end
end
for i = 1 : 7
    cost_star(i)=ECost(label(i));
    period_star(i)=EPeriod(label(i));
end