# 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

——基于蒙特卡洛方法与马尔科夫链

戴天杰 519021910734

**摘 要：**本文着眼于系统的可靠性评估和系统优化问题，笔者从基本元器件的状态出发，逐层上溯至整个系统的工作状态。在研究过程中，本文首先利用统计推断中常见的蒙特卡洛方法与马尔科夫转移链，建立理论模型，通过MATLAB仿真，全面评估系统的实际性能。最后，笔者根据“总线阻塞”构造了新节点结构，对系统结构进行了优化。

**关键词：**蒙特卡洛方法，马尔可夫链

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

**ABSTRACT:** This paper focuses on the problem of system reliability evaluation and system optimization. Starting from the state of basic components, the author traces the working state of the whole system layer by layer. In the research process, this paper first uses the common *Monte Carlo* Method and *Markov Chain* in statistical inference to establish a theoretical model, and comprehensively evaluates the actual performance of the system through MATLAB simulation. Finally, author constructs a new node structure based on *Bus Blocking* and optimizes the system.

**Keywords:** *Monte Carlo* Method, *Markov Chain*

## 1 理论模型的构建

### 1.1 问题背景

在国防、军事等领域，出于审慎稳妥的考量，部署于海底的多节点固定声纳网络一般采取冗余设计，因而，系统的实际工况与总节点 $n$ 的数量息息相关。由常识可以知晓，若总节点数相对较少，当存在元件发生故障时，备份元件往往不足，则极易导致整个系统故障；若总节点数过多，容易导致“总线阻塞”，系统发生致命故障的风险会加大，且会大幅提高研制成本。为了寻找均衡点，我们需要通过建立模型仿真模拟来得出最优解。

## 1.2 模型构建

为简化运算，案例说明中仅把切换器 A 和 B 作为不可靠元件进行研究。切换器可视作多状态元件，两个不同的切换器构成了节点，故单个节点也呈现出多状态的特征，由多个节点构成的系统整体故可看作为是一个多状态系统。三者的具体联系如图 1 所示。

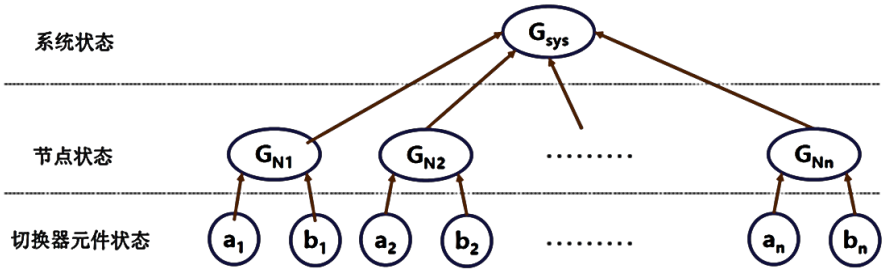



**图 1 模型建立层次图[1]**

接下来，笔者对三个层次的理论建模进行简要概括，具体内容详见《工程问题建模与仿真之案例课题 2_V2.11 20210420》[2]。

### 1.2.1 元件层次

由案例说明可知，切换器 A 是四状态元件，$g_{A0}$、$g_{A1}$、$g_{A2}$和$g_{A3}$分别表示 A 正常工作、故障 A1、A2 和故障 A3 这四种不同状态；切换器 B 是三状态元件，与 A 类似，$g_{B0}$、$g_{B1}$和$g_{B2}$分别表示 B 正常工作、故障 B1 和故障 B2 这三种不同状态。

由于切换器元件使用寿命的概率密度分布都遵从参数已知的负指数分布，因而我们可以计算出产生不同状态的概率。

### 1.2.2 节点层次

在本案例中，一个节点包含 A、B 切换器各 1 个，故其状态由内部切换器的不同状态组合一致确定。由案例说明可知，12 种不同切换器状态组合对应 6 种节点状态，由全概率公式可知，$t$ 时，节点处于状态 $k$ 的概率 $p_{Nk}(t)$ 为：

$$p_{Nk}(t)=\sum_{(i,j)\in\{(g_{Ai},g_{Bi})\to g_{Nk}\}} P_{Ai}(t)\cdot P_{Bj}(t) \qquad k=0,1...5、\ i=0,...3、\ j=0,1,2$$

### 1.2.3 系统层次

在本案例中，系统的工况由其所包含的 $n$ 个节点共同决定，按照节点状态的数量等准则可以划分为 4 种不同状态，即$G_{sys1}$、$G_{sys2}$、$G_{sys3}$和$G_{sys4}$。

## 1.3 指标说明

为了量化系统实际性能，案例说明中定义了如表 1 中所述的 3 种衡量指标。

**表 1 系统性能衡量指标释义**

| 名　称 | 释　义 |
|---|---|
| 首次失效时间/TTF | 系统从初始时间到首次发生失效的时间，上限为 9 万小时 |
| 平均首次失效时间/MTTF | 平均工作寿命，即首次失效时间的统计平均 |
| 系统可靠性/$R$（$w$） | 系统工作寿命超过某一定值 $w$ 的概率 |

# 2 基于蒙特卡洛算法的仿真求解

蒙特卡洛方法（*Monte Carlo* Method），也称统计模拟方法，是二十世纪四十年代中期由于科学技术的发展和电子计算机的发明，而被提出的一种以概率统计理论为指导的一类非常重要的数值计算方法。蒙特卡洛方法在金融工程学、计量经济学、计算物理学（如粒子输运计算、量子热力学计算、空气动力学计算）等领域应用广泛[3]。

在业界应用中，蒙特卡洛方法又衍生为了两种：第一种是时间按固定步长（颗粒度）演进，随机模拟每个元件的状态，最后推得整体工况；第二种是时间按变化步长演进，初始化时随机生成元件的故障类型和使用寿命，对故障点进行处理后更新系统工况。在本次仿真求解实践过程中，笔者选用的是第二种方法，对不同节点数下的 10 万套系统分别进行模拟。

## 2.1 算法步骤说明

笔者以已知节点数下的单套系统为例进行算法步骤说明，整个流程可以分为切换器状态、寿命的确定，节点状态的确定，系统寿命的确定等过程。其中，系统状态与寿命的确定是整个流程的核心，蒙特卡洛方法正是模拟这一过程的有效手段。

### 2.1.1 切换器状态、寿命的确定

对该套系统而言，我们需要随机生成最基本元器件——切换器 A、B 的初始状态。已知切换器 A、B 寿命的概率密度分布，以及发生故障时处于不同状态的条件概率，我们可以随机产生切换器的寿命，若发生故障时，随机生成故障类型。我们总计需要生成（2×节点数）个最基本元器件的寿命与类型，并将这些初始数据保存下来。

### 2.1.2 节点状态的确定

节点状态的确定相对较易理解，根据案例说明中的对应关系，我们很容易得出包含该 A、B 切换器的节点的状态。

### 2.1.3 系统状态、寿命的确定

由案例说明可得，当系统在进行蒙特卡洛模拟时，若系统状态为 2 或 3 时，系统则能正常工作；当状态首次切换到 1 或 4 时，系统失效，此即为模拟终止条件。

在一轮蒙特卡洛过程中，我们需要找到（2×节点数）个切换器中寿命最小的那个，当时间为其寿命时，其状态发生变化。为了表示已经处理过这个元件，我们将其寿命设置为无穷大（+*inf*），相当于删除了这个元件。与此同时，我们需要更新与该被处理元件相关联的节点状态，再根据处于不同状态的节点数量，确定系统变化后的状态。若状态切换到 1 或 4，则模拟结束，此时该切换器的寿命即为失效时间[1]。

该步骤具体的算法流程如图 2 中的伪代码所示。

**Algorithm 1** *Monte Carlo* Adopted in 2.1.3

```
function SYSTEM(n, Life, Cons)
    Expectlife ← 0
    Consys ← 2
    while Consys == 2 or 3 do
        if all the switches have been tackled then
            ExpectLife ← 90000
        end if
        minimum ← min(Life)
        [x, y] ← corresponding coordinate
        Life(x, y) ← +inf
        Update conditions of the corresponding node
        Count numbers of nodes with different conditions
        Update Consys according to the Guide Book
        if Consys == 1 or 4 then
            ExpectLife ← minimum
        end if
    end while
    return ExpectLife
end function
```

**图 2 确定系统寿命的伪代码**

至于求解 10 万个系统，我们只需将上述 2.1.1~2.1.3 步骤重复 10 万次即可，工作寿命的均值即为 MTTF，寿命超过 25000 小时的系统占比即为可靠性。将节点数从 5 调整至 20，我们可以从中分别找出 MTTF、系统可靠性最高的两个数值，即为案例说明中要求得到的解。

## 2.2 代码结构说明

基于上述算法思路以及数据处理要求，笔者进行了代码实现。为了更加清晰地表述代码中不同模块的功能，笔者将自定义函数的名称与功能列写在表 2 中。

**表 2 代码涉及的自定义函数名称与功能说明**

| 函数名称 | 功能说明 |
|---|---|
| *main* | 顶层函数。计算 5~20 个节点时，10 万套系统的 MTTF 与可靠性 |
| *SwitchA* | 利用参数已知的负指数分布，随机产生元件 A 的寿命与状态 |
| *SwitchB* | 利用参数已知的负指数分布，随机产生元件 B 的寿命与状态 |
| *Node* | 利用元件 A、B 的状态，得出节点的状态 |
| *Record* | 记录所有元件 A、B 的寿命与状态 |
| *System* | 根据元件的寿命、状态与节点的状态，得出系统的寿命与状态 |

其中，不同函数之间的逻辑层次、组织架构如图 3 所示。

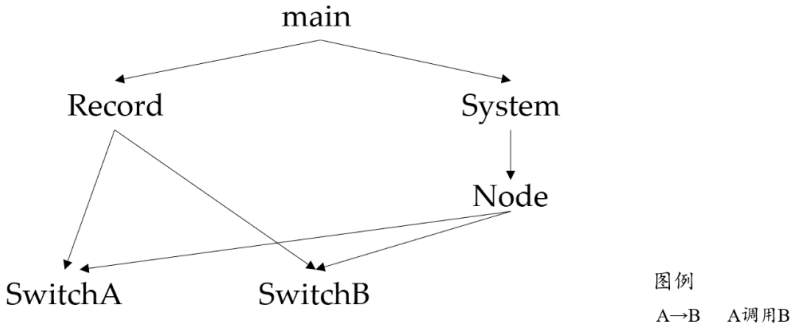



**图 3 函数间的调用关系**

## 2.3 运行结果探讨

代码实现后，笔者进行多次仿真实验，表 3 列出了某次代码运行结果。

**表 3 代码运行示例**

| 点数 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|
| MTTF | 11736.65 | 22879.15 | 32908.74 | 41930.50 | 49652.76 | 56234.49 | 61403.84 | 65516.05 |
| 系统可靠性 | 0.1172 | 0.3276 | 0.5409 | 0.7052 | 0.8205 | 0.8858 | 0.9241 | 0.9423 |
| 点数 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| MTTF | 68796.74 | 71003.74 | 72815.65 | 73948.09 | 74522.29 | 75001.81 | 75036.88 | 74772.52 |
| 系统可靠性 | 0.9499 | 0.9516 | 0.9496 | 0.9467 | 0.9433 | 0.9414 | 0.9366 | 0.9309 |

由表 3 可以看出，在该次运行过程中，$E(T_f)$ 最大值出现在节点数为 19 时，约为 75306.88；$R(w)|_{w=25000h}$ 最大值出现在节点数为 14 时，约为 95.16%。

为了避免偶然误差造成影响，笔者重复进行了 8 次实验，并记录下当 MTTF、系统可靠性取最大时的节点数。

**表 4 MTTF、系统可靠性最大时的节点数**

| 次数 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| $MTTF_{Max}$ 节点数 | 19 | 19 | 19 | 19 | 19 | 19 | 19 | 19 |
| 可靠性 $_{Max}$ 节点数 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 |

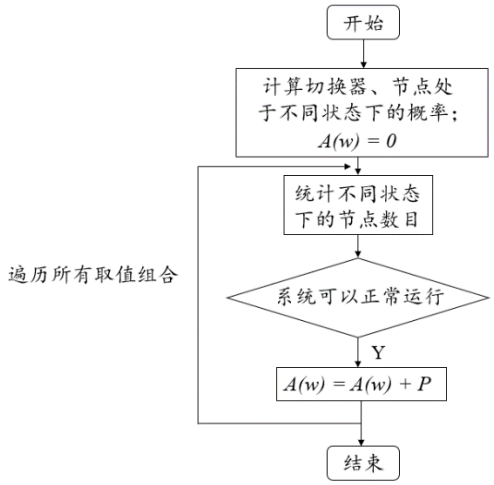
显然，由表 4 可得，$E(T_f)$ 最大的节点数是 19，取均值可得，平均寿命最长大约为 75010.6；$R(w)|_{w=25000h}$ 最大的节点数是 14，取均值可得，系统可靠性大约为 95.1%。

总之，当总节点增加时，两个指标都呈现出快速上升——缓慢上升——略微下降的趋势。这表明：在该声纳系统中，冗余节点的数量绝非越多越好，否则会增加总线阻塞现象的概率。

## 3 系统可用性的理论值求解

在实际应用中，用数学直接推导系统可靠性的难度极大，且过程较繁琐。在实验材料[4]中，袁焱、李安琪老师利用概率统计知识，通过理论推断与数值求解，得出了含有 $n$ 个节点的系统在时刻 $w$（25000 小时）时可用性的求解方法，以此作为系统可靠性的近似。

显然，根据前述内容，我们可以得到切换器、节点位于不同状态下的概率，再将能够使得系统正常工作的组合对应的概率加总即为可用性。具体计算流程如图 4 所示。




**备 注：**

①当系统状态为 2 时：

$$P=\underbrace{C_n^{k_1}C_{n-k_1}^{k_2}C_{n-k_1-k_2}^{k_3}\cdots}_{\text{共计6项}}\underbrace{P_1^{k_1}P_2^{k_2}P_3^{k_3}\cdots}_{\text{共计6项}}$$

②当系统状态为 3 时：

$$P=\underbrace{C_n^{k_1}C_{n-k_1}^{k_2}C_{n-k_1-k_2}^{k_3}\cdots}_{\text{共计6项}}\underbrace{P_1^{k_1}P_2^{k_2}P_3^{k_3}\cdots}_{\text{共计6项}}\times\frac{P_{DM}}{P_{DM}+P_{PF}}$$

**图 4 可用性计算流程图**

按照上述思路，笔者计算求解出了系统总节点数目在 5~20 之间，$w$=25000h 时的系统可用性，具体结果如表 5 所示。

**表 5 系统可用性与可靠性对比**

| 点数 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|
| 系统可用性 | 0.1281 | 0.3401 | 0.5516 | 0.7162 | 0.8260 | 0.8915 | 0.9272 | 0.9446 |
| 系统可靠性 | 0.1172 | 0.3276 | 0.5409 | 0.7052 | 0.8205 | 0.8858 | 0.9241 | 0.9423 |
| 点数 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 系统可用性 | 0.9519 | 0.9536 | 0.9525 | 0.9499 | 0.9467 | 0.9430 | 0.9392 | 0.9352 |
| 系统可靠性 | 0.9499 | 0.9516 | 0.9496 | 0.9467 | 0.9433 | 0.9414 | 0.9366 | 0.9309 |

结合指导材料与表 5，我们可以发现，系统的可用性指标稍稍大于可靠性，这与系统中存在“复活”现象有关。在系统可靠性的计算过程中，我们选择系统首次失效时间作为系统的工作寿命；在系统可用性的计算过程中，我们仅把目光聚焦于 $w$ 时刻系统的工况，统计此时处于不同工况的节点数目，而忽略了之前失效的情况。当然，发生“复活”现象的概率较小，因而两组指标相近，故而我们可以将可用性作为可靠性的推断值与一种近似解[4]。

## 4 基于切换器 B 的系统优化

在指导材料[1]中，笔者了解到，“总线阻塞”对系统寿命影响非常大；与此同时，“总线阻塞”与切换器 B，尤其是故障 B1 类型密切相关。有鉴于此，笔者在切换器 A、B 后串联一个同样的切换器 B（记作 B'），作为新的节点内部结构，新节点内部结构如图 5 所示。表 6 则呈现了切换器 B、B' 处于不同状态组合时整体的状态。

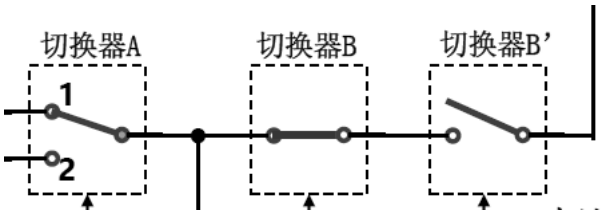



**图 5 切换器 B 的节点结构优化**

**表 6 切换器 B 与 B'状态与整体状态**

| 整体状态 | $B$、$B'$可能的对应状态组合 |
|---|---|
| *B1* | （*B1*、*B1*） |
| *B2* | 两者中至少有一个 *B2* |
| *B0* | 剩余 3 种组合 |

紧接着，笔者在此思想的指导下开展实验。首先开展可用性计算，与原系统相比，仅需根据表 6 修改 B0、B1、B2 状态概率即可，以 B1、B2 为例：

$$P_{B1(New)} = P_{B1}^2$$

$$P_{B2(New)} = 2 \times P_{B2} - P_{B2}^2$$

运行可用性代码，具体数据如表 7 所示。显然，系统的可用性得到了明显提升，在节点数为 11 时，就超过了表 5 中旧系统可用性的最大值。

**表 7 新系统的可用性**

| 点数 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|
| 新系统 | 0.1305 | 0.1305 | 0.5638 | 0.7335 | 0.8479 | 0.9177 | 0.9572 | 0.9784 |
| 点数 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 新系统 | 0.9892 | 0.9946 | 0.9971 | 0.9983 | 0.9988 | 0.9991 | 0.9991 | 0.9991 |

然后，笔者对新系统的工作寿命、可靠性进行研究。与之前步骤最大的不同在于确定 B、B'的整体寿命和整体状态。改进切换器 B 的指标计算方法如图 6 中的伪代码所示。

**Algorithm 2** Whole Life and Condition of *Updated Switch B* in Chapter 5

```
if LifeB < LifeB' then
    if ConditionB = 1 then
        LifeWholeB ← LifeB'
        ConditionWholeB ← ConditionB'
    else
        LifeWholeB ← LifeB
        ConditionWholeB ← ConditionB
    end if
else
    if ConditionB' = 1 then
        LifeWholeB ← LifeB
        ConditionWholeB ← ConditionB
    else
        LifeWholeB ← LifeB'
        ConditionWholeB ← ConditionB'
    end if
end if
```

**图 6 改进切换器 B 的计算方法**

如图 7 所示，系统的工作寿命直逼最大寿命 90000 小时，可靠性渐进于 1，效果非常好。

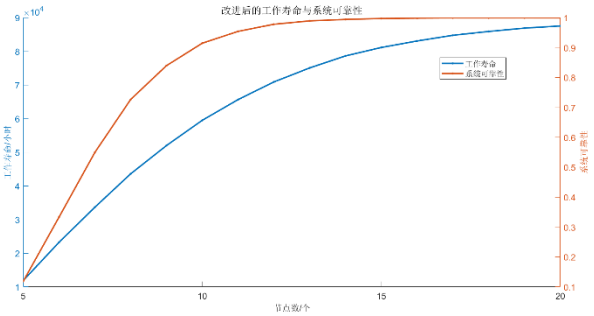



**图 7 改进系统的工作寿命与可靠性**

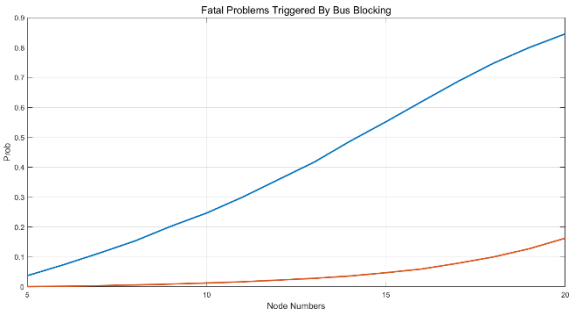



**图 8 总线阻塞引发的故障占比对比**

最后，笔者对旧系统与新系统中，由于总线阻塞引发整体故障的情况进行了统计，即节点状态 3、5 在系统故障时的占比。由图 8 可得，经过优化之后，系统总线阻塞引起的故障占比得到了显著的下降，从超过 80%降到了不到 20%，实现了预设目标。

综上所述，优化后的系统有着显著的优势，无论是可用性、可靠性还是工作寿命，都有着明显的增幅。此外，系统总线阻塞的问题得到了缓解。因而，该优化策略是非常成功的。

## 5 致谢

感谢两位老师一学期以来的付出；也感谢周睿文学长和倪泽林同学，在成稿过程中与他们进行了很多的交流。值得一提的是，在今年美赛中，周学长获得了 O 奖，倪同学和笔者共同获得了 F 奖，参加美赛对培养建模思想、完成课程任务亦有着一定的作用。

## 参考资料

[1] 工程问题建模与仿真讲座 16_关于案例 2 课题求解方案的几点探讨.

[2] 工程问题建模与仿真之案例课题 2_V2.11 20210420.

[3] http://baike.baidu.com/item/蒙特·卡罗方法/8664362?fromtitle=蒙特卡罗方法&fromid=214674&fr=aladdin.

[4] 指导材料：案例 2 系统可用性数值的理论求解方法介绍 20200420.

## 附 录

## main.m

```
function [ LA , CA ] = SwitchA(  )
LA = exprnd(37000);
if LA < 90000
    thereshold = rand;
    if thereshold <= 0.26
        CA = 1;
    elseif thereshold <= 0.52
        CA = 2;
    else
        CA = 3;
    end
else
    LA = 90000;
    CA = 0;
end
end
```

```
function [ LB , CB ] = SwitchB(  )
LB = exprnd(480000);
```

```
if LB < 90000
    thereshold = rand;
    if thereshold <= 0.35
        CB = 1;
    else
        CB = 2;
    end
else
    LB = 90000;
    CB = 0;
end
end
```

---

```
function [ LB , CB ] = SwitchB(  )
LB1 = exprnd(480000);
LB2 = exprnd(480000);

if LB1 < 90000
    thereshold = rand;
    if thereshold <= 0.35
        CB1 = 1;
    else
        CB1 = 2;
    end
else
    LB1 = 90000;
    CB1 = 0;

if LB2 < 90000
    thereshold = rand;
    if thereshold <= 0.35
        CB2 = 1;
    else
        CB2 = 2;
    end
```

```
else
    LB2 = 90000;
    CB2 = 0;

if LB1 < LB2
    if CB1 == 1
        LB = LB2;
        CB = CB2;
    else
        LB = LB1;
        CB = CB1;
    end
else
        if CB2 == 1
            LB = LB1;
            CB = CB1;
        else
            LB = LB2;
            CB = CB2;
end
```

---

```
function [ Life , Cons ] = Record( n )
Life = zeros(n,2);
Cons = zeros(n,2);
for i = 1 : n
    [Life(i,1),Cons(i,1)] = SwitchA();
    [Life(i,2),Cons(i,2)] = SwitchB();
end
end
```

---

```
function [ Con ] = Node( CA , CB )
ConRecordMatrix = zeros(4,3);
ConRecordMatrix(1,1) = 0;  ConRecordMatrix(1,2) = 3;
ConRecordMatrix(1,3) = 1;  ConRecordMatrix(2,1) = 1;
```

```
ConRecordMatrix(2,2) = 5;  ConRecordMatrix(2,3) = 1;
ConRecordMatrix(3,1) = 2;  ConRecordMatrix(3,2) = 3;
ConRecordMatrix(3,3) = 4;  ConRecordMatrix(4,1) = 4;
ConRecordMatrix(4,2) = 4;  ConRecordMatrix(4,3) = 4;
Con = ConRecordMatrix(CA+1,CB+1);
end
```

---

```
function [ ExpectLife ] = System( n , Life , Cons )
ExpectLife = 0;
ConSys = 2;
AllNodes = zeros(n,1);
while ConSys ==2 || ConSys == 3
    if sum (Life(:) == +inf) == 2 * n
        ExpectLife = 90000;
        break;
    end
    minimum = min(min(Life));
    [x,y] = find(Life == minimum);
    if y == 1
        ConA = Cons(x,y);
        if Life(x,2) == +inf
            ConB = Cons(x,2);
        else
            ConB = 0;
        end
    else
        ConB = Cons(x,y);
        if Life(x,1) == +inf
            ConA = Cons(x,1);
        else
            ConA = 0;
        end
    end
    Life(x,y) = +inf;
    AllNodes(x) = Node(ConA,ConB);
```

```
    Q0 = sum(AllNodes(:)==0);   Q1 = sum(AllNodes(:)==1);
    Q2 = sum(AllNodes(:)==2);   Q3 = sum(AllNodes(:)==3);
    Q4 = sum(AllNodes(:)==4);   Q5 = sum(AllNodes(:)==5);
    C1 = (Q5 >= 1);
    C2 = (Q3 >= 2);
    C3 = (Q0 + Q3 + Q2 == 0);
    C4 = ((Q0 + Q1 +((Q3 + Q2) > 0)) < 5);
    C5 = (Q5 == 0);
    C6 = (Q3 == 1 && Q0 + Q1 >= 4);
    C71 = (Q3 == 0 && Q0 >= 1 && Q0 + Q1 >= 5);
    C72 = (Q3 == 0 && Q0 == 0 && Q1 >= 4 && Q2 >= 1);
    C7 = (C71 || C72);
    C8 = (Q5 + Q3 == 0);
    C9 = (Q0 >= 1 && Q0 + Q1 == 4 && Q2 >= 1);
    if C1 || C2 || C3 || C4
        ConSys = 1;
    elseif C5 && (C6 || C7)
        ConSys = 2;
    elseif C8 && C9
        P = Q2 / (Q2 + Q0);
        if rand <= P
            ConSys = 3;
        else
            ConSys = 4;
        end
    end
        if ConSys == 1 || ConSys == 4
        ExpectLife = minimum;
    end
end
end
```

---

```
function main()
for n = 5 : 20
    Val = 0;
```

```
    ExpectLife = zeros(100000,1);
    for i = 1 : 100000
        [ Life , Cons ] = Record(n);
        ExpectLife(i) = System( n , Life , Cons );
        if ExpectLife(i) >= 25000
            Val = Val + 1;
        end
    end
    ExpectLife = mean(ExpectLife);
    Reliablity = Val / 100000;
    fprintf('When the number of node is %d, MTTF is %5.2f, reliability
is %5.4f\n',n , ExpectLife , Reliablity);
end
end
```

---

## KeYongXing.m

---

```
PA0 = exp(-25000 / 37000);  PB0 = exp(-25000 / 480000);
PA1 = 0.26 * (1 - PA0);     PA2 = 0.26 * (1 - PA0);
PA3 = 0.48 * (1 - PA0);     PB1 = 0.35 * (1 - PB0);
PB2 = 0.65 * (1 - PB0);
PBase = PB0;
PB1 = 0.35 * 0.35 * (1 - PBase)^2;
PB2 = 2 * 0.65 * (1 - PBase) - 0.65 * 0.65 * (1 - PBase)^2;
PB0 = 1 - PB1 - PB2;

PN1 = PA0 * PB0;     PN2 = (PA0 + PA2) * PB1;
PN3 = PA0 * PB2 + PA1 * PB0 + PA1 * PB2;    PN4 = PA1 * PB1;
PN5 = PA2 * PB0;     PN6 = PA2 * PB2 + PA3;

for n = 5 : 20
    for k1 = 0 : n
```

```
       for k2 = 0 : (n - k1)
           for k3 = 0 : (n - k1 - k2)
               for k4 = 0 : (n - k1 - k2 - k3)
                   for k5 = 0 : (n - k1 - k2 - k3 - k4)
                       k6 = n - k1 - k2 - k3 - k4 - k5;
                       if (k4 >= 1)||(k2 >= 2)||(k1 + k2 + k5 == 0)||(k1 +
k3 + sum(k2 + k5 > 0) < 5)
                           continue;
                       end
                       if (k2 == 1 && k1 + k3 >= 4)||(k2 == 0 && k1 >= 1 &&
k1 + k3 >= 5)||(k2 == 0 && k1 == 0 && k5 >= 1 && k3 >= 4)
KeYongXing(n) = KeYongXing(n) + nchoosek(n, k1) * nchoosek(n-k1, k2) *
nchoosek(n-k1-k2, k3) *nchoosek(n-k1-k2-k3, k4) * nchoosek(n-k1-k2-k3-k4,
k5) * nchoosek(n-k1-k2-k3-k4-k5, k6) * PN1^(k1) * PN2^(k2) * PN3^(k3) *
PN4^(k4) * PN5^(k5) * PN6^(k6);
                       else
KeYongXing(n) = KeYongXing(n) + nchoosek(n, k1) * nchoosek(n-k1, k2) *
nchoosek(n-k1-k2, k3) *nchoosek(n-k1-k2-k3, k4) * nchoosek(n-k1-k2-k3-k4,
k5) * nchoosek(n-k1-k2-k3-k4-k5, k6) * PN1^(k1) * PN2^(k2) * PN3^(k3) *
PN4^(k4) * PN5^(k5) * PN6^(k6) * k5 / (k5 + k1);
                       end
                   end
               end
           end
       end
   end
end
```

---

# BusBlockingFigure.m

---

```
hold on; ylabel('Prob', 'fontsize', 12);
A= [0.0373    0.0734    0.1124    0.1533    0.2022    0.2467    0.2997
0.3584    0.4169    0.4868    0.5517    0.6194    0.6860    0.7476
```

```
0.8000    0.8456];
B= [0.0007    0.0020    0.0037    0.0062    0.0092    0.0127    0.0167
0.0225     0.0284   0.0362     0.0471     0.0596     0.0787     0.0999
0.1274    0.1625];
plot(5:20, A, '-', 'markersize', 9, 'Linewidth', 2);
plot(5:20, B, '-', 'markersize', 9, 'Linewidth', 2);
hold off; grid on; box on;
xlabel('Node Numbers', 'fontsize', 12);
title('Fatal Problems Triggered By Bus Blocking', 'fontsize', 14);
```

---

## LifeandConditionFigure.m

---

```
hold on;
yyaxis left; ylabel('工作寿命/小时', 'fontsize', 12);
plot(5 : 20, ExpectLife(5 : 20), '.-', 'Linewidth', 2);

yyaxis right; ylabel('系统可靠性', 'fontsize', 12);
plot(5 : 20, Reliablity(5 : 20), '.-', 'Linewidth', 2);

hold off;
legend('工作寿命', '系统可靠性', 'location', 'southeast');
xlabel('节点数/个', 'fontsize', 12);
title('改进后的工作寿命与系统可靠性', 'fontsize', 14);
```

---

**备 注:**

含有下划线的代码为第 5 节中优化系统时所用