

**软件可靠性课程实验报告**

**题 目:** J-M软件可靠性模型

**院 系:** 计算机科学与技术学院

**专 业:** 软件工程

**学生姓名:** 贺银香

**学 号:** 061710401

**二零一九 年 9 月 26日**

目录

[1.引言 3](#_Toc529726265)

[1．1编写目的 3](#_Toc529726266)

[2.模型理论 3](#_Toc529726267)

[2.1 模型背景 3](#_Toc529726268)

[2.2 模型假设 3](#_Toc529726269)

[2.3 模型推导 4](#_Toc529726270)

[3.算法实现 6](#_Toc529726271)

[3.1 流程图 6](#_Toc529726272)

[3.2 算法伪代码 7](#_Toc529726273)

[3.3 算法实现 8](#_Toc529726274)

[3.5 数据来源 11](#_Toc529726275)

[3.6 结果展示 11](#_Toc529726276)

[4.总结 12](#_Toc529726277)

[参考文献 12](#_Toc529726278)

# 1.引言

## 1．1编写目的

随着软件规模越来越大，结构日趋复杂，应用日趋广泛。软件危机依然是我们难以逾越的障碍，软件质量问题越来越突出。软件可靠性问题占装备可靠性问题的比例越来越高，软件可靠性问题呈逐年上升趋势。

因为软件可靠性问题造成的产品故障甚至事故或灾难屡见不鲜，因此重视软件可靠性工程的研究并付诸于实践，是改进软件过程，提高软件可靠性的根本出路。加强软件可靠性工程的研究和实践，势在必行，势在必然。

此次试验以JM模型为核心，帮助我们理解什么是软件可靠性评估及其基本原理，掌握和应用JM模型。

# 2.模型理论

## 2.1 模型背景

该模型是由Jelinski、Moranda于1972年开发的可靠性模型，是最早建立的软件可靠性模型之一，曾用于麦克唐奈道格拉斯海军工程中。

该模型以一种简便和合乎直觉的方式表明如何根据软件缺陷的显露历程来预计未来软件可靠性的行为，它包含软件可靠性建模中若干典型和最主要的假设，因而引发出后来的许多种变形。

事实上，现有大多数软件可靠性模型要么可认为是其变形或扩展，要么与其密切相关。该模型对软件可靠性定量分析技术的建立和发展做出了重要的贡献，是软件可靠性研究领域的第一个里程碑。

## 2.2 模型假设

* 程序中的固有错误数N0是一个未知的常数
* 程序中的各个错误是相互独立的，每个错误导致系统发生失效的可能性大致相同，各次失效间隔时间也相互独立
* 测试过程中检测到的错误，都被排除，每次排错只排除一个错误，排错时间可以忽略不计，在排错过程中不引入新的错误
* 程序的失效率在每个失效间隔时间内是常数，其数值正比于程序中残留的错误数，在第i个测试区间，其失效率函数为:
* 错误以相等的可能发生，且相互独立，错误检测率正比于当前程序中的错误数
* 软件的运行方式与预期的运行方式相似

## 2.3 模型推导

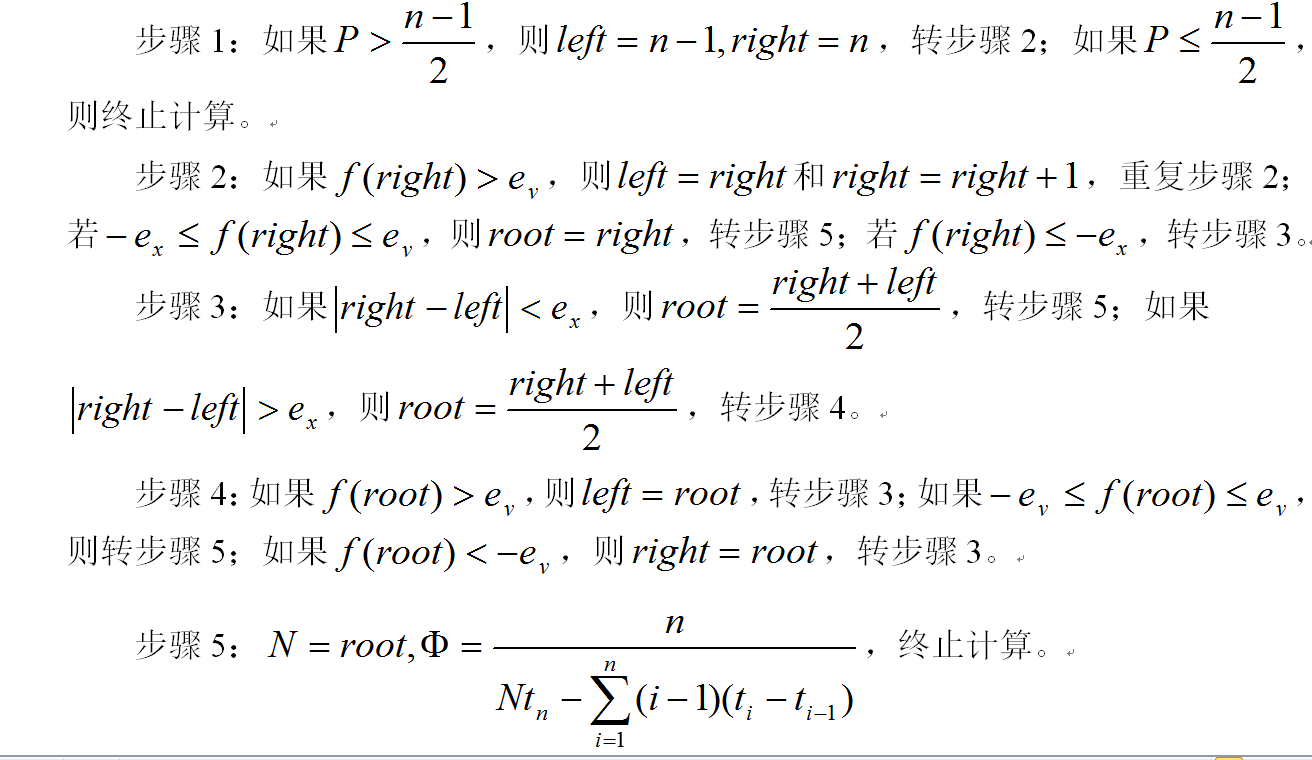
JM模型给出的失效率函数为：其中Φ和N0是未知参数，Φ是比例常数，N0是固有错误数，也就是我们要进行估计的参数，以第i-1次失效为起点的第i次失效发生的时间是一个随机变量，它服从以Z(Xi)为参数的指数分布，其概率密度函数为

我们用最大似然法对上述两个未知参数进行估计，最终得到的估计值是从下列两个方程中求解的：

由上述两个方程我们得知在编程实现中，我们就需要计算如下两个值：

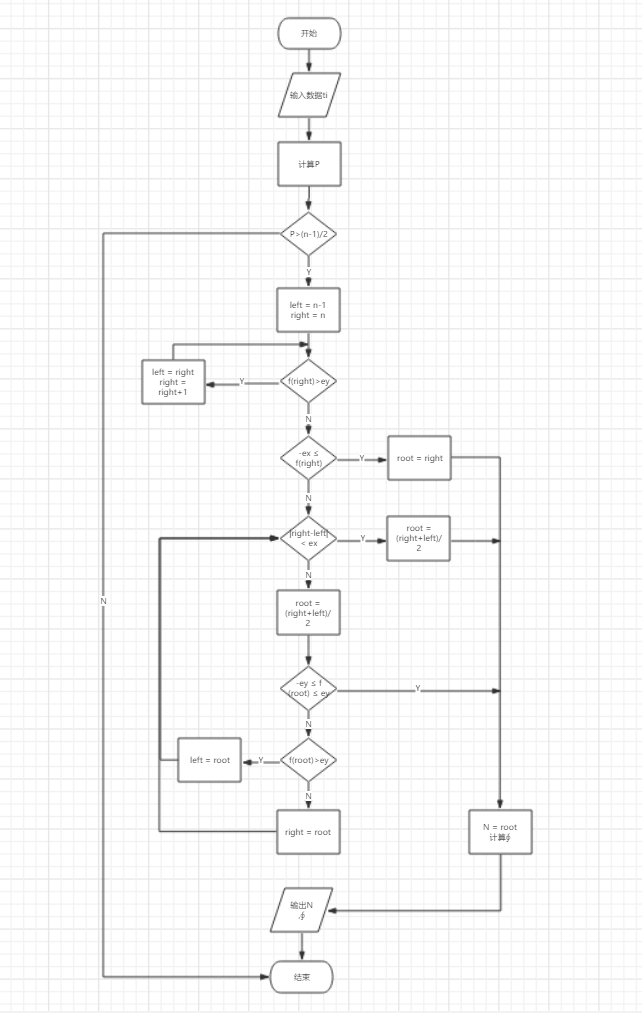
同时这里设为对自变量x的误差精度控制值，为对自变量y的误差精度控制值。

接下来根据如下算法步骤进行求解N和Φ：



# 3.算法实现

## 3.1 流程图

****

## 3.2 算法伪代码

本程序最重要的就是五个步骤的执行，其伪代码如下：

//步骤1

if(P > (n-1)/2){

left = n-1;

right = n;

}

else return;

//步骤2

while (f(right) > ey){

left = right;

right = right + 1;

}

if(-ex <= f(right) && f(right) <= ey){

root = right;

N = root;

return;

}

else {

while (true) {

//步骤3

if (|right-left| < ex) {

root = (right+left)/2;

N = root;

return;

}

else {

root = (right+left)/2;

}

//步骤4

if (-ey <= f(root) && f(root) <= ey) {

N = root;

return;

}

else if (f(root) > ey) {

left = root;

}

else {

right = root;

}

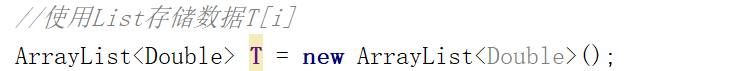
}

}

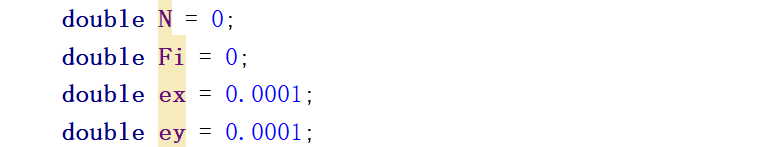
3.3 算法实现

本次实验采用Java语言编写，运行的IDE是IntelliJ IDEA。

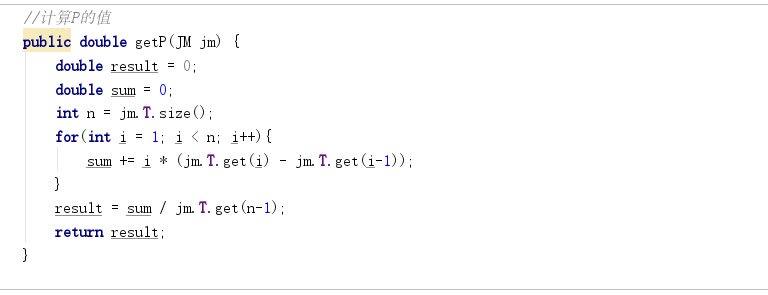
数据结构：

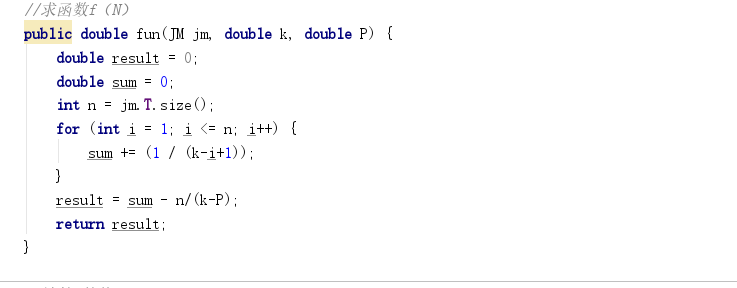


数据的初始化：

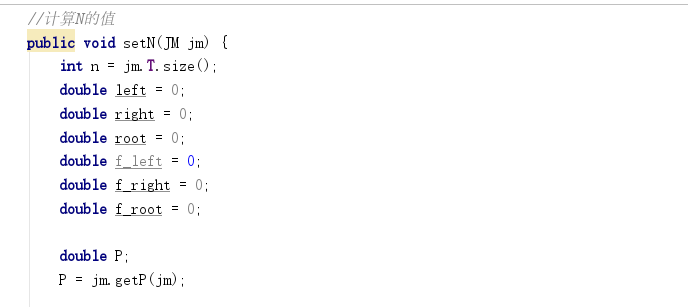


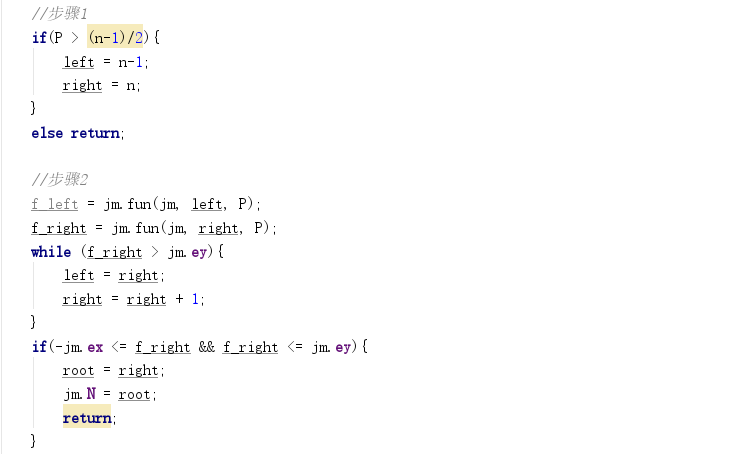
实现的关键代码如图所示：

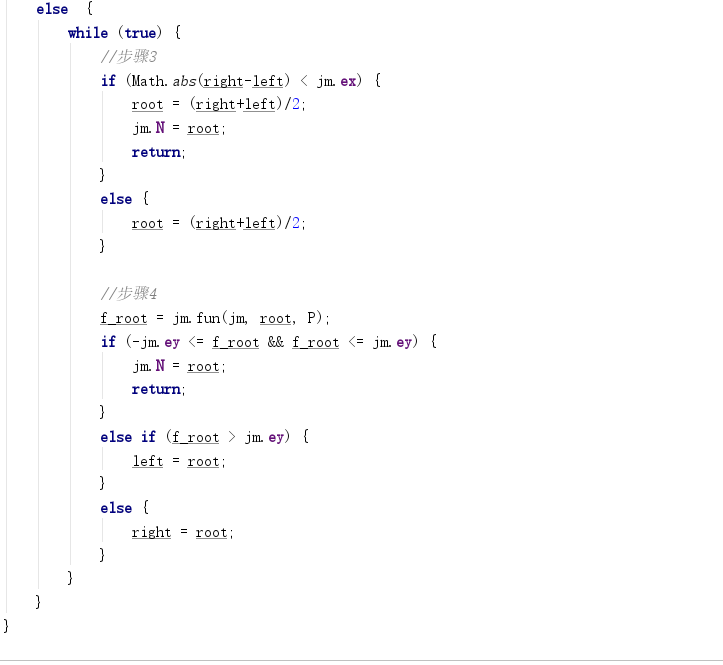


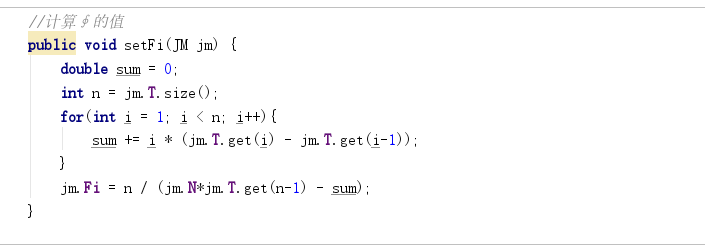










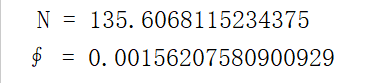


## 3.5 数据来源

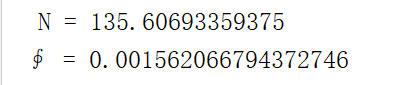
数据来源是老师提供的失效错误数据SYS1(failue\_count).txt，我已放入了java工程目录下。

## 3.6 结果展示

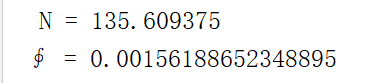
时：



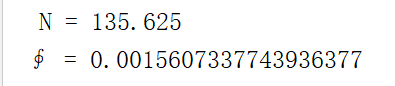
时：



时：



时：



# 4.总结

通过改变和的值，我们会得到不同的输出结果，同时能够理解自变量误差精度控制对参数估计的影响。由实验结果我们可以看出，ex和ey对结果是有一定影响的，但影响随着ex、ey的增长越来越小，这不难理解，因为ex和ey的意义是控制精度，必然不能很大，如果很大了，则说明误差越大，那么得到的结果毫无意义，算法也规避了这种情况，具体体现在了步骤2中。

由此看来JM模型在评估开发结束时软件的可靠性水平MTBF或失效率还是十分可靠的。

通过这次实验，我对于JM模型有了一定的了解，与此同时也对于软件可靠性有了更深刻的理解。

# 参考文献

[1] (美)[M.R.柳]Michael R.Lyu主编; 刘喜成,钟婉懿等译.软件可靠性工程手册（Handbook Of Software Reliabilty Engineering）[M].电子工业出版社，1997