一个多节点声纳系统中同步时钟机制的可靠性评估和系统优化问题

组号：57　　　　小组成员：桑锐 518021911085

摘要：本文是上海交通大学电子信息与电气工程学院“工程建模与仿真”案例二的论文。本案例以一个多节点声呐系统中的同步时钟机制为主要研究对象，模拟一套由多结点、元件组成的系统，通过使用数学建模思想，蒙特卡洛方法以及马尔科夫链以模拟仿真出该系统，对系统的使用寿命和可靠性做出评估。
关键词：蒙特卡洛方法，马尔科夫链，可靠性

Reliability Evaluation and System Optimization of Synchronous Clock Mechanism in a Multi-node Sonar System

Abstract: This article is the second case of "Engineering Modeling and Simulation" in Shanghai Jiaotong University School of Electronic Information and Electrical Engineering. The case takes the synchronous clock mechanism in a multi-node sonar system as the main research object, simulating a system composed of multiple nodes and components, using mathematical modeling ideas, Monte Carlo method and Markov chain to simulate simulation Out of the system, to assess the service life and reliability of the system.
Keyword: Monte Carlo method, Markov chain, Reliability

1 引言

在工程建模中，一个分布式系统通常被简化成由多个结点构成的系统，每个结点又由多个元件组成。由于使用寿命的限制，元件在实际的工作过程中会出现不同种类的故障，进而影响结点和系统的工作情况。要评估系统的可靠性与使用寿命，主要有两种方法：

1. 列举法。列举所有元件的可能状态，逐一分析系统的工作状态。
2. 蒙特卡洛方法。使用大量随机事件的统计平均来预测系统的工作状态。

随着元件种类的多样化以及数量的增加，枚举法逐渐不适用于问题模型的建立。在本案例中，我们采用蒙特卡洛方法，用大规模随机事件的统计平均来计算系统的可靠性和使用寿命。

本案例研究分布式部署的声呐系统，该系统由多个独立结点构成，各结点物理同构且相互独立。各节点必须严格保持时钟信号的同步才能使系统正常工作。结点的工作状态由内部的两个元件的工作状态所决定。

2 马尔科夫链

2.1 马尔科夫链的原理介绍

马尔科夫链（Markov chain）是数学中具有马尔科夫性质的离散事件随机过程。该过程要求具备“无记忆”的性质：下一状态的概率分布只能由当前状态决定，过去对于预测将来是无关的。这种特定类型的“无记忆性”称作马尔科夫性质[3]。

在马尔科夫链的每一个状态切换过程中，系统根据概率的分布，可以从当前状态转移到另一个状态，也可以保持当前状态不变。对于一个随机过程$X(t)$，在离散取值的时间过程中，对 $t_1 < t_2 < t_3 < \cdots < t_{n-1} < t_n$，$X(t)$的条件概率函数满足等式：

$$\begin{aligned} \mathrm{P}\{X(t_n) = x_n | X(t_{n-1}) = x_{n-1}, \dots, X(t_2) = x_2, X(t_1) = x_1\} \\ = P\{X(t_n) = x_n | X(t_{n-1}) = x_{n-1}\} \end{aligned} \tag{1}$$

即在$X(t_{n-1})$确定的情况下，$X(t_n)$的取值不依赖于t_{n-1}之前时刻的取值[1]。

马尔科夫链较为真实地反映了实际电子元件的工作寿命特性，在极大地简化工程仿真建模难度的同时增加了对元件模拟的精确度，使模拟过程趋向便利化与动态化。

2.2 马尔科夫链的实现

在本案例中，我们假设元件 A 和元件 B 的使用寿命服从负指数分布，使用寿命的概率密度函数为：

$$f_T(t) = \lambda e^{-\lambda t} \ \ (t \geq 0) \tag{2}$$

则对应的分布函数为：

$$F_T(t) = \int_0^t \lambda e^{-\lambda \tau} \, d\tau = 1 - e^{-\lambda t} \ \ (t \geq 0) \tag{3}$$

因此，在$t = t_n$时刻故障未发生的前提下，$t = t_n + t'$时刻故障仍不发生的条件概率是：

$$\begin{aligned} P(t > t_n + t' | t > t_n) &= \frac{P(t > t_n + t', t > t_n)}{P(t > t_n)} = \frac{P(t > t_n + t')}{P(t > t_n)} \\ &= \frac{1 - P(t \leq t_n + t')}{1 - P(t \leq t_n)} = \frac{1 - F_T(t_n + t')}{1 - F_T(t_n)} = \frac{e^{-\lambda(t_n + t')}}{e^{-\lambda(t_n)}} \\ &= e^{-\lambda t'} = 1 - F_T(t') = 1 - P(t \leq t') = P(t > t') \end{aligned} \tag{4}$$

由式(4)得到：元件 A 或 B 在工作了时间t_0后，再正常工作t'的概率与起始时间正常工作t'时间的概率相等，即负指数分布具有无记忆性，元件 A 和 B 的使用寿命符合马尔科夫链的定义。

在实际的仿真过程中，我们只需设定元件 A 和 B 的使用寿命的概率密度函数服从负指数分布，就能保证元件在某一时刻之后发生故障的概率与此时刻之前正常工作的时间无关。

3 蒙特卡洛方法

3.1 蒙特卡洛方法的原理介绍

蒙特卡罗方法（Monte Carlo method），也称统计模拟方法，是 1940 年代中期由于科学技术的发展和电子计算机的发明，而提出的一种以概率统计理论为指导的数值计算方法。是指使用随机数（或更常见的伪随机数）来解决很多计算问题的方法。

当所求解问题是某种随机事件出现的概率，或者是某个随机变量的期望值时，通过某种“实验”的方法，以这种事件出现的频率估计这一随机事件的概率，或者得到这个随机变量的某些数字特征，并将其作为问题的解[2]。

3.2 蒙特卡洛方法的实现

蒙特卡洛方法的解题过程通常为：

1. 构造或描述概率过程。
2. 实现从已知概率分布抽样。
3. 建立各种估计量。

在本案例中，我们对某一确定的结点数量 n，分别计算 100000 套系统的可靠性与寿命，并计算二者的 100000 个平均值作为在结点总数为 n 下的最终仿真结果。

4 系统模型状态的分析

4.1 元件 A 和 B 的状态分析

元件 A 为四状态元件，可能出现三种类型的故障,其使用寿命的概率密度服从参数为λ_A的负指数分布，$\frac{1}{\lambda_A} = 2.72 \times 10^4 (hours)$。正常工作的状态称作 A0，三种故障状态分别称作 A1、A2、A3。在时刻t，这四种状态对应的出现概率为：

$$\{P_{A0}(t), P_{A1}(t), P_{A2}(t), P_{A3}(t)\} \tag{5}$$

其中：

$$P_{A0}(t) = P(T_A \geq t) = \int_t^{+\infty} f_{T_A}(\tau)\, d\tau = e^{-\lambda_A t} \tag{6}$$

又因为：

$$\sum_{i=1}^{3} P_{Ai}(t) = P(T_A < t) = 1 - P_{A0}(t) \tag{7}$$

所以：

$$P_{A1}(t) = P_{EA1} \cdot P(T_A < t) = 0.3 \cdot (1 - e^{-\lambda_A t}) \tag{8}$$

$$P_{A2}(t) = P_{EA2} \cdot P(T_A < t) = 0.3 \cdot (1 - e^{-\lambda_A t}) \tag{9}$$

$$P_{A3}(t) = P_{EA3} \cdot P(T_A < t) = 0.4 \cdot (1 - e^{-\lambda_A t}) \tag{10}$$

类似地，元件 B 为三状态元件，可能出现两种类型的故障,其使用寿命的概率密度服从参数为λ_B的负指数分布，$\frac{1}{\lambda_B}=3.32\times10^5(hours)$。正常工作的状态称作 B0，两种故障状态分别称作 B1、B2。在时间t时，各状态出现的概率为：

$$P_{B0}(t)=P(T_B\geq t)=\int_t^{+\infty}f_{T_B}(\tau)\,d\tau=e^{-\lambda_B t} \tag{11}$$

$$P_{B1}(t)=P_{EB1}\cdot P(T_B<t)=0.33\cdot(1-e^{-\lambda_B t}) \tag{12}$$

$$P_{B2}(t)=P_{EB2}\cdot P(T_B<t)=0.67\cdot(1-e^{-\lambda_B t}) \tag{13}$$

4.2 结点的状态分析

结点的状态由节点内部 A、B 元件的状态所决定。经分析，节点性能状态可以归为 6 种：

$$G_{Ni}(t)=\{g_{N0},g_{N1},g_{N2},g_{N3},g_{N4},g_{N5}\} \tag{14}$$

元件与结点状态的映射关系见表 1。

表 1 元件-结点状态的映射关系表[1]

元件 A 状态	元件 B 状态	结点状态	状态别名
g_{A0}	g_{B0}	g_{N0}	g_{PF}
	g_{B1}	g_{N3}	g_{MO}
	g_{B2}	g_{N1}	g_{SO}
g_{A1}	g_{B0}	g_{N1}	g_{SO}
	g_{B1}	g_{N5}	g_{FB}
	g_{B2}	g_{N1}	g_{SO}
g_{A2}	g_{B0}	g_{N2}	g_{DM}
	g_{B1}	g_{N3}	g_{MO}
	g_{B2}	g_{N4}	g_{DN}
g_{A3}	g_{B0}	g_{N4}	g_{DN}
	g_{B1}	g_{N4}	g_{DN}
	g_{B2}	g_{N4}	g_{DN}

4.3 系统的状态分析

按照案例二规定，系统有 4 种工作状态，分别是：系统确定不能有效工作的状态G_{sys1}，系统确定能有效工作的状态G_{sys2}，系统恰能有效工作的状态G_{sys3}和系统恰不能有效工作的状态G_{sys4}。

根据所有组成系统的结点状态，分别计算出$g_{N0},g_{N1},g_{N2},g_{N3},g_{N4},g_{N5}$状态的结点数量，依据案例二规定的结点-系统状态映射关系，可以得出系统的工作状态。

5 程序设计

5.1 程序设计思路概述

程序设计的主要思路见图 1。

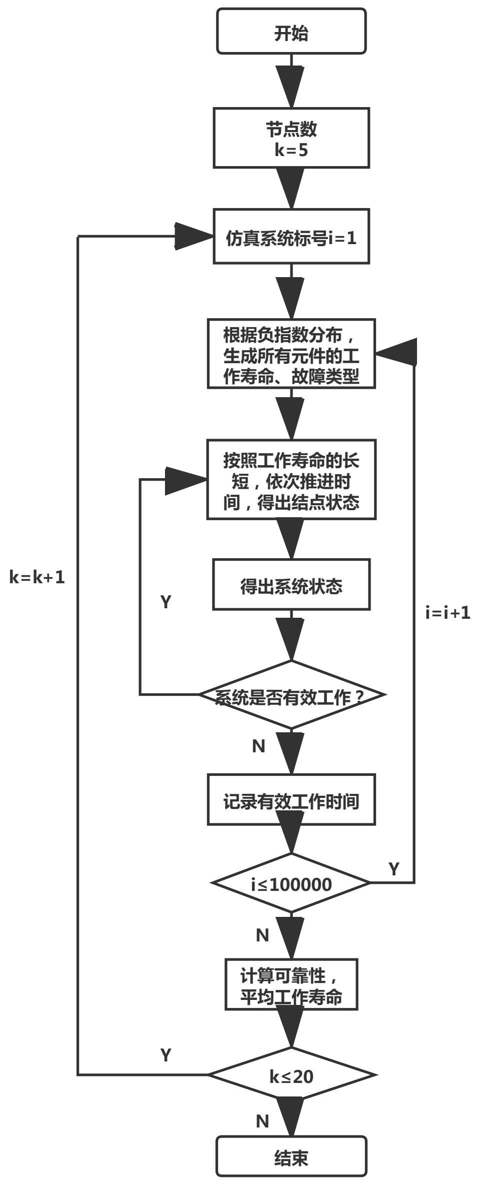

图 1 程序设计流程图

在主程序中，我们使用第一层循环来计算在不同结点数量的情况下，系统的工作寿命与可靠性。在第二层循环中，我们令循环遍量循环 100000 次，来模拟 100000 个相同且独立系统的工作寿命与可靠性，并对所有输出值取平均，作为该结点数量下系统的工作寿命与可靠性。

程序代码详见附录。

5.2 使用蒙特卡洛方法生产元件的使用寿命

在本案例中，我们使用 MATLAB 的 exprnd()函数，生成指定大小的以负指数分布的随机数矩阵，并令该随机数矩阵作为元件 A 和元件 B 发生故障的时间。之后，再利用伪随机数的产生，决定每个元件的故障类型。

相比较于使用循环的方式实现这一步骤，利用蒙特卡洛方法与 exprnd()函数显著提升了程序的运行效率。

5.3 变化步长的方法

在程序设计中，我们采用变化步长的方法对系统的状态进行模拟。具体实现如下：根据 n 个结点中 A、B 元件发生故障的时间与故障类型，利用 MATLAB 中的 sort()函数，从中依次选取时间最早的故障，确定该时刻结点的工作状态，进而分析得到系统的工作状态。若当系统状态变为故障，则取该时刻作为系统的工作寿命。

考虑到该模型的漏洞，我们对系统“工作寿命无限”的情况进行补充。当系统的工作寿命超过 90000 小时时，我们将其工作寿命视为 90000 小时。

5.4 程序的输出

经过模拟仿真，程序最终输出：系统的最大可靠性，在获得最大可靠性时对应的系统结点数量，系统的最大平均工作寿命，在获得最大平均工作寿命时对应的系统结点数量。

系统的可靠性由以下公式定义：

$$R(w)=\frac{\sum_{i=1}^{100000}1(T_f^i\geq w)}{100000} \tag{15}$$

其中$w=25000$小时。系统的平均公式由以下公式定义：

$$E(T_f)=\frac{\sum_{i=1}^{100000}T_f^i}{100000} \tag{16}$$

5.5 程序的优化与调整

调用 MATLAB 内置函数的方式往往会精简代码结构，减少代码量。本次程序设计中，在通过结点的工作状态确定系统的工作状态的代码实现上，可以使用 MATLAB 的 find()函数取代程序代码中的遍历操作。具体详见附录中被注释的代码段。

但是，经过实际对比，使用了 find()函数后，程序单次运行时间增加约 30 秒，执行效率约降低 40%。此现象说明 find()函数在被调用时，会消耗较多的计算资源。

在本次代码中，涉及大量的逻辑操作。选择合适的逻辑关系，可有效降低运算复杂度，提高程序运行效率。

6 结论

程序运行 30 次的结果见表二。系统可靠性与工作寿命随结点数量变化曲线见图 2、图 3。

表二 程序 30 次运行结果的平均值

结点数量	Rw（可靠性）平均值	Et（平均寿命）平均值（小时）
5	0.0768	9634.79
6	0.2356	19186.35
7	0.4232	28290.73
8	0.5887	36926.01
9	0.7172	44511.99
10	0.8145	50779.15
11	0.8575	56071.58
12	0.8856	60167.22
13	0.9004	63185.03
14	0.9076	65357.24
15	0.9098	66811.16
16	0.9059	67843.25
17	0.9012	68343.68
18	0.8959	68459.23
19	0.8873	68199.87
20	0.8830	67649.01

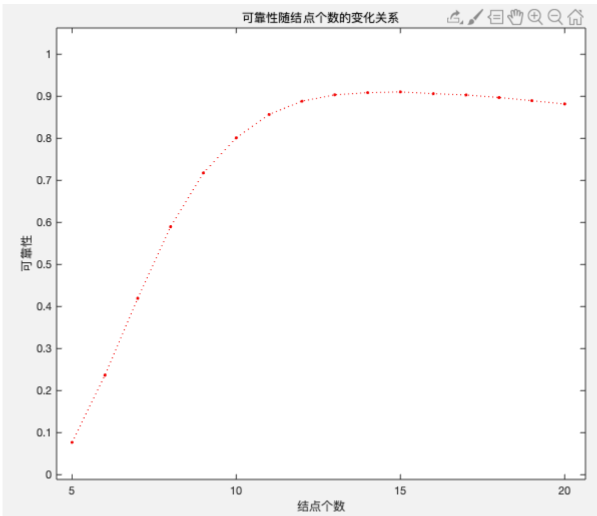

图 2 系统可靠性随结点数量变化关系图

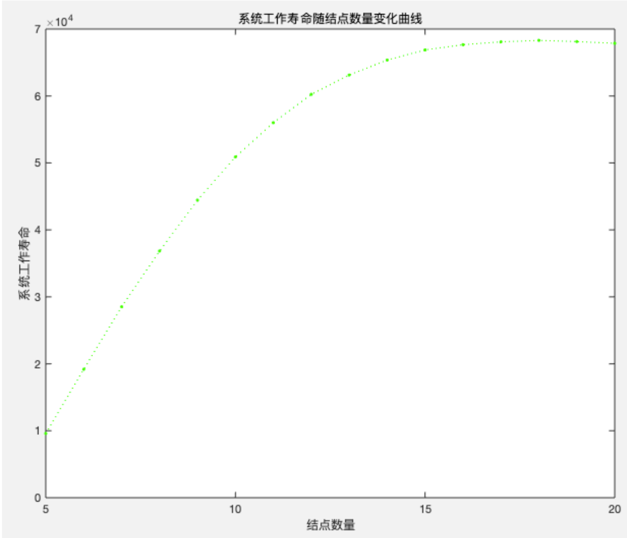

图 3 系统工作寿命随结点数量变化曲线

分析可知，当结点数量为 14、15 时，系统达到最大可靠性，约为 91%；当结点数量为 17、18 时，系统达到最大工作寿命，约为 68460 小时。整体来看，系统的可靠性与工作寿命均随着结点数量的增加呈现出先增加后减小的变化趋势。

7 参考文献

[1] 上海交通大学电子工程系，工程问题建模与仿真之案例 2_V2.8 20171022。ftp://202.120.39.248

[2] 蒙特卡洛方法_百度百科。https://baike.baidu.com/item/%E8%92%99%E7%89%B9%C2%B7%E5%8D%A1%E7%BD%97%E6%96%B9%E6%B3%95/8664362?fr=aladdin;

[3] 维基百科——马尔可夫链：https://zh.wikipedia.org/wiki/%E9%A9%AC%E5%B0%94%E5%8F%AF%E5%A4%AB%E9%93%BE

8 附录

程序代码：

```
% 切换器 A 的使用寿命Ta指数分布, 1/lambdaA = 2.72e4
% 条件概率 Pa1=Pa2=0.3, Pa3=0.4
% 切换器 B 的使用寿命Tb指数分布, 1/lambdaA = 3.32e5
% 条件概率 Pb1=0.33, Pb2=0.67
% 最少的工作结点数为k=5
% 元件一旦发生故障, 故障类型确定, 不变；故障均不可修复
% A 在 t 时刻正常工作的概率：exp(-lambda * t), 故障的总概率：1 - exp(-
lambda * t)
% B 同 A
```

```matlab
% 模拟s套系统的运行状况, s>=1e5

tic

k = 5;
w = 25000;
total_number_of_system = 1e5;
Rw = zeros(1,20); % 存储可靠性
Et = zeros(1,20); % 存储平均工作寿命
lambdaA = 1/2.72e4;
lambdaB = 1/3.32e5;
n = 5;

for num_of_node = 5:20 % 结点总数为5~20
    work_time = zeros(1,total_number_of_system);
    for num_of_sys = 1:total_number_of_system % 使用蒙特卡洛法, 模拟10000
套系统的运行状态
        Ta = exprnd(1/lambdaA, 1, num_of_node);
        Tb = exprnd(1/lambdaB, 1, num_of_node);

        % 元件 A/B 的状态
        % 判断各个结点中, 元件 A 的故障种类
        A_error = ones(1,num_of_node);
        for i = 1:num_of_node
            random_num = rand();
            if random_num <=0.3
                continue;
            elseif random_num <= 0.6
                A_error(i) = 2;
            else
                A_error(i) = 3;
            end
        end
        % 判断各个结点中, 元件 B 的故障种类
        B_error = ones(1,num_of_node);
        for i = 1:num_of_node
            random_num = rand();
            if random_num <=0.33
                continue;
            else
                B_error(i) = 2;
            end
        end
```

```
        node_state = zeros(1, num_of_node); % store the states of each
node
        A_state = zeros(1, num_of_node); % store the states of A
        B_state = zeros(1, num_of_node); % store the state of B
        % The state of node is determined by the state of A/B, so we
only
        % consider the time when A/B changes its own state
        times = [Ta Tb];
        % sort() 改变原顺序，按照数字从大到小排列原list，并将排列后的每个元素的
原索引保存在index中，从1开始indexing
        [times, index] = sort(times);

        for i = 1:2*num_of_node % 每一个时间结点都需要判断
            % 先把 A,B 元件的“此时刻的”状态搬运到 A_state & B_state 中
            label_of_node = index(i); % 默认是 A 结点的结点标号；是 B ，则需
映射
            if label_of_node <= num_of_node
                % 说明这个时间点是元件 A 的状态发生改变
                A_state(label_of_node) = A_error(label_of_node);
            else
                % 此时间点元件 B 的状态发生改变
                label_of_node = label_of_node - num_of_node; % 把
num_of_node+1 ~ 2*num_of_node 映射到 1 ~ num_of_node
                B_state(label_of_node) = B_error(label_of_node);
            end

            % Determine the state of nodes from A&B of each node
            if A_state(label_of_node) == 0
                if B_state(label_of_node) == 0
                    node_state(label_of_node) = 0;
                elseif B_state(label_of_node) == 1
                    node_state(label_of_node) = 3;
                else
                    node_state(label_of_node) = 1;
                end
            elseif A_state(label_of_node) == 1
                if B_state(label_of_node) == 0
                    node_state(label_of_node) = 1;
                elseif B_state(label_of_node) == 1
                    node_state(label_of_node) = 5;
                else
                    node_state(label_of_node) = 1;
                end
            elseif A_state(label_of_node) ==2
```

```
        if B_state(label_of_node) == 0
            node_state(label_of_node) = 2;
        elseif B_state(label_of_node) == 1
            node_state(label_of_node) = 3;
        else
            node_state(label_of_node) = 4;
        end
    else
        if B_state(label_of_node) == 0
            node_state(label_of_node) = 4;
        elseif B_state(label_of_node) == 1
            node_state(label_of_node) = 4;
        else
            node_state(label_of_node) = 4;
        end
    end

    % Calculate the amount of different state of nodes
    pf = 0; % perfectly functioning
    so = 0; % slave only
    dm = 0; % disable/master
    mo = 0; % master only
    dn = 0; % disable node
    fb = 0; % failed bus
    for m = 1:num_of_node
        if node_state(m) == 0
            pf = pf + 1;
        elseif node_state(m) == 1
            so = so + 1;
        elseif node_state(m) == 2
            dm = dm + 1;
        elseif node_state(m) == 3
            mo = mo + 1;
        elseif node_state(m) == 4
            dn = dn + 1;
        else
            fb = fb + 1;
        end
    end
%          % 尝试使用find ()
%         pf = size(find(node_state==0),2);
%         so = size(find(node_state==1),2);
%         dm = size(find(node_state==2),2);
%         mo = size(find(node_state==3),2);
```

```
%           dn = size(find(node_state==4),2);
%           fb = size(find(node_state==5),2);

          % Determine the state of the system from the state of the
nodes:
          sys_state = 0;
          if fb>=1 || mo>=2 || (pf+mo+dm)==0 || (pf+so+((mo+dm)>0))<k
              sys_state = 1;
          elseif fb==0 && (((mo==1 && pf+so>=k-1) ||((mo==0 && pf>=1
&& (pf+so)>=k) || (mo==0 && pf==0 && dm>=1 && so>=k-1)))
              sys_state = 2;
          elseif (fb+mo)==0 && (pf>=1 && pf+so==k-1 && dm>=1)
              possibility = dm / (dm + pf);
              if rand() <= possibility
                  sys_state = 3;
              else
                  sys_state = 4;
              end
          end

          % Whether to jump out and record time
          total_time = 0;
          if sys_state == 1 || sys_state == 4
              total_time = times(i);
              break;
          end
      end

      % Revise the bug
      if total_time == 0 || total_time > 90000
          total_time = 90000;
      end

      work_time(num_of_sys) = total_time;
  end

  % calculate the reliability
  reliable_sys = 0;
  for i = 1:total_number_of_system
      if work_time(i) >= w
          reliable_sys = reliable_sys + 1;
      end
  end
  % store the reliability and the average working time of the
```

```
system
    Rw(num_of_node) = reliable_sys/total_number_of_system;
    Et(num_of_node) = sum(work_time)/total_number_of_system;
end

% print
max_Rw = max(Rw);
max_num_of_node_Rw = find(Rw==max_Rw, 1);
max_Et = max(Et);
max_num_of_node_Et = find(Et==max_Et, 1);

fprintf('最大可靠性为%7.4f.\n', max_Rw)
fprintf('此时对应节点数为%d.\n', max_num_of_node_Rw)
fprintf('最大平均工作寿命为%7.4f.\n', max_Et)
fprintf('此时对应节点数为%d.\n', max_num_of_node_Et)
toc
```