MARCOS ANTONIO COQUE JUNIOR

# Modelo de confiabilidade para sistemas reparáveis considerando diferentes condições de manutenção preventiva imperfeita

São Paulo
2016

MARCOS ANTONIO COQUE JUNIOR

# Modelo de confiabilidade para sistemas reparáveis considerando diferentes condições de manutenção preventiva imperfeita

Tese apresentado à Escola Politécnica da Universidade de São Paulo como requisito para obtenção do Título de Doutor em Ciências.

São Paulo
2016

MARCOS ANTONIO COQUE JUNIOR

# Modelo de confiabilidade para sistemas reparáveis considerando diferentes condições de manutenção preventiva imperfeita

Tese apresentado à Escola Politécnica da Universidade de São Paulo como requisito para obtenção do Título de Doutor em Ciências.

Área de Concentração:
Engenharia Mecânica de Projeto de Fabricação

Orientador:
Prof. Dr. Gilberto Francisco Martha de Souza

São Paulo
2016

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, _____ de _____ de _____

Assinatura do autor: _____

Assinatura do orientador: _____

Catalogação-na-publicação

COQUE JR., MARCOS A.
Modelo de confiabilidade para sistemas reparáveis considerando diferentes condições de manutenção preventiva imperfeita / M. A. COQUE JR. -- versão corr. -- São Paulo, 2016.
184 p.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecânica.

1.TEORIA DA CONFIABILIDADE 2.ESTATÍSTICA 3.MANUTENÇÃO PREVENTIVA I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecânica II.t.

Dedico este trabalho a minha amada esposa Fabiana
e ao meu querido filho Matheus.

AGRADECIMENTOS

Agradeço a Deus por possibilitar eu vivenciar e experimentar tudo o que vivi.

Agradeço a minha esposa Fabiana por todo seu carinho e apoio em todas as ocasiões para realização deste projeto.

Agradeço ao meu filho Matheus por seus sorrisos que me davam força para prosseguir.

Agradeço ao professor Gilberto F. M. Souza pela confiança quando me aceitou como orientando e por todo apoio durante a execução do doutorado.

Agradeço a Universidade de São Paulo por fornecer toda estrutura para minha formação acadêmica, da graduação ao doutorado.

A simplicidade é o último grau de sofisticação.

(Leonardo da Vinci)

# RESUMO

Um sistema reparável opera sob uma estratégia de manutenção que exige ações de recuperação preventiva em tempos pré-definidos e ações de reparo quando ocorre a perda de função do sistema. A manutenção preventiva (MP) é programada periodicamente e muitas vezes possui um intervalo de tempo fixo para ações. No entanto, as atividades de MP podem não restaurar o sistema para uma condição similar ao início de vida deste, mas para uma situação intermediária. Nesse caso, a MP é denominada de imperfeita. Além disso, ao longo da vida do sistema, são executados diferentes planos de manutenção com condições e atividades distintas que podem afetar a intensidade de falha de diferentes maneiras. Para modelar essas características da MP em um sistema reparável, propõe-se uma nova classe de modelo de fator de melhoria, denominado fator de melhoria variável que possibilita a modelagem da situação de manutenção perfeita. A formulação da função de verossimilhança foi desenvolvida para estimação dos parâmetros bem como desenvolvidos testes de verificação da qualidade de ajuste, intervalos de confiança para os parâmetros e otimização da periodicidade de realização da MP com base no enfoque dos novos modelos propostos. Os resultados foram aplicados em dados reais e verificou-se uma parametrização mais flexível a MP imperfeita e maior versatilidade nas análises de confiabilidade do sistema quando utilizado os novos modelos.

Palavras-chave: Sistemas reparáveis. Manutenção preventiva imperfeita. Processo de Poisson não homogêneo. Lei da potência. Fator de melhoria. Função de verossimilhança.

**ABSTRACT**

A repairable system operates under a maintenance strategy that calls for preventive repair actions at prescheduled times and the repair actions that restore system when failure occurs. The preventive maintenance (PM) is scheduled periodically and it often holds a fixed time interval for PM actions. However, PM activities are generally imperfect and cannot restore the system to as good as new condition but to an intermediate situation, which is called imperfect PM. In addition, throughout system life are implemented diverse maintenance policies with different activities and conditions that may affect the failure intensity in different ways. To model these PM characteristics, proposes a new model class of improvement factor called variable improvement factor that also enables modeling perfect maintenance situation. The likelihood function is developed for parameter estimation as well as goodness-of-fit tests and confidence intervals for the parameters are developed, and optimization of the PM intervals based on the proposed models is presented. The proposed model was applied to a data set and a more flexible parameterization for imperfect PM and greater versatility in the system reliability analysis were verified with the use of the new model.

Keywords: Repairable system. Imperfect preventive maintenance. Nonhomogeneous Poisson process. Power law Process. Improvement factor. Likelihood function.

## LISTA DE ABREVIATURA E SIGLAS

| | |
|---|---|
| **BSC** | Balanced Scored Card |
| **CVM** | Cramér Von Mises |
| **EMV** | Estimador de Máxima Verossimilhança |
| **FVM** | Função Valor Médio |
| **LP** | Lei da Potência |
| **MC** | Manutenção Corretiva |
| **MP** | Manutenção Preventiva |
| **MPI** | Manutenção Preventiva Imperfeita |
| **MPP** | Manutenção Preventiva Perfeita |
| **PP** | Processo de Poisson |
| **PPNH** | Processo de Poisson Não Homogêneo |
| **PR** | Processo de Renovação |
| **RF** | Redução de Intensidade de Falha |
| **RFE** | Redução de Intensidade de Falha com relação Exponencial |
| **RFL** | Redução de Intensidade de Falha com relação Linear |
| **RFP** | Redução de Intensidade de Falha com relação Potência |
| **RI** | Redução de Idade |
| **RIE** | Redução de Idade com relação Exponencial |
| **RIL** | Redução de Idade com relação Linear |
| **RIP** | Redução de Idade com relação Potência |
| **TRV** | Teste de Razão de Verossimilhança |

**SUMÁRIO**


**1. INTRODUÇÃO** ..... 6
1.1 Definição do problema ..... 8
1.2 Objetivos. ..... 9
1.3 Organização da tese ..... 9
1.4 Revisão bibliográfica ..... 10
**2. CONCEITOS DE SISTEMAS REPARÁVEIS** ..... 15
2.1 Introdução ..... 15
2.2 Modelo probabilístico: o processo de Poisson ..... 17
2.3 Estimação por máxima verossimilhança ..... 19
**3. MODELOS PARA MANUTENÇÃO PREVENTIVA IMPERFEITA** ..... 22
3.1 Modelo de fator de melhoria ..... 23
3.1.1 Modelo de fator de melhoria com método de redução de idade ..... 24
3.1.2 Modelo de fator de melhoria com método de redução da intensidade de falha ..... 27
3.2 Fator de melhoria variável ..... 29
3.2.1 Modelo de redução de idade com fator de melhoria variável ..... 30
3.2.2 Modelo de redução de intensidade de falha com fator de melhoria variável ..... 32
3.2.3 Modelo de *Balanced Scored Card* para severidade da MP ..... 34
**4. ESTIMAÇÃO NOS MODELOS PROPOSTOS** ..... 36
4.1 Definição das funções de verossimilhança ..... 36
4.2 Estimação dos parâmetros ..... 40
4.3 Cálculo dos limites de confiança ..... 41
4.4 Avaliação das propriedades assintóticas dos estimadores dos modelos propostos 42
4.4.1 Processo de simulação ..... 43
4.4.2 Resultados da simulação ..... 45
4.4.3 Conclusões sobre o estudo de simulação ..... 54
**5. MÉTODO PARA AVALIAÇÃO DO AJUSTE DOS MODELOS** ..... 56
5.1 Teste de razão de verossimilhança para comparar modelos ..... 56
5.2 Teste e gráfico para avaliação da qualidade do ajuste do modelo ..... 58
5.2.1 Teste *Cramér Von Mises* ..... 58

5.2.2 Gráfico para avaliação da qualidade do ajuste dos modelos..... 59
5.3 Metodologia para avaliação do melhor modelo..... 61
**6. OTIMIZAÇÃO DA PERIODICIDADE DA MP IMPERFEITA**..... 62
**7. FUNÇÃO CONFIABILIDADE**..... 65
**8. APLICAÇÃO DO MODELO**..... 68
8.1 Estimativa e seleção dos modelos..... 70
8.2 Modelo selecionado: redução de idade com relação exponencial (RIE)..... 73
8.3 Função do fator de melhoria variável..... 75
8.4 Número de falhas considerando o fator de melhoria variável..... 76
8.5 Confiabilidade considerando o fator de melhoria variável..... 77
8.6 Otimização da periodicidade de MP considerando o fator de melhoria variável... 79
**9. CONCLUSÃO**..... 81
**REFERÊNCIAS**..... 83
**APÊNDICE A – programas em R**..... 88
A.1 Programa referente ao modelo de Manutenção Preventiva Perfeita (MPP)..... 88
A.2 Programa referente ao modelo de Redução de Idade (modelo RI)..... 98
A.3 Programa referente ao modelo de Redução de Idade com relação Exponencial (modelo RIE)..... 109
A.4 Programa referente ao modelo de Redução de Idade com relação Linear (modelo RIL) ..... 120
A.5 Programa referente ao modelo de Redução de Idade com relação Potência (modelo RIP) ..... 131
A.6 Programa referente ao modelo de Redução de Intensidade Falha (modelo RF).... 142
A.7 Programa referente ao modelo de Redução de Intensidade Falha com relação Exponencial (modelo RFE) ..... 152
A.8 Programa referente ao modelo de Redução de Intensidade Falha com relação Linear (modelo RFL) ..... 163
A.9 Programa referente ao modelo de Redução de Intensidade Falha com relação Potência (modelo RFP) ..... 174

# 1. **INTRODUÇÃO**

A análise de confiabilidade busca desenvolver modelos que permitam representar probabilisticamente o comportamento de falha de um sistema e elaborar predições que suportem a tomada de decisão. Assim, é possível reduzir o número de eventos indesejáveis e o impacto que estes podem causar financeiramente ao meio ambiente e à vida humana.

Um dos campos de estudo em confiabilidade é a análise de sistemas reparáveis. Um sistema é definido como reparável quando recebe ações de reparo e manutenção para restaurar seus componentes quando estes apresentaram uma falha e este possa ser reconduzido para o estado em que ele está apto a realizar todas as suas funções (RIGDON; BASU, 2000). Essas ações podem afetar o sistema devido às diferentes características de práticas de manutenção existentes. Basicamente, existem duas categorias principais de ações de manutenção: Manutenção Corretiva (MC) e Manutenção Preventiva (MP).

De acordo com o MIL-STD-721C (1981), MC constitui uma ação de manutenção realizada em decorrência de uma falha, para restaurar o sistema a uma condição especificada. MP significa ações realizadas na tentativa de manter um item em uma condição especificada, executando inspeção sistemática, detecção e prevenção de falhas. A MP é uma ação de manutenção que ocorre quando o sistema está em funcionamento, antecipando-se a ocorrência de falhas.

Pode-se considerar que o nível de eficácia ideal da ação de reparo é aquele em que, após a ação de manutenção, o equipamento funciona exatamente conforme sua condição inicial de operação, o que constitui uma manutenção perfeita. Porém, devido a limitações técnicas e econômicas, isso, muitas vezes, não é possível. Geralmente as atividades de manutenção não resultam em uma situação extrema de melhoria do componente, mas em uma condição intermediária. Manutenção imperfeita consiste no conceito sobre o qual ação de manutenção não traz o sistema para uma condição de início de vida, mas para um estado intermediário entre o novo e o estado anterior à falha ou parada preventiva.

Sistemas reparáveis podem ser analisados utilizando modelos de confiabilidade baseados em processos estocásticos. Os modelos probabilísticos utilizados para modelar o comportamento de falha de um sistema são principalmente o Processo de Renovação (PR) e Processos de Poisson Não Homogêneo (PPNH) (MEEKER; ESCOBAR, 1998).

Os sistemas que são modelados por um PR consideram que, após o reparo de uma falha, o sistema retorna para uma condição de equivalente de novo. Seria uma ação de

manutenção perfeita, refletindo uma condição otimista da ação de reparo, o que muitas vezes não é observado na realidade. Os sistemas que normalmente adotam o PR como válido são aqueles que, quando falhos, são imediatamente substituídos por outros novos (sistemas não reparáveis).

Os sistemas modelados por um PPNH representam condição mínima em relação à ação de manutenção. Nestes, o reparo corretivo atua, apenas, para retornar o sistema à operação na condição antes da falha, o que, segundo Rigdon e Basu (2000), significa que o sistema sofreu manutenção mínima ou reparo mínimo.

Tanto o PR quanto o PPNH consideram as possibilidades extremas de ações de manutenção. Quando aplicam-se esses conceitos em sistemas complexos, ou seja, composto de diversos itens, assumir MC mínima parece ser aceitável, visto que o reparo corretivo em alguns componentes promoverá a volta da função do sistema, mas nenhuma melhoria na condição atual. Porém, essas suposições podem não ser observadas quando se trata de MP, uma vez que é possível supor que os procedimentos de MP são elaborados visando à redução do número de falhas e promovendo melhorias no desempenho do sistema, mas não a sua restauração para uma condição de atuação igual ao início de sua operação.

Outro ponto relevante na execução dos planos de MP de um sistema refere-se ao fato de que, geralmente, as atividades executadas em cada ação preventiva são revisadas e alteradas ao longo da vida do sistema. Dessa forma o impacto da MP no desempenho do sistema é diferente para cada ação realizada, promovendo melhorias de distintas magnitudes dependendo da extensão da ação realizada.

A caracterização precisa do comportamento de falha do sistema considerando as diferentes possibilidades de manutenção é fundamental na definição de política de manutenção que possibilite desenvolver estratégias para reduzir a degradação da confiabilidade, prevenir falhas e reduzir custos de manutenção. Dessa forma, faz-se necessário o desenvolvimento de modelos estatísticos que considerem as características da manutenção preventiva no processo de falha do sistema.

Este trabalho tem por objetivo desenvolver métodos estatísticos para modelos que permitam caracterizar o comportamento de falha considerando diferentes condições de manutenção preventiva imperfeita. Utilizando um PPNH, a condição de MP imperfeita será caracterizada pela abordagem de fator de melhoria. Os modelos de fator de melhoria serão representados pelo método de redução da idade e pelo método de redução de intensidade de falha. Uma nova classe de modelos é proposta, a qual utiliza a estrutura do fator de melhoria, porém são substituídas por funções paramétricas para considerar as diferentes condições de

manutenção preventiva ao longo do tempo. São desenvolvidas funções lineares, exponenciais e potenciais para a relação entre condição de execução da manutenção e o efeito da ação de manutenção na intensidade de falha.

Propostos os modelos, serão definidos e implantados computacionalmente método de estimação dos parâmetros dos modelos utilizando a Teoria de Verossimilhança, teste para comparar os modelos, teste de qualidade de ajuste e estimação de intervalos de confiança para os parâmetros. Também será apresentada uma forma de otimização do tempo para execução da manutenção preventiva, considerando-se os modelos desenvolvidos anteriormente.

Para avaliação das propriedades assintóticas dos estimadores dos modelos propostos, será desenvolvido um estudo de simulação.

Em suma, este trabalho visa contribuir com a proposição de uma nova classe de modelos e de um processo de obtenção de estimativas de confiabilidade em sistemas reparáveis considerando o processo de manutenção preventiva imperfeita sob diferentes condições de execução de manutenção preventiva. E, por conseguinte, será aprimorado o método de otimização da periodicidade do tempo de execução da ação preventiva.

## 1.1 **Definição do problema**

Os sistemas reparáveis sofrem diferentes ações de manutenção durante sua vida útil que visam manter seu desempenho acima de um valor mínimo previamente especificado. Porém, a maioria dos modelos estatísticos desenvolvidos para a mensuração de confiabilidade supõem que a manutenção preventiva melhora a condição do sistema para uma condição similar ao início de operação. Na prática, isso pode não ser verdade, o que acarreta erros nas métricas de confiabilidade podendo promover equívocos na estratégia de manutenção. Esse fato pode ocasionar aumento do número de falhas e aumento dos custos de manutenção.

Além disso, em geral, são realizados diferentes planos de manutenção ao longo do período de funcionamento do sistema, ou seja, em cada manutenção, é executado um número de inspeções, substituições e tarefas distintas. As ações de manutenção preventiva podem não ser uniformes durante a vida do equipamento, pois estas sofrem mudanças em decorrência da alteração de política de manutenção da empresa.

Para uma caracterização precisa do comportamento de falha, é interessante desenvolver métodos estatísticos que permitam mensurar o efeito da manutenção imperfeita e as diferentes condições de manutenção ao longo do período de funcionamento do sistema.

Nesse caso, as métricas de confiabilidade serão mais assertivas, permitindo decisões de manutenção mais assertivas, aumento da confiabilidade do sistema e redução de custos.

## 1.2 **Objetivos**

O principal objetivo desta tese consiste em desenvolver métodos estatísticos que permitam considerar a manutenção preventiva imperfeita na avaliação das métricas de confiabilidade, propondo um novo modelo que permita mensurar as diferentes condições de manutenção preventivas realizadas ao longo da vida do sistema.

Para tanto, os seguintes objetivos específicos são enumerados:

- realização de revisão bibliográfica sobre o desenvolvimento de modelos para manutenção imperfeita;
- proposição de novos modelos estatísticos que considerem ação de manutenção preventiva imperfeita e os diferentes planos de manutenção;
- desenvolvimento do método de estimação dos parâmetros envolvidos nos modelos propostos;
- cálculo dos limites de confiança;
- desenvolvimento do procedimento para avaliação da qualidade do ajuste dos modelos propostos;
- avaliação das propriedades assintóticas dos estimadores sugeridos;
- aplicação dos modelos propostos para otimização da periodicidade de realização da manutenção preventiva e cálculo da confiabilidade;
- implantação computacional, na linguagem de programação R, dos pontos supracitados para utilização em análise de dados de sistemas reparáveis.

## 1.3 **Organização da tese**

Esta tese está organizada da seguinte forma: na próxima seção, será apresentada a revisão bibliográfica sobre o desenvolvimento de modelos para manutenção imperfeita. No Capítulo 2, serão apresentados os conceitos de sistemas reparáveis e o processo de modelagem e estimação do modelo sem considerar o conceito de manutenção imperfeita. No Capítulo 3, será apresentado o conceito de modelagem da manutenção imperfeita por meio do fator de melhoria e proposta uma nova classe de modelo que considere uma ação de

manutenção preventiva imperfeita e os diferentes planos de manutenção denominado fator de melhoria variável. No Capítulo 4, estão as funções de verossimilhança para estimação de parâmetros no modelo proposto no capítulo anterior, bem como o cálculo dos limites de confiança para os parâmetros estimados. Ainda no Capitulo 4, será desenvolvido um estudo para avaliação das propriedades assintóticas dos estimadores dos modelos propostos. No Capítulo 5, será apresentada metodologia baseada em testes estatísticos e em análise gráfica para escolha do modelo que melhor se ajuste aos dados utilizados para estimação. O Capítulo 6 apresentará o método de otimização da periodicidade, para realização da ação preventiva, adaptados para os modelos de manutenção imperfeita. O Capítulo 7 apresentará a definição da função confiabilidade para os modelos propostos. Por fim, nos Capítulos 8 e 9, será apresentado um conjunto de dados, a aplicação da teoria descrita ao longo desta tese e serão formuladas conclusões a respeito deste trabalho e apontamentos para trabalhos futuros em relação à linha de pesquisa proposta neste trabalho.

## 1.4 **Revisão bibliográfica**

Sistemas reparáveis caracterizam-se por serem passíveis de alguma ação de manutenção após perderem suas funções (falha) e podem ser analisados utilizando modelos de confiabilidade baseados em processos estocásticos. Um processo estocástico constitui uma coleção de variáveis aleatórias que, por sua vez, pode representar o tempo até a ocorrência de uma falha e algum mecanismo probabilístico que explique o seu comportamento por meio de distribuições de probabilidade (ROSS, 2012). As classes de modelos probabilísticos mais utilizados para modelar o comportamento de falha de um equipamento/sistema reparável são o Processo de Renovação (PR) e Processos de Poisson Não Homogêneo (PPNH) (MEEKER; ESCOBAR, 1998).

Rigdon e Basu (2000) definem PR como um processo pontual no qual os tempos entre falhas são variáveis aleatórias independentes e identicamente distribuídas (i.i.d.) e são representadas por uma distribuição de probabilidade. Esse tipo de processo é utilizado para sistemas que retornam à função, após uma ação de manutenção, com o mesmo desempenho que tinham no início de operação. Essa ação é denominada manutenção perfeita e essa modelagem é mais utilizada para sistemas não reparáveis. PPNH é definido como um processo que assume que o reparo recupera o sistema falho a uma condição de desempenho similar ao imediatamente antes da falha, o que é denominado de reparo mínimo. Assim, as suposições de independência dos tempos de falha e também o fato de que estes sejam

identicamente distribuídos não são sempre válidas (RIGDON; BASU, 2000). Consequentemente, os dados de falha de sistemas reparáveis precisam receber um tratamento estatístico diferenciados dos sistemas não reparáveis. Modelos baseados nessas suposições foram discutidos em Zhao e Xie (1996), Taghipour e Banjevic (2011) e Jiang (2013) entre outros.

Em geral, na prática a ação de manutenção melhora o desempenho de um sistema a uma condição intermediária entre a manutenção perfeita e a manutenção mínima, denominada manutenção imperfeita (PHAM; WANG, 1996), que compreende o efeito intermediário e as situações extremas de ação perfeita e mínima.

Sendo assim, considerar a característica de manutenção adequada é de fundamental importância para que as métricas de confiabilidade representem a realidade. Outro fato importante refere-se à definição do tempo ótimo para a periodicidade de execução das ações preventivas no sistema. A otimização do tempo de MP tem suma importância, pois possibilita aos gestores tomarem melhores decisões de forma a melhorar a confiabilidade, prevenindo falhas e reduzindo os custos de manutenção. A otimização do tempo de MP pode ser feita considerando critérios de custo de manutenção ou critérios de confiabilidade. Malik (1979) sugere utilizar o critério de custos para sistemas de produção e o critério de confiabilidade para sistemas que proveem serviços. Nos trabalhos de Phelps (1981), Park et al. (2000) e Wang (2002) foram apresentados métodos para definição de politica preventiva utilizando o critério de reparo mínimo. Nesta tese aborda-se a otimização da periodicidade da execução da ação preventiva pelo critério de custos de manutenção, similar ao método apresentado em Barlow e Hunter (1960), que, apesar de não considerarem manutenção imperfeita, será de grande importância para o método proposto neste trabalho.

Diversos métodos foram desenvolvidos para tratar a questão da manutenção imperfeita. Em Pham e Wang (1996) e Tanwar et al. (2014) é apresentada uma revisão que reúne diversos métodos de tratamento de reparo imperfeito e esses podem ser divididos em duas categorias:

1) **método (p,q)** – proposto inicialmente por Brown e Proschan (1983);
2) **método Fator de Melhoria** – proposto inicialmente por Malik (1979).

Brown e Proschan (1983) apresentam um modelo que considera o parâmetro p, que é a probabilidade do reparo perfeito e (1-p), ou seja, a probabilidade de reparo mínimo. A crítica a esse modelo refere-se ao fato de que ele assume que todos os reparos ora são perfeitos ora são mínimos, ou seja, esse modelo não consegue representar estágios intermediários de

reparo. Esse modelo é conhecido como Modelo B-P e originou alguns trabalhos como Sheu e Griffith (1992) e Cui et al. (2004).

A outra categoria introduzida por Malik (1979), em um problema de programação de manutenção, foi denominada como fator de melhoria. Ele apresenta a ideia que uma ação de manutenção muda o comportamento da taxa de falha a qual é ajustada por um fator que pode variar e fazer com que a taxa de falha fique igual ao momento da condição inicial de operação do sistema ou na condição similar ao momento da ação manutenção. O grau de alteração na taxa de falha é denominado fator de melhoria.

As parametrizações do modelo de fator de melhoria consideram duas possibilidades. A primeira considera um parâmetro para reduzir a idade do sistema após a ação de manutenção, a qual foi apresentada, inicialmente, por Kijima et al. (1988) e Kijima (1989), e foi denominada idade virtual do sistema. A segunda parametrização foi introduzida por Nakagawa (1988) e utiliza um parâmetro para reduzir a taxa de falha do equipamento. Lin et al. (2000), Zhou et al. (2007) e Wu e Zuo (2010) apresentam essas parametrizações para modelar o comportamento de falhas e otimização do tempo de manutenção, considerando manutenção imperfeita, mas não abordam a questão da estimação dos parâmetros. Nakagawa (1988) e Nakagawa e Zhao (2011) apresentam modelo baseado na redução de idade e redução da taxa de falha, porém também não fazem menção ao método para estimação.

Shin et al. (1996) e Jack (1998) abordam a questão da estimação do fator de melhoria utilizando a teoria de verossimilhança e otimização de preventiva considerando o modelo de redução de idade do sistema, bem como propõe um método para a construção de intervalos de confiança para os parâmetros. Outros trabalhos abordam a questão do reparo imperfeito como Yanez et al. (2002), Ramírez e Utne (2013), Pandey et al. (2013) e Yevkin e Krivtsov (2013).

Doyen e Gaudoin (2004) propõem duas novas classes de modelos de reparo imperfeitos baseados no tempo e intensidade de falha que consideram um efeito de memória do reparo no comportamento de falha. Eles são denominados Redução da Aritmética de Idade (modelo ARA do inglês *Arithmetic Reduction of Age*) cuja eficiência de reparo é expressa por uma redução na idade do sistema, e Redução Aritmética de Intensidade (modelo ARI do inglês *Arithmetic Reduction of Intensity*) em que a eficiência de reparo é caracterizada pela redução da função intensidade de falha do processo. Os modelos ARA e ARI consideram a possibilidade de incluir um efeito de memória das ações no modelo, porém são mais aplicáveis as manutenções corretivas pois necessitam de muitos ciclos de reparo para o efeito de memoria ser significativo na análise. Doyen (2010), Doyen e Gaudoin (2011) e Corset e Doyen (2012) abordam questões de eficiência sobre a estimação dos seus modelos ARA e

ARI pelo método de verossimilhança e inferência bayesiana, incluindo o cálculo de intervalos de confiança, mas não os utilizam para otimização do tempo de manutenção preventiva.

Alguns dos trabalhos supracitados que abordam a otimização do tempo de MP, consideram a suposição de que os parâmetros do modelo são conhecidos ou estimados empiricamente. Isso pode ser explicado, principalmente, em virtude da complexidade da função de verossimilhança e até mesmo em virtude das soluções numéricas que podem ser extremamente difíceis de obter. Também não tratam da questão de verificação de ajuste do modelo, tampouco abordam os testes de comparação de modelos.

Na prática, esses parâmetros têm que ser estimados com base em dados. Toledo (2014) trata da questão de estimação dos modelos ARA e ARI para o caso de manutenção corretiva imperfeita, incluindo o cálculo de intervalos de confiança e otimização de preventiva, porém considerando esta última como manutenção perfeita.

Os trabalhos analisados consideram que os efeitos dos fatores de melhoria na manutenção possam ser interpretados como o efeito médio durante o período de observação. Não consideram o fato que as manutenções possam ser realizadas sob diferentes condições de severidade e tratam os fatores como uniformes durante os ciclos de manutenções. Jaturonnatee et al. (2006) referem-se à ideia de que é difícil imaginar que as ações preventivas possam ser controladas e possuem efeitos similares ao longo da vida do sistema, isso em decorrência do ambiente e das atividades executadas durante a sua realização.

Este trabalho visa complementar a lacuna referente a métodos de inferência para modelos de MP imperfeita em condições de manutenção distintas, apresentando a metodologia para a estimação de parâmetros de modelos na categoria de PPNH, utilizando o conceito de fator de melhoria para modelar a manutenção preventiva imperfeita e propondo uma nova classe de modelos que utiliza uma função para os fatores de melhoria.

Outro ponto relevante pouco abordado na literatura dos modelos de manutenção imperfeita é a questão de testes estatísticos para verificação da qualidade do ajuste dos modelos e comparação de ajustes entre modelos. Dessa forma, serão adaptados testes estatísticos e análises gráficas para os modelos apresentados. Esses testes são importantes, pois sua execução permitirá que se verifique qual o melhor modelo entre os propostos e se o modelo, considerando manutenção imperfeita, é melhor do que o modelo tradicional que considera a manutenção perfeita. Serão, também, desenvolvidos intervalos de confiança para os parâmetros a fim de verificar a variabilidade estatística dos valores obtidos.

Como não há referência completa sobre os métodos estatísticos de inferência para os modelos de manutenção imperfeita, talvez, por isso, não haja nenhum *software* comercial que

permita a estimação dos seus parâmetros para esses modelos e a otimização dos tempos de manutenção. Desse modo, fará parte deste trabalho o desenvolvimento e implantação em linguagem de programação R (www.R-project.org, v.3.1.2) o algoritmo para estimação proposta, bem como o método de otimização desenvolvido.

Com base nos principais estudos referentes à MP imperfeita considerando fatores de melhoria e otimização de tempos de MP utilizando custos de manutenção, foi desenvolvida a Tabela 1.1 com os assuntos abordados nesses estudos e o que este trabalho, denominado Coque, desenvolverá para complementar a lacuna de uma visão integrada de métodos estatísticos para os modelos de manutenção imperfeita e otimização.

Tabela 1.1 – métodos abordados nas referências analisadas sobre manutenção imperfeita

| Referência | Manutenção imperfeita | | | Estima parâmetros | Teste para avaliar o ajuste | Intervalos de confiança | Otimização do tempo de MP |
|---|---|---|---|---|---|---|---|
| | Redução da idade | Redução da intesidade de falha | Diferentes condições de MP | | | | |
| Corset e Doyen (2012) | X | | | X | | | |
| Doyen e Gaudoin (2004) | X | X | | X | | | |
| Doyen (2010) | X | X | | X | | X | |
| Doyen e Gaudoin (2011) | X | | | X | | | |
| Jack (1998) | X | | | X | | X | X |
| Lin et al. (2000) | X | X | | | | | X |
| Malik (1979) | X | | | | | | X |
| Nakagawa (1988) | X | X | | | | | X |
| Nakagawa e Zhao (2011) | X | | | | | | X |
| Ramirez e Utne (2013) | X | | | X | | X | |
| Shin et al. (1996) | X | | | X | | X | |
| Toledo (2014) | X | X | | X | | X | X |
| Wu e Zuo (2010) | X | X | | | | | X |
| Zhou et al. (2007) | X | X | | | | | X |
| Coque | X | X | X | X | X | X | X |

Fonte: elaborada pelo autor

# 2. **CONCEITOS DE SISTEMAS REPARÁVEIS**

## 2.1 **Introdução**

Este capítulo aborda as questões fundamentais ao entendimento dos modelos probabilísticos para caracterizar o comportamento da ação de manutenção imperfeita. São introduzidos conceitos relacionados à análise de sistemas reparáveis e aos modelos probabilísticos para análise de dados de falha desses sistemas.

O termo “sistema” é utilizado neste trabalho como uma classificação que pode compreender diversos níveis e subníveis, seja equipamento, item ou componente. A sua utilização dependerá do contexto em que está inserido ou buscará, apenas, demonstrar uma forma de generalização da aplicação.

Sistema Reparável (ASCHER; FEINGOLD, 1984) é um sistema que, após falhar em realizar pelo menos uma de suas funções, pode ser reconduzido (reparado) para o estado em que estava apto a realizar todas as suas funções anteriores utilizando um procedimento que não seja a substituição total (completa) deste. Portanto sistema reparável é um sistema que pode retornar à condição de operação após uma ação de manutenção. Por exemplo, um automóvel constitui um sistema reparável, pois a maioria das falhas pode ser corrigida por meio do reparo de algum componente, sem a necessidade de troca do veículo.

Em geral, as aplicações para sistemas reparáveis são motivadas em sistemas mecânicos, mas também possibilitam aplicações em outras áreas, tais como indústrias químicas, farmacêuticas ou qualquer outra área em que se possam identificar eventos recorrentes.

O objetivo da manutenção é o de manter ou restaurar um sistema em um estado no qual este possa executar sua função requerida, buscando, sempre que possível, contínuo e progressivo melhoramento em suas ações. Outro objetivo é o de controlar o processo de deterioração que leva o sistema a um estado falho. Para tanto, combina técnicas e ações administrativas, incluindo ações de supervisão e controle conforme descritas em Moubray (1997).

As ações de manutenção podem ser classificadas de diferentes formas. Algumas das mais comuns são apresentadas nos parágrafos seguintes.

Manutenção Corretiva (MC): também denominada de reparo, ocorre após a confirmação da falha. Seu objetivo consiste em trazer o sistema falho à operação o mais breve possível e a uma condição suficiente que o mantenha em operação até que ocorra uma

intervenção mais adequada. Esse reparo pode ser direto sobre a causa da falha, substituindo o item falho ou utilizando redundâncias.

Manutenção Preventiva (MP): trata-se da manutenção efetuada quando um sistema encontra-se em um nível de degradação, em que a probabilidade de ocorrência da falha passa a representar um aspecto impactante na disponibilidade do sistema, incorrendo em maiores custos, impactos ao meio ambiente ou até em vidas humanas.

O objetivo da MP é reduzir a probabilidade de falha do sistema, trazendo-o a uma condição melhor do que estava antes da intervenção e, assim, garantir que o sistema opere em nível adequado de desempenho, durante um tempo de vida requerido. Pode ser efetuada de diversas formas, tais como inspeções, ajustes, lubrificação, troca de peças, calibração e reparo de itens que apresentem sinais que irão falhar.

Cada ação de manutenção pode ser classificada em termos do grau de desempenho do sistema da seguinte forma (PHAM; WANG, 1996):

– **Manutenção Perfeita:** uma ação de manutenção que restaura a condição de operação do sistema para a condição similar a inicial de operação, ou como denominado na literatura “tão bom como novo”.

– **Manutenção Mínima:** uma ação de manutenção que restaura o estado do componente para uma condição operacional, pouco antes da falha.

– **Manutenção Imperfeita:** a ação de manutenção restaura o estado de funcionamento do sistema para a condição intermediaria da manutenção mínima e da manutenção perfeita.

Uma das formas de análise de sistemas reparáveis ocorre empregando processos estocásticos pontuais. Uma classe de processos estocásticos bem desenvolvidos para análise de confiabilidade de sistemas reparáveis são os modelos de Processo de Poisson.

Uma grande parte dos métodos probabilísticos de tratamento de ações de manutenção possuem limitações com relação ao impacto do reparo no desempenho do sistema, ou seja, consideram que são renovados (manutenção perfeita) ou sofrem reparo mínimo. Não consideram os estágios intermediários de desempenho do sistema.

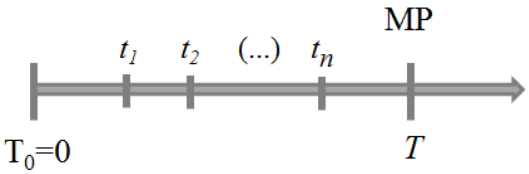
Uma alternativa aos modelos tradicionais é o modelo baseado na utilização do fator de melhoria na intensidade de falha. Esse modelo será exposto em maior detalhe no Capítulo 3, bem como a proposição de um novo modelo para considerar diferentes condições de manutenção imperfeita.

## 2.2 **Modelo probabilístico: o processo de Poisson**

O processo de Poisson (PP) é uma classe de processos estocástico útil para modelar o comportamento de falha de sistemas reparáveis.

Considere um sistema reparável que entre em operação no tempo 0 e a ação de manutenção preventiva (MP) ocorre em tempo pré-definido *T*. Esse sistema é sujeito à manutenção corretiva (MC) com reparo mínimo (a ação restaura o estado da máquina para a sua condição pouco antes da falha), e os *n* tempos de falhas sucessivas observadas são denotados por $t_1, t_2, ..., t_n$ ($0 < t_1 < t_2 < ... < t_n$).

Figura 2.1 – distribuição dos tempos até a falha *t* e o tempo T de MP para um sistema




Fonte: elaborada pelo autor

Um pressuposto importante para utilização desse modelo em análise de sistemas reparáveis refere-se ao fato de que o tempo para executar as ações de manutenção é assumido como sendo desprezível em relação ao tempo total de vida útil, de modo que as falhas podem ser vistas como processos pontuais. PP são especialmente úteis para descrever os processos de falha que possuem certas tendências, como o crescimento ou deterioração de confiabilidade (MEEKER; ESCOBAR, 1998).

O Processo de Poisson é uma contagem do evento, que pode ser definida como um conjunto de variáveis aleatórias independentes *N(t)*, com $t \geq 0$, em que *N(t)* é o número de eventos no intervalo (0, *t*]. Esse processo é caracterizado pela função intensidade de falha definida como,

$$u(t) = \lim_{\Delta t \to 0} \frac{P[N(t+\Delta t) - N(t) = 1 | \mathcal{H}_t]}{\Delta t} \tag{2.1}$$

em que $\mathcal{H}_t$ é o passado do processo de falha no tempo *t*, isto é, o conjunto de todos os eventos ocorridos antes de *t*. A função intensidade de falha pode ser interpretada como a taxa instantânea de mudança no número esperado de falhas.

Quando duas ou mais falhas não ocorrem no mesmo instante de tempo e sob ação de reparo corretivo mínimo, a função intensidade de falha também é conhecida como ROCOF (*Rate of Occurrence of Failures*), cuja demonstração pode ser vista em Rigdon e Basu (2000).

Com a função de intensidade de todo o período de operação, é possível analisar a vida do equipamento em diversas fases, que são frequentemente citadas na literatura por meio da chamada "curva da banheira", que descreve, graficamente, o comportamento da taxa de falha ou intensidade de falha com o tempo, para maiores detalhes veja Rigdon e Basu (2000).

Os sistemas podem se deteriorar, melhorar ou manter-se nas mesmas condições, a frequência de falhas pode ser crescente, decrescente ou constante no tempo. Nesses casos, a frequência de taxa de falha é variável no tempo com intensidades distintas. Assim o PP pode ser dividido em duas classes: Processo de Poisson Homogêneo e o Processo de Poisson Não Homogêneo.

A primeira classe é o Processo de Poisson Homogêneo (PPH) é um processo de Poisson com função intensidade de falha constante no tempo. Esse é o modelo mais simples para sistemas reparáveis, sendo a função de intensidade constante, o PPH não pode ser usado para modelar sistemas que deterioram ou melhoram ao longo do tempo. A classe mais geral do PP refere-se ao momento em que a função de intensidade é dependente do tempo e denominando-se como Processo de Poisson Não Homogêneo (PPNH). Mais informações sobre o processo de Poisson no contexto do sistema reparável pode ser obtido em Rigdon e Basu (2000).

Um processo de contagem de eventos modelado por um PPNH pode ser especificado pelo número médio de eventos entre (0, $t$] que é dado pela função valor médio $\mu(t) = E[N(t)] = \int_0^t u(x)dx$. A função de valor médio descreve o número acumulado de falhas esperada até o tempo $t$.

Uma das formas paramétricas mais utilizadas para caracterizar a função intensidade de falha é a Lei de Potência (LP), quando se assume que a intensidade de falha cresce proporcional a uma potência no tempo conforme função 2.2 (CROW, 1974).

$$u(t) = \frac{1}{\lambda^{\beta}}.\beta.t^{(\beta-1)} \tag{2.2}$$

em que, $\lambda$ (Lambda), $\beta$ (Beta) são parâmetros do modelo ($\lambda$ >0, $\beta$ > 0).

Isso implica que: $\mu(t) = \frac{1}{\lambda^{\beta}}.t^{\beta}$

A Lei de Potência tem grande aplicação em modelos de análise de confiabilidade de sistemas reparáveis devido a sua flexibilidade, pois, com apenas alterações em um de seus parâmetros, ela permite assumir comportamento crescente, decrescente e constante. Outra forma utilizada como função intensidade é a Log-linear que tem a forma $u(t) = exp(\lambda + \beta . t)$, (MEEKER; ESCOBAR, 1998).

A estimativa desses parâmetros fornece informações úteis sobre o comportamento de falha do sistema em análise. Existem diversos métodos para estimar os parâmetros das funções mencionadas anteriormente, mas o método mais popular é a estimação de parâmetros por meio da maximização da função de verossimilhança que será definida na próxima seção.

## 2.3 **Estimação por máxima verossimilhança**

A estimação por máxima verossimilhança é possível para casos em que há dados de falhas disponíveis e é usado para estimar os parâmetros para um ou mais sistemas idênticos com mesmo modo de falha. Há duas maneiras de observar os dados de um sistema reparável:

– dados truncados na falha, quando o sistema é observado até um número fixo de falhas ocorridas;

– dados truncado no tempo, quando o sistema é observado por um período de tempo fixo.

Em geral a estratégia de definição de política de periodicidade de MP é baseada no tempo de operação do sistema (MOUBRAY, 1997). Portanto, os dados de falha dos sistemas reparáveis sujeitos a manutenção preventiva, geralmente, são truncados no tempo.

Considere $m$ sistemas reparáveis idênticos e sob a mesma condição de uso. E a observação de $n_k$ falhas com reparação mínima no $k$-ésimo sistema ($k$=1,2,...,$m$), sendo $t_{ki}$ ($i$=1,2,...,$n_k$) o tempo de falha do $k$-ésimo sistema com $0 < t_{k1} < t_{k2} < ... < t_{ki}$ e observado por um período de tempo $(0, T_k]$ (truncado no tempo).

Assumindo que o processo para os $m$ sistemas possam ser modelados por um processo de Poisson com parâmetros $\omega = (\lambda,\beta)$, a função verossimilhança é resultado da combinação da função densidade das falhas dos $m$ sistemas e é dada pela empressão (2.3) (RIGDON; BASU, 2000).

$$L(\omega) = \prod_{k=1}^{m} \left[ \prod_{i=1}^{n_k} u(t_{ki}) . exp\left( - \int_0^{T_k} u(x) dx \right) \right] \tag{2.3}$$

Quando os dados de um determinado sistema *k* forem observados até uma determinada quantidade de falhas (truncado na falha), basta substituir o $T_k$ na expressão (2.3) pelo tempo de falha $t_{knk}$.

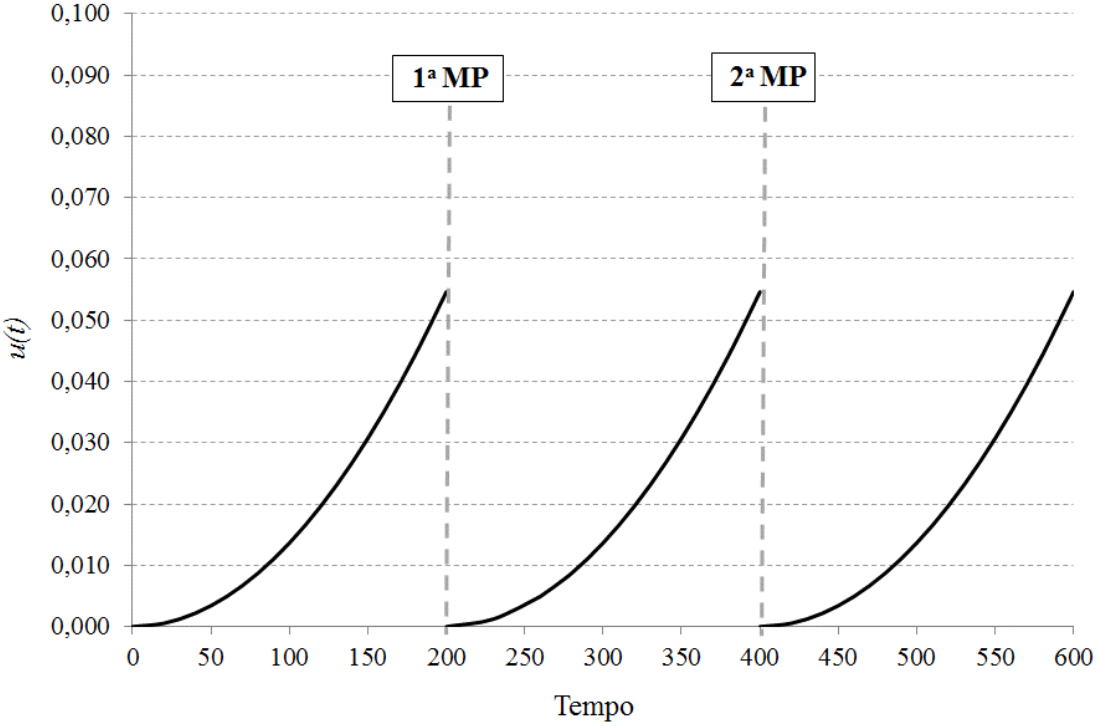
Utilizando o modelo paramétrico da Lei da Potência, com parâmetros $\omega = (\lambda,\beta)$, como a função intensidade de falha, a função verossimilhança fica dada por 2.4.

$$L(\lambda,\beta) = \prod_{k=1}^{m}\left[\prod_{i=1}^{n_k}\frac{1}{\lambda^{\beta}}.\beta.t_{ki}{}^{(\beta-1)}.exp\left(-\frac{1}{\lambda^{\beta}}.T_k{}^{\beta}\right)\right] \tag{2.4}$$

Em geral, esses parâmetros são desconhecidos e precisam ser estimados com base nos dados de falhas observados. Os valores de $\lambda, \beta$ que maximizam $L(\lambda, \beta)$ são denominadas as estimativas de máxima verossimilhança. A maximização da função verossimilhança (2.4) pode ser feita analiticamente seguindo a formulação apresentada em Rigdon e Basu (2000) ou, de forma numérica, utilizando algoritmos de otimização. As estimativas encontradas podem ser substitutas nas funções $u(t)$ e $\mu(t)$ para predição das métricas do comportamento de falha do sistema

O modelo PPNH para sistemas reparáveis tem uma limitação com relação ao impacto do reparo preventivo no desempenho do sistema, ou seja, consideram que são renovados para a condição de início de vida quando sofrem uma ação preventiva, ou seja, não consideram os estágios intermediários de reparo preventivo. A Figura 2.2 representa um exemplo da função da intensidade do modelo PPNH considerando a manutenção preventiva perfeita (MPP) e parâmetros Beta=3 e Lambda=130.

Figura 2.2 – função intensidade de falha do modelo PPNH




Fonte: elaborada pelo autor

No Capitulo 3 será apresentado um modelo que permite atender estados intermediários da ação de manutenção preventiva, bem como a proposição de um novo modelo para considerar diferentes condições de realização das atividades de manutenção preventiva imperfeita.

# 3. MODELOS PARA MANUTENÇÃO PREVENTIVA IMPERFEITA

Uma ação de manutenção tem o objetivo de retornar um sistema a um estado em que este tenha capacidade de desempenhar sua função adequadamente. Entre os modelos mais utilizados para representar o comportamento de falha de um sistema, estão o Processo de Renovação (PR) e o Processo de Poisson (PP).

O PR é um processo estocástico no qual os tempos entre falhas são variáveis aleatórias independentes e igualmente distribuídas (i.i.d.) e são representadas por uma distribuição de probabilidade. Ele é utilizado para sistemas os quais, após uma ação de reparo corretivo, retornam à operação com o mesmo desempenho que tinham quando novos, em geral, são utilizados para sistemas não reparáveis e que são substituídos por um novo.

O PP assume que a manutenção corretiva recupera o sistema falho a uma condição de desempenho que este possuía imediatamente antes da falha e não altera seu desempenho. Para utilizar o PP quando o sistema recebe a ação de manutenção preventiva, supõe-se que após a ação o sistema retorna à operação com o mesmo desempenho que tinha quando novo. Esta é uma hipótese que, na maioria das vezes, não acontece na prática já que a manutenção preventiva melhora o desempenho do sistema para uma condição intermediária e não para a condição de novo. Essa ação é denominada Manutenção Preventiva Imperfeita (MPI).

Conforme descrito na Tabela 1.1, alguns estudos estão utilizando a suposição MPI considerando um novo parâmetro na função de intensidade, porém poucos têm explorado a questão da estimativa desse novo parâmetro com base em dados. Existe, então, a necessidade de desenvolver métodos estatísticos para estimação desse novo parâmetro, bem como o cálculo dos limites de confiança para avaliação da variabilidade das estimativas e métodos para avaliação do ajuste adequado dos modelos.

Há outra característica relevante na modelagem do comportamento de falha de um sistema reparável. As ações de MP realizadas ao longo da vida do equipamento podem ser diferentes com relação ao número de tarefas realizadas e itens substituídos (MOUBRAY, 1997). Desse modo o parâmetro que caracteriza a MPI não é constante ao longo do tempo e é dependente da característica da ação de MP realizada. Neste trabalho será denominada “severidade” como a característica relacionada ao número de tarefas realizadas e/ou itens substituídos da MP.

Nesse capitulo serão apresentados dois modelos para considerar a MP imperfeita: redução de idade e redução de intensidade de falha. Também será proposto um novo modelo que considera a severidade da MP realizada, possibilitando relacionar o efeito da MP com a

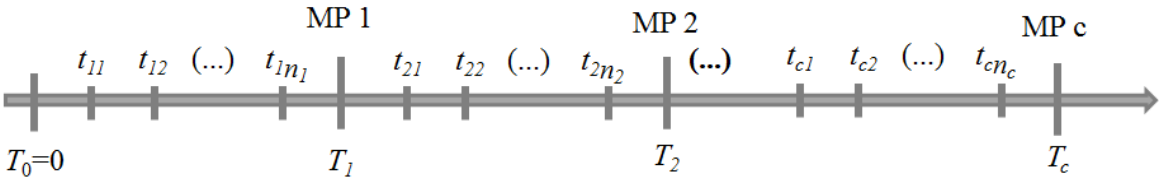
severidade de execução da ação de manutenção. Para cada parametrização (redução de idade e redução de intensidade), serão consideradas três possibilidades de relação com a severidade: linear, exponencial e potencial.

## 3.1 **Modelo de fator de melhoria**

Considere um sistema reparável que entre em operação no tempo 0 e as ações de manutenção preventiva (MP) ocorre em tempos predefinidos e os $c$ tempos sucessivas de execução das MPs são denotados $T_1, T_2, \ldots, T_c$ ($T_0 = 0 < T_1 < T_2 < \ldots < T_c$ ). O tempo 0 não significa necessariamente o início das atividades do sistema, mas o início de sua observação.

Este sistema é sujeito à manutenção corretiva (MC) com reparo mínimo (a ação restaura o estado da máquina para a sua condição pouco antes da falha), e observados $n$ tempos de falhas ($n=n_1+n_2+\ldots+n_c$) com $n_j$ ($j$=1,2,...,$c$) o número de falhas em cada ciclo e sendo os tempos de falhas sucessivas de um determinado ciclo $j$ de MP ($j$=1,2,...,$c$) denotados por $t_{j1}, t_{j2}, \ldots, t_{jn_c}$ ($0 < t_{j1} < t_{j2} < \ldots < t_{jn_c}$). Esquematicamente tem-se a Figura 3.1.

Figura 3.1 – distribuição dos tempos até a falha $t$ e os ciclos de MP para um sistema.




Fonte: elaborada pelo autor

Quando há dados disponíveis para mais de um sistema, a definição anterior pode ser expandida inserindo mais um índice na notação das falhas referente a cada sistema. Ou seja, considerando $m$ sistemas reparáveis idênticos, sob a mesma condição de uso e por isso possam ser modelados por um mesmo processo de Poisson, insere-se um índice $k$ referente aos dados de falhas do $k$-ésimo sistema ($k$=1, 2, ... , $m$) observado.

O processo de modelagem usual considera que, após a execução de uma MP, o sistema retorna à condição em que estava no tempo de vida igual a zero, seja em relação à intensidade de falha ou à sua idade, ou seja, é considerado como novo. Porém na prática a ação preventiva muitas vezes restaura o equipamento até um determinado grau de renovação, trazendo seu estado a um nível entre novo e o seu estado anterior à execução da ação de manutenção. Uma alternativa para considerar essa condição em um processo de Poisson ocorre pela utilização do

conceito de fator de melhoria na função intensidade de falha. Ao estimar o fator de melhoria possibilita avaliar o efeito da MP imperfeita no comportamento de falha do sistema.

O fator de melhoria inclui um parâmetro na função intensidade de falha para modelar o efeito da MP ao longo da vida do sistema e há duas classes mais utilizadas do fator de melhoria no processo de contagem. A primeira considera um parâmetro para reduzir a idade do sistema após a ação de manutenção (KIJIMA et al., 1988), denominada idade virtual do sistema. A segunda classe foi introduzida por Nakagawa (1988) e utiliza um parâmetro para reduzir a intensidade de falha do equipamento. As duas parametrizações são as mais aplicáveis, pois consideram o deslocamento horizontal (redução de idade) e vertical (redução da intensidade) da intensidade de falha.

O comportamento da falha é modelado usando a Teoria do Processo de Contagem, mais especificamente os processos de Poisson não homogêneo (PPNH), utilizando a Lei da Potência como função intensidade e tempos de falhas sucessivos observados. Antes da primeira ação de MP, ou seja, para um tempo menor do que $T_1$, a função intensidade é denominada de função intensidade inicial $u_0(t)$. Dessa forma, a função intensidade, considerando os ciclos ($j$ = 1, 2, 3, ..., $c$) de MP, é a função intensidade inicial $u_0(t)$ ajustada por um fator de melhoria (redução de idade ou redução da intensidade) a cada ação preventiva no sistema.

Além de considerar uma situação mais realista na modelagem, outras vantagens da utilização do fator de melhoria para modelar um processo com MP imperfeita:

1. inclusão de somente mais um parâmetro;
2. permite mensurar a eficiência da MP;
3. utilizar a estimativa para procedimento de otimização de MP;

Nas seções 3.1.1 e 3.1.2, esses modelos serão descritos em detalhes.

### 3.1.1 **Modelo de fator de melhoria com método de redução de idade**

Nesse modelo considera-se que a MP restaura o sistema de tal modo que a função de intensidade no tempo $t$ é a mesma função, porém com um valor de intensidade de falha referente a um tempo menor. O conceito é que as ações de MP reduzem a idade do sistema em um fator denominado $a$ que é utilizado para medir o efeito de uma MP imperfeita (JACK, 1998). Após o j-ésimo ciclo de MP, a idade do sistema é reduzida $a.T_j$ unidades de tempo ou a idade virtual do sistema é $(1-a)T_j$.

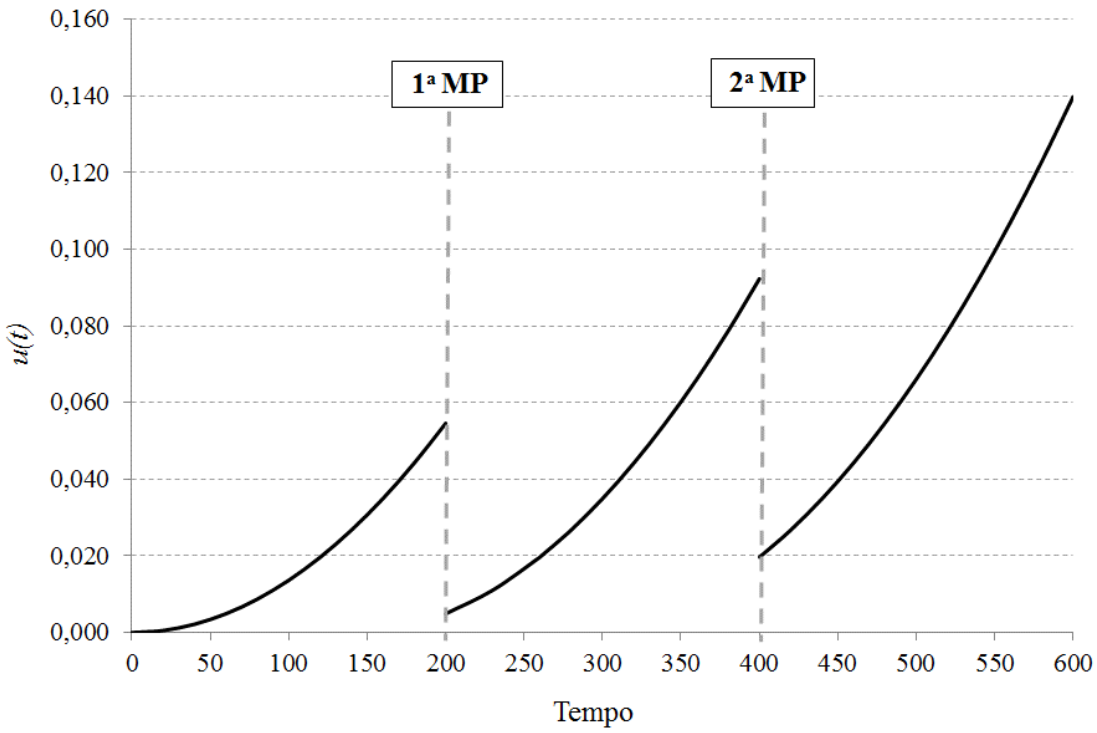
Utilizando uma parametrização similar a de Kijima et al. (1988), sendo $u_0(t)$ a função de intensidade inicial antes da primeira ação preventiva e transcorrido *j* ciclos de MP, tem-se a função de intensidade conforme 3.1.

$$u(t) = \begin{cases} u_0(t), \quad se \;\; T_0 = 0 \leq t < T_1 \\ u_0(t - aT_1), \;\; se \;\; T_1 \leq t < T_2 \\ u_0(t - aT_2), \;\; se \;\; T_2 \leq t < T_3 \\ \qquad \ldots \\ u_0\left(t - aT_{j-1}\right), \;\; se \;\; T_{j-1} \leq t < T_j \end{cases} \tag{3.1}$$

No modelo acima, quando $a$=1 corresponde ao modelo de manutenção preventiva perfeita (MPP), enquanto $a$=0 significa uma manutenção tão ruim que mantém a mesma idade do sistema após a ação preventiva. O caso de um 0< $a$ <1 corresponde a uma MP imperfeita. Casos não esperados seriam $a$<0, que é uma ação de manutenção que piora a idade do equipamento, ou seja, o equipamento fica pior que antes da ação de manutenção e o caso em que $a$>1 sugere um sistema restaurado a uma condição de melhor do que novo, ou seja, o equipamento é modernizado.

Em termos físicos, o fator denominado $a$ pode ser interpretado como um índice de eficácia da ação de manutenção. A Figura 3.2 representa a função da intensidade do modelo com $a$ = 0,70.

Figura 3.2 – efeito na função intensidade de falha do modelo redução de idade




Fonte: elaborada pelo autor

Sendo a função valor médio inicial $\mu_0(t)$ a função antes da primeira ação preventiva e utilizando a relação entre $u(t)$ e $\mu(t)$, segue-se que a função de valor médio considerando o fator de melhoria de redução de idade é definida na equação (3.2):

$$\mu(t) = \sum_{j=1}^{c-1} \left\{ \left[ \mu_0(T_j - aT_{j-1}) - \mu_0\left((1-a)T_{j-1}\right) \right] \right\} + \left[ \mu_0(t - aT_{c-1}) - \mu_0\left((1-a)T_{c-1}\right) \right]$$
$$com\ \ T_{j-1} \leq t < T_j \qquad (3.2)$$

Será utilizada a função potência proposta por Crow (1974) e definida em (2.2) como intensidade de falha inicial $u_0(t)$. Essa escolha se faz principalmente devido à flexibilidade da função potência, pois, com apenas alterações no parâmetro $\beta$, ela permite assumir comportamento crescente, decrescente e constante da intensidade de falha do processo ao longo do tempo.

Utilizando a lei da potência como forma funcional da intensidade de falha do processo, têm-se as seguintes funções de intensidade e valor médio respectivamente:

$$u(t) = \frac{1}{\lambda^{\beta}}.\beta.\left(t - aT_{j-1}\right)^{(\beta-1)}, \;\; com \;\; T_{j-1} \leq t < T_j \tag{3.3}$$

$$\mu(t) = \sum_{j=1}^{c-1}\left\{\left[\frac{1}{\lambda^{\beta}}.\left(T_j - aT_{j-1}\right)^{\beta} - \frac{1}{\lambda^{\beta}}.\left((1-a)T_{j-1}\right)\right]\right\} + \left[\frac{1}{\lambda^{\beta}}.(t - aT_{c-1})^{\beta} - \frac{1}{\lambda^{\beta}}.\left((1-a)T_{c-1}\right)^{\beta}\right]$$
$$com \;\; T_{j-1} \leq t < T_j \tag{3.4}$$

Portanto, (3.3) e (3.4) são respectivamente as funções intensidade e valor médio considerando o ajuste da idade do equipamento pelo fator de melhoria em cada ação preventiva no sistema.

### 3.1.2 **Modelo de fator de melhoria com método de redução da intensidade de falha**

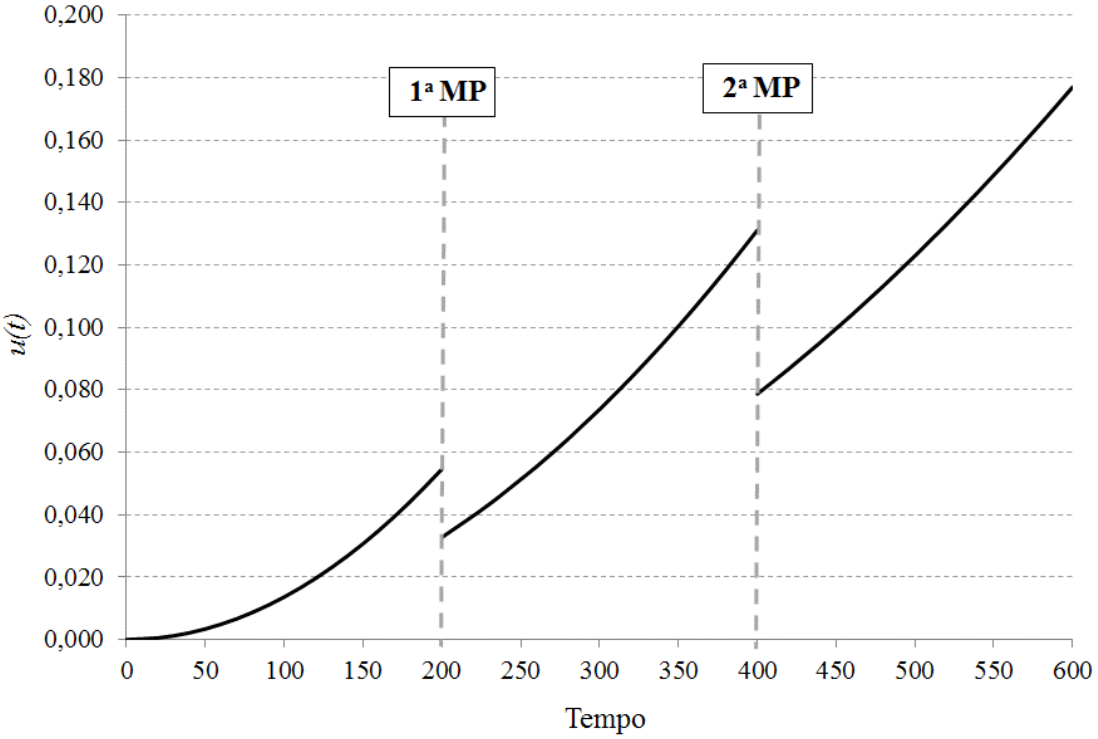
Nessa classe de modelo, ação de MP rejuvenesce o sistema, pois a função intensidade é reduzida pela multiplicação de um fator de melhoria. Utilizando o fator denominado *b*, é possível adicionar uma redução da função após uma ação de MP imperfeita. Sendo $u_0(t)$ a função de intensidade inicial antes da primeira MP e depois de transcorridos *j* ciclos de MP, tem-se a seguinte função de intensidade:

$$u(t) = \begin{cases} u_0(t), \;\; se \;\; T_0 = 0 \leq t < T_1 \\ (1-b).u_0(t), \;\; se \;\; T_1 \leq t < T_2 \\ (1-b)^2.u_0(t), \;\; se \;\; T_2 \leq t < T_3 \\ \ldots \\ (1-b)^{j-1}.u_0(t), \;\; se \;\; T_{j-1} \leq t < T_j \end{cases} \tag{3.5}$$

Se $0<b<1$ corresponde à manutenção imperfeita. O caso $b=0$ sugere que não há efeito da MP na intensidade de falha do sistema. Os casos de $b\geq1$ são inexistentes, pois resultaria em uma função intensidade negativa ou zerada. Casos não esperados, seriam $b<0$ que é uma ação de MP que piora a função intensidade.

Foi utilizada a parametrização *(1-b)* na função intensidade apenas para facilitar a interpretação dos fatores de melhoria que, quanto maior o fator, mais eficiente seria o efeito da MP. A Figura 3.3 mostra a função intensidade do modelo com $b$ = 0,40.

Figura 3.3 – efeito na função intensidade de falha do modelo redução de falha




Fonte: elaborada pelo autor

Sendo a função valor médio inicial $\mu_0(t)$, a função antes da primeira ação preventiva e integrando a função $u(t)$ definida anteriormente, a função de valor médio é definida em (3.6):

$$\mu(t) = \sum_{j=1}^{c-1}\{(1-b)^{j-1}.[\,\mu_0(T_j) - \mu_0(T_{j-1})]\} \; + \; (1-b)^{c-1}.[\,\mu_0(t) - \mu_0(T_{c-1})]$$

$$com \;\; T_{j-1} \leq t < T_j \qquad (3.6)$$

Substituindo a função potência proposta por Crow (1974) em $u_0(t)$, para o modelo de redução de intensidade, tem-se:

$$u(t) = (1-b)^{j-1}.\frac{1}{\lambda^{\beta}}.\beta.t^{(\beta-1)}, \;\; com \;\; T_{j-1} \leq t < T_j \qquad (3.7)$$

Consequentemente tem-se a seguinte função valor médio:

$$\mu(t) = \sum_{j=1}^{c-1} \left\{ (1-b)^{j-1} . \left[ \frac{1}{\lambda^{\beta}} . \left(T_j\right)^{\beta} - \frac{1}{\lambda^{\beta}} . \left(T_{j-1}\right)^{\beta} \right] \right\} + (1-b)^{c-1} . \left[ \frac{1}{\lambda^{\beta}} . t^{\beta} - \frac{1}{\lambda^{\beta}} . {T_{c-1}}^{\beta} \right]$$

$$com \;\; T_{j-1} \leq t < T_j \tag{3.8}$$

Portanto (3.7) e (3.8) são, respectivamente, as funções intensidade e valor médio considerando-se o ajuste da função intensidade pelo fator de melhoria que reduz a intensidade de falha em cada ação preventiva.

As duas parametrizações apresentadas são as mais aplicáveis na modelagem do fator de melhoria, pois considera o deslocamento horizontal (redução de idade) e vertical (redução da intensidade) da função intensidade de falha, possibilitando capturar diversos efeitos da ação preventiva no comportamento de falha.

## 3.2 **Fator de melhoria variável**

A manutenção imperfeita é uma condição importante a se considerar nos modelos de sistemas reparáveis, porém há outra característica relevante que pode ser considera na modelagem do comportamento de falha de um sistema. Durante a vida de um equipamento, diferentes planos de manutenção podem ser realizados com diferentes números de tarefas realizadas ou itens substituídos (MOUBRAY, 1997). Desse modo os parâmetros do fator de melhoria apresentados anteriormente não são constantes para as diferentes MP e são dependentes da severidade da ação de manutenção realizada no sistema.

Na prática, os efeitos dos fatores de melhoria de redução de idade e de redução de intensidade de falha, denominados respectivamente de $a$ e $b$, podem não ser uniformes durante os ciclos de manutenções preventivas de um mesmo equipamento. Ou seja, a execução da manutenção pode ser mais ou menos severa ao longo da vida de um sistema fazendo que o fator de melhoria possa ser diferente em cada ação preventiva. Assim, os fatores de melhoria apresentados anteriormente podem ser interpretados como o efeito médio durante o período de observação.

Para obter um modelo com maior precisão e que possibilite mensurar o efeito da ação de manutenção preventiva considerando diferentes condições de realização (número de tarefas

realizadas e/ou itens substituídos), será proposto um modelo que considere a severidade da MP realizada para cada parametrização do fator de melhoria, possibilitando relacionar o efeito da MP com a severidade de execução da ação de MP. Essa nova classe será denominada como modelos com Fator de Melhoria Variável, pois considera uma função em substituição ao parâmetro fixo de fator de melhoria. Em cada parametrização do fator para redução de idade e redução de intensidade, três possibilidades de funções serão consideradas: linear, exponencial e potencial.

Esse modelo possibilitará mensurar o impacto na confiabilidade do sistema de um novo plano de manutenção, bem como a otimização da periodicidade de preventiva para um novo plano de atividades ações de manutenção preventiva.

### 3.2.1 **Modelo de redução de idade com fator de melhoria variável**

Nesse modelo o conceito é que as ações de MP reduzem a idade do sistema, porém, para cada atuação preventiva, o fator de redução de idade pode apresentar diferentes níveis de redução em decorrência das distintas atividades de manutenção que são realizadas no sistema. Em decorrência desse fato, o fator de melhoria, denominado *a*, será substituído por uma função que relaciona a redução de idade com a severidade do ciclo de manutenção preventiva. Será denominada severidade como a característica relacionada as atividades de MP, como, por exemplo, número de tarefas realizadas e/ou itens substituídos.

Dessa forma não haverá um fator fixo para mensurar a redução de idade para ações preventivas, mas haverá uma função que relaciona a redução de idade com a severidade, denominada *s*, das atividades de manutenção preventiva.

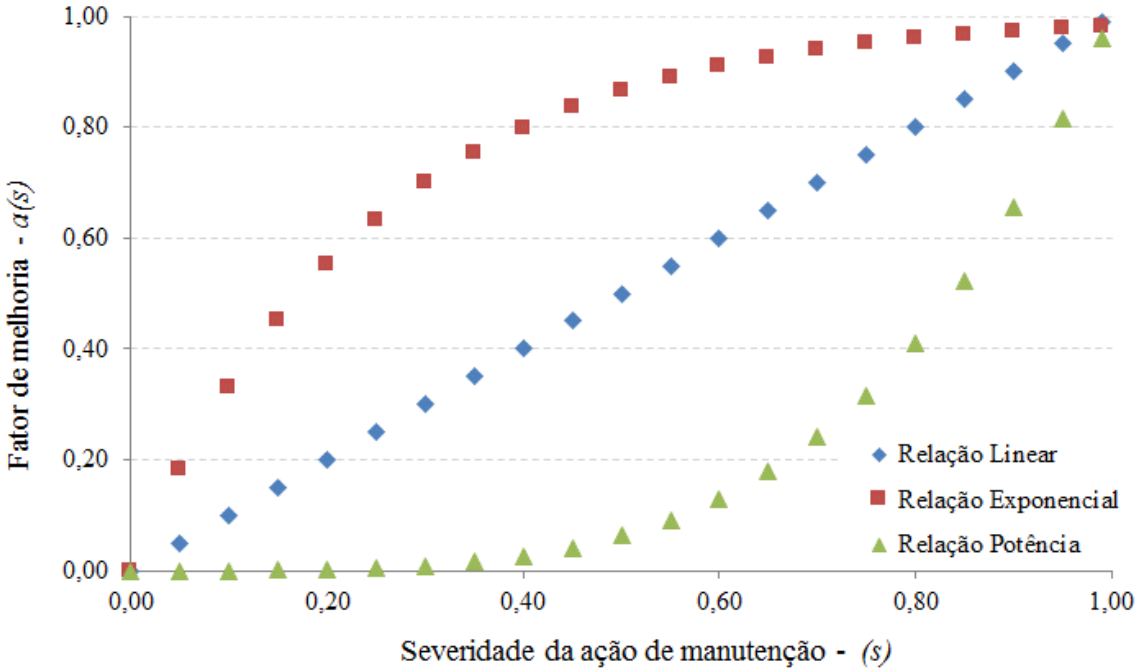
Para representar matematicamente essa relação, será definida a função denominada função fator de melhoria *a(s)*, composta pela variável $s$ ($0 < s \leq 1$) relacionada a severidade das atividades de manutenção executada e pelo parâmetro Theta ($\theta$), que representa o grau de variação do fator de melhoria para a variação da severidade de manutenção executada.

Foram selecionadas três formas funcionais que possam capturar os diversos comportamentos desta relação:

1. **Relação Linear:** $a_L(s) = s\,.\,\theta$ , com ($0 \leq \theta \leq 1$)
2. **Relação Exponencial:** $a_E(s) = 1 - Exp(-s.\theta)$, com ($0 \leq \theta \leq \infty$)
3. **Relação Potência:** $a_P(s) = s^{\,\theta}$, com ($0 \leq \theta \leq \infty$)

A Figura 3.4 representa três exemplos dessas relações entre severidade da manutenção preventiva e a função fator de melhoria *a(s)*.

Figura 3.4 – formas funcionais do fator de melhoria *a(s)*




Fonte: elaborada pelo autor

Utilizando os resultados obtidos no modelo de fator de melhoria com redução de idade e substituindo o fator, denominado *a*, pela função de fator de melhoria, denominada *a(s)*, têm-se as seguintes funções de intensidade e valor médio respectivamente:

$$u(t) = \frac{1}{\lambda^{\beta}}.\beta.\left(t - a(s).T_{j-1}\right)^{(\beta-1)}, \;\; com \;\; T_{j-1} \leq t < T_j \tag{3.9}$$

$$\mu(t) = \sum_{j=1}^{c-1}\left\{\left[\frac{1}{\lambda^{\beta}}.\left(T_j - a(s).T_{j-1}\right)^{\beta} - \frac{1}{\lambda^{\beta}}.\left(\left(1 - a(s)\right)T_{j-1}\right)\right]\right\} + \left[\frac{1}{\lambda^{\beta}}.(t - a(s).T_{c-1})^{\beta} - \frac{1}{\lambda^{\beta}}.\left(\left(1 - a(s)\right)T_{c-1}\right)^{\beta}\right]$$

$$com \;\; T_{j-1} \leq t < T_j \tag{3.10}$$

Ressaltando que a função *a(s)* pode ser substituída pelas três formas funcionais propostas e o parâmetro $\theta$ será estimado pelo método proposto no capitulo 4 desta tese.

### 3.2.2 **Modelo de redução de intensidade de falha com fator de melhoria variável**

O modelo de redução de intensidade considera que MP rejuvenesce o sistema, pois a função intensidade é reduzida por um fator de melhoria denominado *b*. Porém, em cada ação preventiva o nível de redução pode ser distinto em decorrência das atividades de manutenção realizada como número de tarefas realizadas e/ou itens substituídos. Desta forma, o fator *b* fixo será substituído por uma função que relaciona a redução de intensidade com a severidade, denominada *s*, das atividades realizadas no ciclo de manutenção preventiva.

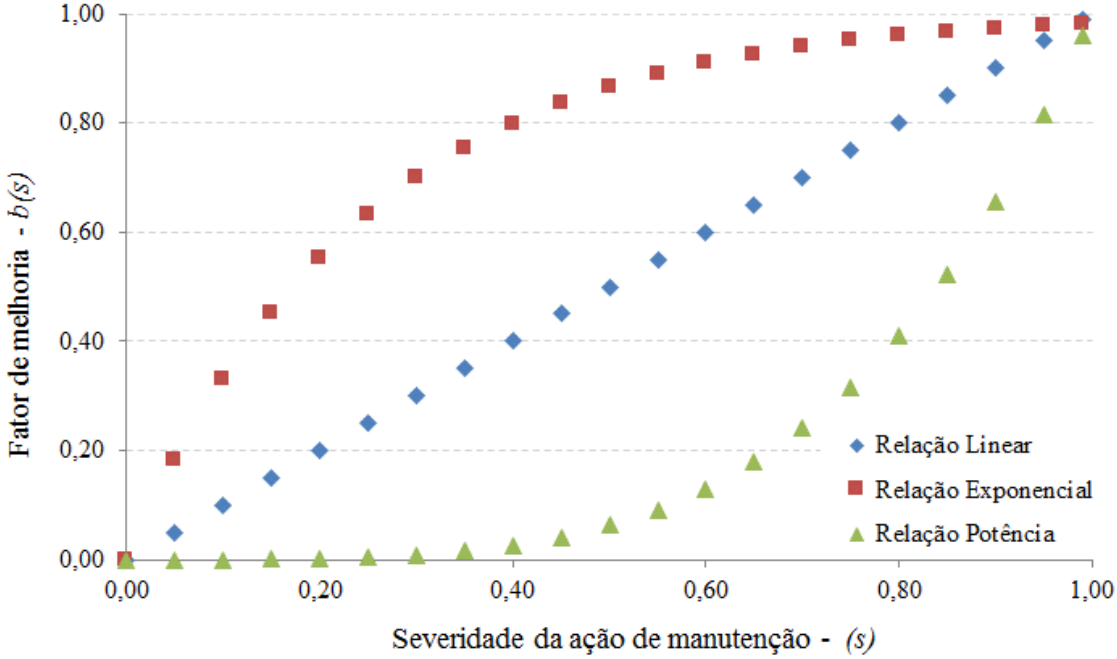
Para representar matematicamente a relação citada, será definida a função denominada função fator de melhoria *b(s)*, composta pela variável *s* ($0 < s \leq 1$) que relaciona a severidade das atividades de manutenção realizadas e pelo parâmetro Theta ($\theta$) que representa o grau de variação do fator de melhoria para a variação da severidade de manutenção.

Esta função será denominada função fator de melhoria *b(s)* e foram selecionadas três formas funcionais que possam capturar os diversos comportamentos desta relação:

1. **Relação Linear:** $b_L(s) = s\,.\,\theta$, com $(0 \leq \theta \leq 1)$
2. **Relação Exponencial:** $b_E(s) = 1 - Exp(-s.\theta)$, com $(0 \leq \theta \leq \infty)$
3. **Relação Potência:** $b_P(s) = s^{\,\theta}$, com $(0 \leq \theta \leq \infty)$

A Figura 3.5 representa três exemplos dessas relações entre severidade da manutenção e a função fator de melhoria *b(s)* que é similar a Figura 3.4 por se tratar das mesmas formas funcionais.

Figura 3.5 – formas funcionais do fator de melhoria *b(s)*




Fonte: elaborada pelo autor

Utilizando os resultados obtidos no modelo de fator de melhoria de redução de intensidade de falha e substituindo o fator fixo, denominado *b*, pela função de fator de melhoria *b(s)*, têm-se as seguintes funções de intensidade e valor médio respectivamente:

$$u(t) = \left(1 - b(s)\right)^{j-1} . \frac{1}{\lambda^{\beta}} . \beta . t^{(\beta-1)}, \;\; com \;\; T_{j-1} \leq t < T_j \tag{3.11}$$

$$\mu(t) = \sum_{j=1}^{c-1} \left\{ \left(1 - b(s)\right)^{j-1} . \left[ \frac{1}{\lambda^{\beta}} . \left(T_j\right)^{\beta} - \frac{1}{\lambda^{\beta}} . \left(T_{j-1}\right)^{\beta} \right] \right\} + \left(1 - b(s)\right)^{c-1} . \left[ \frac{1}{\lambda^{\beta}} . t^{\beta} - \frac{1}{\lambda^{\beta}} . {T_{c-1}}^{\beta} \right]$$
$$com \;\; T_{j-1} \leq t < T_j \tag{3.12}$$

A função *b(s)* pode ser substituída pelas três formas funcionais propostas e o parâmetro $\theta$ será estimado pelo método proposto no capitulo 4 desta tese.

Em resumo, propõem-se oito modelos que possam ser utilizados na modelagem de um processo de falha de um sistema reparável, considerando manutenção preventiva imperfeita, possibilitando capturar os diversos comportamentos de falha que possam surgir em uma análise, são eles:

1. Modelo de fator de melhoria pelo método redução de idade. (Modelo RI);
2. Modelo de fator de melhoria pelo método redução de intensidade de falha. (Modelo RF);
3. Modelo pelo método redução de idade com fator de melhoria variável utilizando relação linear. (Modelo RIL);
4. Modelo pelo método redução de idade com fator de melhoria variável utilizando relação exponencial. (Modelo RIE);
5. Modelo pelo método redução de idade com fator de melhoria variável utilizando relação potência. (Modelo RIP);
6. Modelo pelo método redução de intensidade de falha com fator de melhoria variável utilizando relação linear. (Modelo RFL);
7. Modelo pelo método redução de intensidade de falha com fator de melhoria variável utilizando relação exponencial. (Modelo RFE);
8. Modelo pelo método redução de intensidade de falha com fator de melhoria variável utilizando relação potência. (Modelo RFP);

### 3.2.3 **Modelo de *Balanced Scored Card* para severidade da MP**

As formas funcionais dos fatores de melhoria possuem a variável denominada *s* para relacionar a severidade das atividades de manutenção com efeito de melhoria do sistema. Dessa forma, *s* é um número, entre 0 e 1, que mensura o quão severa é a ação de manutenção preventiva e está relacionado ao número de tarefas realizadas e/ou itens substituídos da MP. Em outras palavras, *s* pode ser definido como o grau de intervenção da ação preventiva.

Por exemplo, em um determinado plano de manutenção preventiva denominado A, são realizadas 20 tarefas e 2 itens são substituídos. Em outro plano, denominado B, são realizadas 35 tarefas e 5 itens são substituídos. Portanto, o plano B possui uma maior severidade que a ação A. Isso não quer dizer que o efeito de melhoria da ação B é maior que da ação A, indagação que será respondida após a estimação do modelo proposto, mas diz que a manutenção realizada pela ação B possui uma maior atuação no sistema quando comparado à ação A.

A variável *s* deverá ser predeterminada para cada manutenção preventiva pelos engenheiros de manutenção por meio do conhecimento das ações executadas na atividade. Uma maneira para padronizar a variável *s* é pela proposta de um modelo de *Balanced Scored Card* (BSC) que irá mensurar o valor para essa variável, considerando diferentes condições de manutenção preventiva.

Moubray (1997) propõe diversas análises para definição de planos de manutenção com foco na confiabilidade. Baseadas nessas proposições, foram selecionadas três perguntas que caracterizam a severidade da ação de manutenção em um sistema:

1. quantas tarefas tem o plano de MP? A resposta será denominada *RP1*;
2. quanto tempo (em horas) é executado o plano de MP? A resposta será denominada *RP2*;
3. quantos itens são substituídos na MP? A resposta será denominada *RP3*.

As perguntas podem ser adaptadas ou acrescidas dependendo da especificidade do sistema em análise. Elas devem considerar outros pontos que os engenheiros de manutenção considerem refletir na eficiência da ação de manutenção, por exemplo, uso de tecnologias diferentes em determinadas tarefas. O importante é que sejam determinadas perguntas e pontuações de condições que possam refletir diferenças apenas nas ações de manutenção, pois

com relação a eficiência será avaliada após a estimação do modelo de fator de melhoria variável.

Supondo ter *j* planos de manutenção (*j*=1, ... , *c*) e as três perguntas serem respondidas para cada plano de manutenção (ou condição de manutenção), a severidade *s* de um determinado plano *j* será definida pela seguinte padronização:

$$s_j = \frac{\frac{RP1_j}{\max_j RP1} + \frac{RP2_j}{\max_j RP2} + \frac{RP3_j}{\max_j RP3}}{3} \tag{3.13}$$

A condição de manutenção deverá ser calculada para cada tempo de falha disponível para caracterizar a severidade da manutenção preventiva a que cada falha estava exposta. Após estimar o modelo, também poderá ser definida uma condição de manutenção *s*, sob um novo plano de manutenção, para predição de confiabilidade do sistema considerando uma ação de MP diferente das anteriores.

# 4. **ESTIMAÇÃO NOS MODELOS PROPOSTOS**

Neste capitulo são definidas as funções de verossimilhança associadas aos modelos propostos no Capítulo 3 que consideram a realização de atividades de manutenção preventiva imperfeita durante seu processo de operação.

Essas funções são construídas com base nas informações de falha e manutenções dos sistemas envolvidos na análise e serão utilizadas para obter estimativas dos parâmetros dos modelos propostos. Os valores dos parâmetros que maximizam a função de verossimilhança são denominados Estimadores de Máxima Verossimilhança (EMV) e podem ser substituídos nas funções intensidade de falha e valor médio para análise e previsão do comportamento de falha do sistema.

Apesar dos EMV possuírem ampla aplicação na análise de sistemas reparáveis, há necessidade de descrever e adaptar a obtenção desses estimadores para o escopo dos novos modelos propostos que consideram o fator de melhoria variável. Neste capitulo também será discutida a construção do intervalo de confiança para essas estimativas, e, por fim, será apresentado um estudo para avaliação das propriedades assintóticas dos estimadores propostos.

## 4.1 **Definição das funções de verossimilhança**

Considere $m$ sistemas reparáveis idênticos em que as falhas ocorrem de forma independente uma da outra e sob as seguintes condições:

– em cada falha que ocorre, há a execução de uma ação de reparo mínimo.

– $n_{kj}$ falhas são observadas no k-ésimo sistema ($k$=1,..., $m$) e no j-ésimo ciclo de MP ($j$=1,..., $c_k$), sendo $c_k$ o número de MPs de um determinado sistema $k$.

– seja $t_{kji}$ ($k$=1,…,$m$; $j$=1,…,$c_k$; $i$=1,..,$n_{kj}$) o tempo de falha observado para o k-ésimo sistema, no j-ésimo ciclo de MP e i-ésima falha, sendo os tempos de falhas sucessivas de um determinado sistema $k$ e ciclo $j$ de MP ($0 < t_{kj1} < t_{kj2} < ... < t_{kjn_{kj}}$).

– seja $T_{kj}$ ($k$=1,…,$m$; $j$=1,…,$c_k$) o tempo predefinido da ação de MP do k-ésimo sistema e j-ésimo ciclo de MP. Os $c_k$ são os tempos sucessivos de execução das MPs de um determinado sistema $k$ são denotados $T_{k1}, T_{k2}, ... , T_{kc_k}$ ($T_0 = 0 < T_{k1} < T_{k2} < , ..., < T_{kc_k}$ ).

– seja $s_{kj}$ ($j$=1,…,$c_k$) a severidade das atividades de manutenção preventiva realizada no k-ésimo sistema ($k$=1,..., $m$) e no j-ésimo ciclo de MP.

Nos modelos propostos, a função de intensidade depende dos ciclos de MP. Ou seja, os tempos de MP proporcionam intervalos disjuntos fixos, ($T_0=0$, $T_{k1}$], ($T_{k1}$, $T_{k2}$], ..., ($T_{kCk-1}$, $T_{kCk}$] e, a cada MP executada, uma ação de um determinado grau é executada. Portanto, para o intervalo de tempo ($T_{k\ j-1}$, $T_{k\ j}$) do j-ésimo ciclo de manutenção, tem-se uma função de intensidade *u(t)* ajustada pelo fator de melhoria.

Em decorrência desse ajuste da função intensidade a cada ação de MP, há necessidade de ajustar a função de verossimilhança definida em (2.4), pois, no processo que considera a MP imperfeita, a função de verossimilhança será a combinação da função densidade probabilidade conjunta de um PPNH e considerando que função de intensidade se altera ao longo dos ciclos de preventiva.

Fazendo a adaptação descrita e denominando $\omega$ como sendo o vetor genérico de parâmetros da função utilizada como intensidade de falha, tem-se a seguinte função de verossimilhança:

$$L(\omega) = \prod_{k=1}^{m}\left\{\prod_{j=1}^{c_k}\left[\prod_{i=1}^{n_{kj}} u(t_{kji})\right].exp\left(-\int_0^{T_{k\,c_k}} u(x)dx\right)\right\} \tag{4.1}$$

Como, para cada intervalo de tempo ($T_{k\,j-1}$, $T_{k\,j}$) há uma correção distinta para função de intensidade *u(t)*, a função de verossimilhança considerando o modelo de fator de melhoria pode ser reescrita da seguinte forma:

$$L(\omega) = \prod_{k=1}^{m}\left\{\prod_{j=1}^{c_k}\left[\prod_{i=1}^{n_{kj}} u(t_{kji})\right].exp\left[-\sum_{j=1}^{c_k}\left(\int_{T_{k\,j-1}}^{T_{k\,j}} u(x)dx\right)\right]\right\} \tag{4.2}$$

Utilizando a relação entre *u*(*t*) e $\mu(t)$, a função de verossimilhança revela-se como:

$$L(\omega) = \prod_{k=1}^{m}\left\{\prod_{j=1}^{c_k}\left[\prod_{i=1}^{n_{kj}} u(t_{kji})\right].exp\left[-\sum_{j=1}^{c_k}\left(\mu(T_{k\,j}) - \mu(T_{k\,j-1})\right)\right]\right\} \tag{4.3}$$

A verossimilhança definida em (4.3) constitui a função para os modelos de fator de melhoria propostos sendo necessária a substituição das funções *u(t)* e $\mu(t)$ do modelo correspondente definido no capítulo 3. A seguir serão apresentadas as funções de verossimilhança para cada modelo proposto com as substituições específicas.

**Função verossimilhança para o modelo RI**

Considere um processo de falhas como definido anteriormente e modelado por um PPNH com função de intensidade inicial seguindo a Lei de Potência, com modelo de fator de melhoria de redução de idade (modelo RI) e com parâmetros $\omega = (\lambda, \beta, a)$. Substituindo na função de verossimilhança (4.3) as funções de intensidade de falha e valor médio do modelo RI, definidas em (3.3) e (3.4) respectivamente, tem-se:

$$L(\lambda,\beta,a) = \prod_{k=1}^{m}\left\{\prod_{j=1}^{c_k}\left[\prod_{i=1}^{n_{kj}}\frac{1}{\lambda^{\beta}}.\beta.\left(t_{kji} - aT_{k\,j-1}\right)^{(\beta-1)}\right].\right.$$
$$\left.\left..\,exp\left[-\sum_{j=1}^{c_k}\left(\frac{1}{\lambda^{\beta}}.\left(T_{k\,j} - aT_{k\,j-1}\right)^{\beta} - \frac{1}{\lambda^{\beta}}.\left((1-a)T_{k\,j-1}\right)^{\beta}\right)\right]\right\} \qquad (4.4)$$

**Função verossimilhança para o modelo RF**

Considere um processo de falhas com MP imperfeita conforme definido e modelado por um PPNH com função de intensidade inicial seguindo a Lei de Potência, modelo de fator de melhoria de redução de intensidade de falha (modelo RF) e com parâmetros $\omega = (\lambda, \beta, b)$. Substituindo na verossimilhança (4.3) as funções de intensidade de falha e valor médio do modelo RF, definidas em (3.7) e (3.8) respectivamente, tem-se:

$$L(\lambda,\beta,b) = \prod_{k=1}^{m}\left\{\prod_{j=1}^{c_k}\left[\prod_{i=1}^{n_{kj}}(1-b)^{j-1}.\frac{1}{\lambda^{\beta}}.\beta.\left(t_{kji}\right)^{(\beta-1)}\right].\right.$$
$$\left.\left..\,exp\left[-\sum_{j=1}^{c_k}(1-b)^{j-1}.\left(\frac{1}{\lambda^{\beta}}.\left(T_{k\,j}\right)^{\beta} - \frac{1}{\lambda^{\beta}}.\left(T_{k\,j-1}\right)^{\beta}\right)\right]\right\} \qquad (4.5)$$

**Função verossimilhança para o modelo RIL, RIE e RIP**

Para definir a função verossimilhança dos modelos de manutenção imperfeita considerando o fator de melhoria varável considere um processo de falhas, o qual a cada ação preventiva é realizada uma atividade de manutenção imperfeita caracterizada pela variável

denominada *s*. Esse processo será caracterizado por um PPNH, com função de intensidade inicial seguindo a Lei de Potência, utilizando o método redução de idade com fator de melhoria variável denominada *a(s)* e com parâmetros $\omega = (\lambda,\beta,\theta)$. Substituindo em (4.3) as funções de intensidade de falha e valor médio do modelo com fator de melhoria variável, definidas em (3.9) e (3.10) respectivamente, tem-se:

$$L(\lambda,\beta,\theta) = \prod_{k=1}^{m}\left\{\prod_{j=1}^{c_k}\left[\prod_{i=1}^{n_{kj}}\frac{1}{\lambda^{\beta}}.\beta.\left(t_{kji} - a\left(s_{k\,j-1}\right).T_{k\,j-1}\right)^{(\beta-1)}\right].\right.$$
$$\left..\,exp\left[-\sum_{j=1}^{c_k}\left(\frac{1}{\lambda^{\beta}}.\left(T_{k\,j} - a\left(s_{k\,j-1}\right).T_{k\,j-1}\right)^{\beta} - \frac{1}{\lambda^{\beta}}.\left((1 - a\left(s_{k\,j-1}\right))T_{k\,j-1}\right)^{\beta}\right)\right]\right\} \quad (4.6)$$

Para definir as funções verossimilhanças dos modelos RIL, RIE e RIP deve-se substituir em (4.6) a função *a(s)* pelas três formas funcionais propostas. Para verossimilhança do modelo RIL deverá substituir *a(s)* pela relação linear $a_L(s) = s\,.\theta$, para do modelo RIE deverá substituir pela relação exponencial $a_E(s) = 1 - Exp(-s.\theta)$ e para do modelo RIP substituir pela relação potência $a_P(s) = s^{\,\theta}$.

**Função verossimilhança para o modelo RFL, RFE e RFP**

A função verossimilhança dos modelos, considerando-se o fator de melhoria varável que utiliza o método de redução de intensidade de falha, pode ser definida considerando um processo de falhas, em que, a cada ação preventiva é realizada uma atividade de manutenção imperfeita caracterizada pela variável denominada *s*. Esse processo será caracterizado por um PPNH, com função de intensidade inicial seguindo a Lei de Potência, utilizando o método redução de intensidade de falha com fator de melhoria variável denominada *b(s)* e com parâmetros $\omega = (\lambda,\beta,\theta)$. Substituindo em (4.3) as funções de intensidade de falha e valor médio do modelo com fator de melhoria variável, definidas em (3.11) e (3.12) respectivamente, tem-se:

$$L(\lambda,\beta,\theta) = \prod_{k=1}^{m}\left\{\prod_{j=1}^{c_k}\left[\prod_{i=1}^{n_{kj}}\left(1-b\left(s_{k\,j-1}\right)\right)^{j-1}.\frac{1}{\lambda^{\beta}}.\beta.\left(t_{kji}\right)^{(\beta-1)}\right].\right.$$

$$\left..\,exp\left[-\sum_{j=1}^{c_k}\left(1-b\left(s_{k\,j-1}\right)\right)^{j-1}.\left(\frac{1}{\lambda^{\beta}}.\left(T_{k\,j}\right)^{\beta}-\frac{1}{\lambda^{\beta}}.\left(T_{k\,j-1}\right)^{\beta}\right)\right]\right\} \qquad (4.7)$$

Para definir as funções verossimilhanças dos modelos RFL, RFE e RFP deve-se substituir em (4.7) a função *b(s)* pelas três formas funcionais propostas. Para verossimilhança do modelo RIL deverá substituir *b(s)* pela relação linear $b_L(s) = s\,.\theta$, para do modelo RIE deverá substituir pela relação exponencial $b_E(s) = 1 - Exp(-s.\theta)$ e para do modelo RIP substituir pela relação potência $b_P(s) = s^{\,\theta}$.

## 4.2 **Estimação dos parâmetros**

Os valores dos parâmetros que maximizam a função de verossimilhança são denominados Estimadores de Máxima Verossimilhança (EMV). Devido à complexidade matemática das funções de verossimilhança não é possível estimar os parâmetros desconhecidos por métodos analíticos. Portanto foi utilizado método numérico para determinar o valor dos parâmetros que maximizem a função de verossimilhança. Para aplicação do método, o logaritmo natural (*ln*) foi aplicado em ambos os lados da função de verossimilhança, o que torna, computacionalmente, mais fácil a maximização da função.

O procedimento realizado utilizou o algoritmo de *Nelder-Mead*, implantado na linguagem R (www.R-project.org, v.3.1.2), na rotina OPTIM. O algoritmo de *Nelder-Mead* foi selecionado por ser um método de otimização de funções multidimensionais e que não precisa das derivadas da função (NELDER; MEAD, 1965). O método baseia-se na avaliação de uma função nos vértices de um simplex, em seguida iterativamente diminuindo o simplex com melhores pontos até que uma determinada precisão seja obtida.

No procedimento de maximização da verossimilhança foi realizada uma verificação para ratificar que o valor encontrado dos parâmetros pelo algoritmo é um ponto de máximo, para tal foi avaliado se os determinantes das matrizes de dimensões 1, 2 e 3 das derivadas parciais da função verossimilhança (denominada Hessiano) possuem alternadamente sinais positivos e negativos no ponto encontrado pelo algoritmo, conforme Bussab et al. (2003).

## 4.3 **Cálculo dos limites de confiança**

Uma propriedade importante para a construção de intervalos de confiança é a que diz respeito à distribuição assintótica do estimador de máxima verossimilhança. Sendo $\omega$ o vetor de $q$ parâmetros ($i$=1,...,$q$) $\omega_i$, essa propriedade define, sob certas condições de regularidade, que $\widehat{\omega}$ sendo os EMVs de $\omega$, possui distribuição Normal multivariada com média $\omega$ e matriz de variância-covariância $Var(\omega)$, os elementos da diagonal principal dessa matriz são as variâncias dos estimadores e os outros elementos, as covariâncias entre eles.

Outro resultado importante diz respeito à precisão desse estimador e estabelece que a matriz de variância-covariância $Var(\omega)$ dos estimadores de máxima verossimilhança pode ser estimada pela matriz negativa inversa da esperança das segundas derivadas (Hessiana) da função de log-verossimilhança. Em situações em que é impossível ou difícil de calcular essa esperança, a matriz é obtida colocando as estimativas na hessiana da log-verossimilhança, ou seja, substituindo $\omega$ por $\widehat{\omega}$.

Casella (2010) demostra essas propriedades em detalhes, sendo assim pode-se determinar uma estimativa para $Var(\omega)$ utilizando a relação (4.8):

$$\widehat{Var(\widehat{\omega_i})} = \frac{1}{-\frac{\partial^2}{\partial {\omega_i}^2} ln\, L(\omega_i)|_{\omega_i=\widehat{\omega_i}}} \tag{4.8}$$

Utilizando-se os resultados assintóticos dos estimadores de máxima verossimilhança e a estimativa da variância (4.8), é possível a determinação de limites de confiança para os parâmetros estimados.

Na construção de intervalos de confiança, é necessária uma estimativa para o erro-padrão do estimador, que pode ser obtido por meio da raiz quadrada da variância dos estimadores. Como os parâmetros dos modelos apresentados são positivos, é uma prática comum utilizar a transformação logarítmica do parâmetro para obter o intervalo de confiança e, em seguida, retornar para o valor do parâmetro sem o logaritmo (MEEKER; ESCOBAR, 1998). Dessa forma, um intervalo de confiança para o vetor de parâmetros $\omega$ com $\gamma$ de coeficiente de confiança é dado por:

$$LIMITE\ INFERIOR\ (\omega_i, \gamma_i) = \frac{\widehat{\omega_i}}{exp\left(z_{1-\gamma_i/2}.\ \sqrt{Var(\widehat{\omega_i})} \Big/ \widehat{\omega_i}\right)} \tag{4.9}$$

$$LIMITE\ SUPERIOR\ (\omega_i, \gamma_i) = \widehat{\omega_i}\ .\ exp\left( z_{1-\gamma_i/2}.\ \sqrt{Var(\widehat{\omega_i})} \Big/ \widehat{\omega_i} \right) \tag{4.10}$$

Em que $z_{1-\gamma}$ é o percentil 1-$\gamma$ da distribuição Normal padrão.

A construção do intervalo de confiança envolve uma matriz com alta complexidade matemática para cálculo da hessiana. Em decorrência disso, a hessiana foi calculada numericamente pelo método de diferenças finitas utilizando a função *fdHess()* implantada na linguagem de programação R (www.R-project.org, v.3.1.2).

Assim, após estimar a matriz de variância-covariância dos parâmetros e com base nas equações dos limites inferior e superior definidas anteriormente, foram construídos os intervalos de confiança para os parâmetros dos modelos propostos.

Também é possível calcular os limites de confiança para as funções do fator de melhoria variável (*a(s)* e *b(s))*. Como essas funções são dependentes do parâmetro Theta, deve-se utilizar o método Delta para o cálculo do intervalo. Esse método estimatima a variância da função de fator de melhoria para uma dada severidade *s*, utilizando uma expansão em série de Taylor e a variância do parâmetro Theta. Detalhes deste método podem ser vistos em Meeker e Escobar (1998)

## 4.4 **Avaliação das propriedades assintóticas dos estimadores dos modelos propostos**

Na construção de intervalos de confiança e testes de hipóteses de parâmetros realizadas por meio do método de máxima verossimilhança pode ser considerada a aproximação assintótica desses estimadores. Segundo a teoria assintótica, para um tamanho de amostra suficientemente grande, os estimadores de máxima verossimilhança têm distribuições aproximadas normais com variâncias dadas pelos elementos da diagonal da inversa da matriz de informação observada. Utilizando-se essa proposição, é possível a construção de intervalos de confiança e o desenvolvimento de testes de hipóteses para os parâmetros de um modelo estatístico.

O conceito de amostra suficiente grande pode ser relativo à situação que está se analisando e as limitações de disponibilidade de dados de um sistema reparável, visto que se procura construir sistemas que sejam confiáveis e tenham poucas falhas.

Caso a suposição de normalidade não seja válida, os coeficientes de confiança utilizados nos intervalos não serão realistas e os níveis de significância adotados nos testes de hipóteses não serão adequados para tomada de decisão. Além disso, pode haver vieses (diferença da estimativa com o verdadeiro valor do parâmetro) significativos dos estimadores que ficam mais evidentes para situações com poucos dados. Casella e Berger (2010) abordam em detalhes as questões assintóticas e vieses dos estimadores de máxima verossimilhança.

Com a finalidade de avaliar o comportamento assintótico e a ordem de grandeza do viés dos estimadores dos modelos propostos, foi realizado um estudo de simulação para um mapeamento da magnitude dos vieses e avaliação do coeficiente de confiança dos estimadores propostos para os modelos de fator de melhoria deste trabalho.

### 4.4.1 **Processo de simulação**

O processo de simulação para avaliar o viés dos parâmetros $\omega$ de um determinado modelo e o coeficiente de confiança $\gamma$ adotado será realizado executando os seguintes passos:

1. definir valor dos parâmetros $\omega$ (verdadeiro valor dos parâmetros) do modelo selecionado para avaliação;

2. definir o número de ciclos de MP, número de sistemas, tempos de preventivas e severidades de MP.

   **Observação:** o tamanho da amostra para um sistema reparável com tempo de manutenção preventiva predefinidos está relacionado à quantidade de ciclos de manutenção e número de sistemas em análise. Portanto, será simulado o processo descrito acima para 2, 3, 4 e 5 ciclos e considerando 1, 5, 10, 20, 30 sistemas observados;

3. Simular valores de tempos de falha do sistema(s) $T_1$, $T_2$,... com base nas definições dos itens 1 e 2;

   **Observação**: utilizando as propriedades de um PPNH, Meeker e Escobar (1998) apresentam o processo de simulação dos tempos de falha de um sistema reparável que segue um PPNH. Suponha $U_k$ um valor aleatório de uma distribuição uniforme entre zero e um. A solução geral pode ser expressa como:

$$T_j = \mu^{-1}\left[-\sum_{k=1}^{j} ln(U_k)\right], \quad j = 1, \ldots \tag{4.11}$$

Algumas vezes é mais conveniente expressar a solução geral da seguinte forma:

$$\mu(T_j) + \sum_{k=1}^{j} ln(U_k) = 0, \quad j = 1, \ldots \tag{4.12}$$

Em que o valor simulado $T_j$ é a solução da equação (4.12).

O processo de simulação dos tempos de falha utilizado neste estudo é similar ao descrito anteriormente, porém deve-se considerar as adaptações necessárias na função valor médio que utilizam os modelos propostos e avaliar em que ciclo de manutenções preventivas ocorre cada tempo de falha simulado.

4. Estimar os parâmetros $\widehat{\omega}_i$ com base nos dados simulados $T_1$, $T_2$,... utilizando a metodologia proposta nesta tese;

5. Calcular os intervalos de confiança dos parâmetros com um determinado coeficiente de confiança $\gamma$. Verifica se o verdadeiro valor do parâmetro está contido no intervalo construído, se sim $NC_i$= 1, caso contrário $NC_i$ =0;

6. Repetem-se os passos de 2 a 5 *n* vezes (*i=1,...,n*). No caso deste estudo *n* será igual a 1000 simulações.

Após a execução desse processo, calculam-se as seguintes medidas para cada parâmetro *j*:

$$\overline{\widehat{\omega_j}} = \sum_{i=1}^{n} \frac{\widehat{\omega}_{j\,i}}{n} \tag{4.13}$$

$$\overline{\gamma_j} = \sum_{i=1}^{n} \frac{NC_{j\,i}}{n} \tag{4.14}$$

Como o viés é uma medida relacionada à ordem de grandeza dos valores dos parâmetros, optou-se por avaliar duas métricas relativas para cada parâmetro: o erro relativo do viés que visa mensurar o percentual de diferença da estimativa e do verdadeiro valor do parâmetro; e o erro relativo do nível de confiança que mede o percentual da diferença do nível de confiança utilizado na construção dos intervalos dos parâmetros e do nível de confiança observado na simulação.

A formulação matemática de ambas as métricas é:

$$ERRO\ RELATIVO\ DO\ VIÉS(\omega_j) = abs\left(\frac{\omega_j - \overline{\widehat{\omega_j}}}{\omega_j}\right) \tag{4.15}$$

$$ERRO\ RELATIVO\ DO\ NÍVEL\ DE\ CONFIANÇA(\omega_j, \gamma_j) = abs\left(\frac{\gamma_j - \overline{\gamma_j}}{\gamma_j}\right) \tag{4.16}$$

Dessa forma é possível avaliar o comportamento do erro relativo de viés e erro relativo do nível de confiança de um determinado modelo à medida que se aumenta o número de sistemas e ciclos de MP acompanhados. O processo de simulação foi desenvolvido e realizado na linguagem de programação R (www.R-project.org, v.3.1.2).

### 4.4.2 **Resultados da simulação**

A simulação conforme processo descrito na seção 4.4.1 foi realizada para cada um dos oito modelos propostos e para o modelo supondo a manutenção preventiva perfeita.

Foram selecionados como valores dos parâmetros lambda igual a 100 e beta igual a 2 por representarem um modelo com intensidade de falha crescente e, desta forma, o sistema reparável estaria sujeito a necessidade de manutenção preventiva. Para os modelos relacionados ao fator de melhoria, os parâmetros foram escolhidos para representar uma eficiência média de 60% da MP no tempo de vida do sistema. A eficência média de 60% é obtida pela alteração dos parâmetros *s* e theta para os modelos de fator de melhoria variável.

Para cada modelo, foram simulados diferentes ciclos, considerando 2, 3, 4 e 5 ciclos de manutenção e diferentes quantidades de sistemas observados, considerando 1, 5, 10, 20, 30 sistemas. Para avaliação do erro relativo do nível de confiança, foi utilizado coeficiente de confiança de 95%. Os resultados estão apresentados nas tabelas 4.1 a 4.9.

Tabela 4.1 – erros relativos da simulação do Modelo PPNH considerando manutenção preventiva perfeita e parâmetros Lambda = 100 e Beta = 2

| | | Erro relativo do Viés | | Erro relativo do Nível de Confiança | |
|---|---|---|---|---|---|
| **Número de sistemas** | **Ciclos** | **Lambda** | **Beta** | **Lambda** | **Beta** |
| 1 | 2 | 6,8% | 13,7% | 6,7% | 0,4% |
| | 3 | 5,1% | 10,9% | 5,1% | 0,4% |
| | 4 | 5,5% | 9,1% | 5,8% | 0,7% |
| | 5 | 4,6% | 6,9% | 4,6% | 0,7% |
| 5 | 2 | 1,4% | 2,3% | 1,6% | 0,1% |
| | 3 | 0,9% | 1,4% | 0,9% | 0,2% |
| | 4 | 1,1% | 1,6% | 1,3% | 0,4% |
| | 5 | 0,7% | 1,3% | 0,7% | 0,2% |
| 10 | 2 | 0,7% | 1,3% | 0,1% | 1,6% |
| | 3 | 0,6% | 0,9% | 0,4% | 0,8% |
| | 4 | 0,2% | 0,7% | 0,0% | 0,7% |
| | 5 | 0,6% | 0,8% | 0,7% | 0,2% |
| 20 | 2 | 0,2% | 0,5% | 0,1% | 0,3% |
| | 3 | 0,5% | 0,4% | 0,9% | 0,5% |
| | 4 | 0,2% | 0,3% | 0,8% | 0,8% |
| | 5 | 0,3% | 0,3% | 0,8% | 1,2% |
| 30 | 2 | 0,1% | 0,2% | 0,3% | 0,1% |
| | 3 | 0,2% | 0,4% | 0,8% | 1,4% |
| | 4 | 0,1% | 0,2% | 0,0% | 0,1% |
| | 5 | 0,2% | 0,3% | 0,2% | 0,6% |

Fonte: elaborada pelo autor

As estimativas dos parâmetros Lambda e Beta apresentam maior vício quando observado, apenas, um sistema e poucos ciclos de MP. O mesmo ocorre com relação ao nível de confiança, que apresenta maior erro relativo. À medida que aumenta a quantidade de sistemas observados, o viés diminui, bem como o erro relativo ao nível de confiança, demonstrando a característica assintótica do estimador desse modelo.

Tabela 4.2 – erros relativos da simulação do Modelo PPNH considerando fator de melhoria pelo método redução de idade (modelo RI) e parâmetros Lambda = 100, Beta = 2 e *a* = 0,60

| | | Erro relativo do Viés | | | Erro relativo do Nível de Confiança | | |
|---|---|---|---|---|---|---|---|
| **Número de sistemas** | **Ciclos** | **Lambda** | **Beta** | ***a*** | **Lambda** | **Beta** | ***a*** |
| 1 | 2 | 13,7% | 19,3% | 10,2% | 3,7% | 3,1% | 0,5% |
| | 3 | 7,4% | 9,2% | 4,5% | 1,8% | 0,5% | 6,2% |
| | 4 | 7,0% | 6,3% | 6,1% | 0,1% | 0,7% | 10,6% |
| | 5 | 5,3% | 4,1% | 9,3% | 0,5% | 1,6% | 10,1% |
| 5 | 2 | 2,7% | 2,9% | 5,8% | 1,4% | 0,4% | 2,7% |
| | 3 | 1,4% | 1,5% | 4,2% | 0,9% | 0,2% | 2,3% |
| | 4 | 1,0% | 0,4% | 5,5% | 0,4% | 0,1% | 2,9% |
| | 5 | 1,4% | 0,2% | 5,9% | 1,4% | 0,0% | 3,3% |
| 10 | 2 | 1,8% | 1,5% | 3,6% | 0,7% | 0,1% | 2,0% |
| | 3 | 0,4% | 0,2% | 3,2% | 0,5% | 0,7% | 1,8% |
| | 4 | 0,6% | 0,3% | 2,4% | 0,7% | 1,4% | 2,2% |
| | 5 | 0,7% | 0,3% | 2,3% | 0,0% | 1,4% | 1,2% |
| 20 | 2 | 0,9% | 0,6% | 2,7% | 0,2% | 0,3% | 1,1% |
| | 3 | 0,4% | 0,1% | 2,1% | 0,5% | 0,8% | 0,9% |
| | 4 | 0,3% | 0,2% | 1,3% | 0,2% | 0,4% | 1,4% |
| | 5 | 0,2% | 0,0% | 1,2% | 0,9% | 0,8% | 0,3% |
| 30 | 2 | 0,2% | 0,3% | 1,2% | 1,4% | 0,7% | 0,7% |
| | 3 | 0,4% | 0,0% | 1,7% | 0,8% | 0,2% | 0,3% |
| | 4 | 0,3% | 0,2% | 0,8% | 0,5% | 0,6% | 1,8% |
| | 5 | 0,1% | 0,0% | 0,3% | 0,5% | 0,6% | 1,4% |

Fonte: elaborada pelo autor

Para o modelo considerando fator de melhoria pelo método redução de idade, nota-se o efeito assintótico do estimador com maior erro relativo do vício quando observado, apenas, um sistema e poucos ciclos de MP.

Tabela 4.3 – erros relativos da simulação do Modelo PPNH considerando fator de melhoria pelo método redução de intensidade de falha (modelo RF) e parâmetros Lambda = 100, Beta = 2 e *b* = 0,70

| | | Erro relativo do Viés | | | Erro relativo do Nível de Confiança | | |
|---|---|---|---|---|---|---|---|
| **Número de sistemas** | **Ciclos** | **Lambda** | **Beta** | ***b*** | **Lambda** | **Beta** | ***b*** |
| 1 | 2 | 10,0% | 28,7% | 1,5% | 8,4% | 2,9% | 21,6% |
| | 3 | 11,0% | 24,8% | 1,4% | 9,4% | 2,4% | 17,3% |
| | 4 | 9,6% | 22,4% | 2,0% | 8,9% | 4,3% | 15,7% |
| | 5 | 14,3% | 23,7% | 2,0% | 9,8% | 4,1% | 12,4% |
| 5 | 2 | 3,5% | 4,5% | 3,6% | 3,8% | 0,3% | 7,1% |
| | 3 | 2,5% | 4,5% | 0,1% | 1,8% | 0,4% | 4,4% |
| | 4 | 2,8% | 4,5% | 0,6% | 2,3% | 1,2% | 4,1% |
| | 5 | 2,3% | 3,6% | 0,4% | 2,6% | 0,9% | 2,9% |
| 10 | 2 | 1,3% | 1,9% | 1,6% | 1,3% | 0,3% | 4,5% |
| | 3 | 1,3% | 2,3% | 0,3% | 1,8% | 0,8% | 2,1% |
| | 4 | 1,6% | 2,3% | 0,3% | 2,2% | 0,1% | 1,4% |
| | 5 | 1,5% | 2,4% | 0,5% | 2,3% | 2,1% | 4,0% |
| 20 | 2 | 1,2% | 1,5% | 0,6% | 1,4% | 0,5% | 3,1% |
| | 3 | 0,8% | 0,9% | 0,3% | 2,0% | 1,1% | 0,9% |
| | 4 | 0,7% | 0,9% | 0,0% | 1,7% | 0,1% | 0,4% |
| | 5 | 0,9% | 0,9% | 0,0% | 0,4% | 0,0% | 0,3% |
| 30 | 2 | 0,2% | 0,4% | 0,8% | 0,2% | 1,1% | 1,8% |
| | 3 | 0,5% | 0,7% | 0,1% | 0,1% | 0,2% | 1,7% |
| | 4 | 0,3% | 0,5% | 0,1% | 1,8% | 0,1% | 1,1% |
| | 5 | 0,1% | 0,0% | 0,3% | 0,6% | 0,4% | 0,8% |

Fonte: elaborada pelo autor

Para o modelo, considerando-se fator de melhoria pelo método redução de intensidade de falha, nota-se o efeito assintótico do estimador, porém um erro relativo do vício do parâmetro Beta e do nível de confiança do parâmetro *b* maior quando comparado ao modelo anterior, analisado na Tabela 4.2, quando observa-se apenas um sistema.

Tabela 4.4 – erros relativos da simulação do Modelo PPNH considerando fator de melhoria pelo método redução de idade com fator de melhoria variável utilizando relação linear (modelo RIL) e parâmetros Lambda = 100, Beta = 2 e Theta = 1

| | | Erro relativo do Viés | | | Erro relativo do Nível de Confiança | | |
|---|---|---|---|---|---|---|---|
| **Número de sistemas** | **Ciclos** | **Lambda** | **Beta** | ***Theta a - Linear*** | **Lambda** | **Beta** | ***Theta a - Linear*** |
| 1 | 2 | 14,5% | 10,8% | 44,5% | 4,0% | 0,7% | 5,3% |
| | 3 | 15,0% | 11,5% | 15,3% | 6,1% | 1,8% | 5,3% |
| | 4 | 14,2% | 8,2% | 13,8% | 6,2% | 1,7% | 5,3% |
| | 5 | 17,8% | 11,1% | 8,7% | 5,8% | 2,3% | 4,8% |
| 5 | 2 | 3,8% | 1,5% | 24,8% | 2,2% | 1,3% | 5,3% |
| | 3 | 2,6% | 1,9% | 2,8% | 0,5% | 0,6% | 3,5% |
| | 4 | 4,2% | 2,2% | 3,2% | 2,9% | 1,3% | 5,1% |
| | 5 | 6,9% | 4,2% | 2,7% | 1,2% | 1,2% | 3,6% |
| 10 | 2 | 1,2% | 0,3% | 18,1% | 0,6% | 1,1% | 5,3% |
| | 3 | 0,4% | 0,3% | 0,3% | 0,5% | 0,7% | 0,6% |
| | 4 | 1,5% | 0,8% | 1,6% | 0,7% | 0,0% | 5,1% |
| | 5 | 3,7% | 2,0% | 1,7% | 1,7% | 0,0% | 2,8% |
| 20 | 2 | 0,2% | 0,8% | 11,8% | 0,8% | 1,3% | 5,3% |
| | 3 | 0,0% | 0,1% | 0,2% | 0,2% | 0,0% | 0,3% |
| | 4 | 0,3% | 0,2% | 0,5% | 0,1% | 1,1% | 4,0% |
| | 5 | 2,3% | 1,3% | 0,9% | 0,6% | 0,4% | 3,2% |
| 30 | 2 | 0,7% | 0,0% | 9,1% | 0,5% | 0,5% | 5,1% |
| | 3 | 0,0% | 0,1% | 0,1% | 0,3% | 0,9% | 0,7% |
| | 4 | 0,0% | 0,0% | 0,2% | 0,8% | 0,1% | 2,7% |
| | 5 | 1,4% | 0,8% | 0,6% | 0,8% | 0,4% | 2,9% |

Fonte: elaborada pelo autor

Para o modelo RIL, nota-se o efeito assintótico do estimador, porém um erro relativo do vício maior para parâmetro Theta quando observa-se, apenas, um sistema para obtenção dos dados.

Tabela 4.5 – erros relativos da simulação do Modelo PPNH considerando fator de melhoria pelo método redução de idade com fator de melhoria variável utilizando relação exponencial (modelo RIE) e parâmetros Lambda = 100, Beta = 2 e Theta = 2

| | | Erro relativo do Viés | | | Erro relativo do Nível de Confiança | | |
|---|---|---|---|---|---|---|---|
| **Número de sistemas** | **Ciclos** | **Lambda** | **Beta** | ***Theta a - Exponencial*** | **Lambda** | **Beta** | ***Theta a - Exponencial*** |
| 1 | 2 | 11,0% | 17,5% | 194,4% | 4,9% | 3,9% | 5,1% |
| | 3 | 8,7% | 9,6% | 136,8% | 4,5% | 1,3% | 5,1% |
| | 4 | 7,8% | 7,2% | 62,6% | 3,4% | 1,3% | 4,5% |
| | 5 | 3,9% | 3,3% | 59,1% | 4,8% | 1,8% | 4,4% |
| 5 | 2 | 2,0% | 2,3% | 18,4% | 1,8% | 0,9% | 2,8% |
| | 3 | 1,5% | 1,1% | 7,0% | 0,2% | 0,2% | 4,1% |
| | 4 | 0,9% | 0,7% | 4,7% | 0,3% | 0,7% | 3,5% |
| | 5 | 0,8% | 0,6% | 0,4% | 0,6% | 0,4% | 4,1% |
| 10 | 2 | 0,8% | 1,1% | 6,1% | 0,8% | 0,4% | 0,3% |
| | 3 | 0,6% | 0,6% | 5,9% | 1,8% | 0,4% | 2,5% |
| | 4 | 0,5% | 0,5% | 1,5% | 0,1% | 0,0% | 2,6% |
| | 5 | 0,4% | 0,5% | 0,9% | 0,5% | 0,2% | 2,7% |
| 20 | 2 | 0,0% | 0,1% | 1,5% | 0,0% | 1,3% | 0,4% |
| | 3 | 0,1% | 0,0% | 0,7% | 0,0% | 0,2% | 1,3% |
| | 4 | 0,6% | 0,6% | 0,7% | 0,2% | 0,0% | 0,5% |
| | 5 | 0,5% | 0,2% | 0,5% | 0,1% | 0,4% | 1,4% |
| 30 | 2 | 0,7% | 0,6% | 0,8% | 1,7% | 0,9% | 0,6% |
| | 3 | 0,1% | 0,1% | 0,9% | 0,0% | 0,5% | 0,5% |
| | 4 | 0,1% | 0,0% | 0,3% | 0,2% | 0,2% | 1,6% |
| | 5 | 0,1% | 0,0% | 0,5% | 0,4% | 0,9% | 1,4% |

Fonte: elaborada pelo autor

Esse modelo apresenta um erro relativo do viés significativamente alto para o parâmetro Theta quando se observa um sistema.

Tabela 4.6 – erros relativos da simulação do Modelo PPNH considerando fator de melhoria pelo método redução de idade com fator de melhoria variável utilizando relação potência (modelo RIP) e parâmetros Lambda = 100, Beta = 2 e Theta = 4

| | | Erro relativo do Viés | | | Erro relativo do Nível de Confiança | | |
|---|---|---|---|---|---|---|---|
| **Número de sistemas** | **Ciclos** | **Lambda** | **Beta** | ***Theta a - Potência*** | **Lambda** | **Beta** | ***Theta a - Potência*** |
| 1 | 2 | 16,0% | 21,8% | 105,0% | 4,2% | 4,1% | 9,8% |
| | 3 | 9,0% | 9,9% | 162,7% | 3,8% | 2,3% | 1,6% |
| | 4 | 4,8% | 5,4% | 176,9% | 1,9% | 1,9% | 1,7% |
| | 5 | 4,9% | 4,0% | 75,4% | 1,4% | 0,7% | 1,9% |
| 5 | 2 | 4,5% | 6,9% | 87,8% | 0,1% | 2,6% | 12,7% |
| | 3 | 0,9% | 1,3% | 36,6% | 0,1% | 0,5% | 2,4% |
| | 4 | 1,5% | 1,5% | 10,2% | 0,4% | 0,2% | 0,8% |
| | 5 | 1,8% | 1,1% | 7,2% | 0,2% | 0,2% | 0,6% |
| 10 | 2 | 1,5% | 3,3% | 89,0% | 0,0% | 1,6% | 12,2% |
| | 3 | 1,0% | 1,1% | 7,3% | 0,6% | 1,2% | 1,9% |
| | 4 | 0,8% | 0,6% | 4,7% | 0,9% | 1,2% | 2,2% |
| | 5 | 0,9% | 0,7% | 2,0% | 0,5% | 0,6% | 0,1% |
| 20 | 2 | 1,0% | 2,2% | 75,4% | 0,5% | 0,5% | 11,8% |
| | 3 | 0,5% | 0,5% | 4,4% | 0,3% | 0,4% | 2,6% |
| | 4 | 0,1% | 0,1% | 2,1% | 0,5% | 0,6% | 0,7% |
| | 5 | 0,2% | 0,1% | 1,5% | 0,7% | 0,7% | 0,1% |
| 30 | 2 | 1,0% | 1,8% | 77,3% | 0,2% | 1,1% | 9,9% |
| | 3 | 0,3% | 0,3% | 2,5% | 0,1% | 0,9% | 1,5% |
| | 4 | 0,5% | 0,4% | 0,9% | 0,1% | 1,3% | 0,3% |
| | 5 | 0,2% | 0,1% | 0,8% | 0,8% | 0,7% | 0,3% |

Fonte: elaborada pelo autor

Esse modelo apresenta um erro relativo do viés significativamente alto para o parâmetro Theta quando se observa um sistema e, também, quando se observa dois ciclos de MP para qualquer quantidade de sistemas.

Tabela 4.7 – erros relativos da simulação do Modelo PPNH considerando fator de melhoria pelo método redução de intensidade de falha com fator de melhoria variável utilizando relação linear (modelo RFL) e parâmetros Lambda = 100, Beta = 2 e Theta = 1

| | | Erro relativo do Viés | | | Erro relativo do Nível de Confiança | | |
|---|---|---|---|---|---|---|---|
| **Número de sistemas** | **Ciclos** | **Lambda** | **Beta** | ***Theta b - Linear*** | **Lambda** | **Beta** | ***Theta b - Linear*** |
| 1 | 2 | 0,4% | 4,1% | 38,2% | 2,2% | 4,2% | 5,3% |
| | 3 | 5,0% | 3,0% | 7,8% | 2,2% | 0,7% | 0,2% |
| | 4 | 8,7% | 6,5% | 1,8% | 3,2% | 0,0% | 3,3% |
| | 5 | 1,2% | 4,3% | 7,3% | 1,6% | 4,0% | 5,3% |
| 5 | 2 | 0,9% | 3,8% | 26,0% | 1,5% | 2,9% | 5,3% |
| | 3 | 0,5% | 0,6% | 1,4% | 0,6% | 1,1% | 4,1% |
| | 4 | 1,4% | 1,2% | 0,1% | 0,5% | 0,7% | 3,5% |
| | 5 | 0,9% | 1,6% | 1,5% | 0,3% | 1,4% | 5,2% |
| 10 | 2 | 0,5% | 2,8% | 17,9% | 1,2% | 3,7% | 5,3% |
| | 3 | 0,4% | 0,2% | 0,8% | 1,1% | 0,3% | 0,2% |
| | 4 | 1,4% | 1,2% | 0,2% | 0,3% | 0,5% | 1,6% |
| | 5 | 0,0% | 0,2% | 0,6% | 0,3% | 0,9% | 4,8% |
| 20 | 2 | 0,6% | 1,8% | 10,5% | 0,8% | 3,8% | 5,3% |
| | 3 | 0,4% | 0,3% | 0,2% | 0,2% | 0,8% | 1,3% |
| | 4 | 0,5% | 0,4% | 0,2% | 0,3% | 0,5% | 2,1% |
| | 5 | 0,1% | 0,1% | 0,1% | 0,5% | 1,8% | 2,5% |
| 30 | 2 | 0,1% | 0,7% | 6,6% | 0,5% | 2,1% | 5,3% |
| | 3 | 0,5% | 0,5% | 0,2% | 0,4% | 0,1% | 0,0% |
| | 4 | 0,2% | 0,2% | 0,1% | 0,0% | 0,4% | 0,6% |
| | 5 | 0,5% | 0,6% | 0,2% | 0,2% | 1,2% | 0,1% |

Fonte: elaborada pelo autor

Para o modelo RFL, nota-se o efeito assintótico do estimador, ressaltando-se um erro maior quando observa-se dois ciclos de manutenção.

Tabela 4.8 – erros relativos da simulação do Modelo PPNH considerando fator de melhoria pelo método redução de intensidade de falha com fator de melhoria variável utilizando relação exponencial (modelo RFE) e parâmetros Lambda = 100, Beta = 2 e Theta = 2

| | | Erro relativo do Viés | | | Erro relativo do Nível de Confiança | | |
|---|---|---|---|---|---|---|---|
| **Número de sistemas** | **Ciclos** | **Lambda** | **Beta** | ***Theta b - Exponencial*** | **Lambda** | **Beta** | ***Theta b - Exponencial*** |
| 1 | 2 | 52,0% | 21,2% | 17,7% | 7,6% | 3,4% | 27,8% |
| | 3 | 12,7% | 19,3% | 28,1% | 8,2% | 2,9% | 8,2% |
| | 4 | 11,5% | 18,6% | 24,0% | 7,8% | 3,7% | 4,6% |
| | 5 | 14,2% | 22,6% | 30,6% | 9,9% | 4,6% | 5,5% |
| 5 | 2 | 2,5% | 4,1% | 6,2% | 3,2% | 0,5% | 5,7% |
| | 3 | 1,0% | 2,5% | 4,8% | 0,9% | 0,8% | 1,3% |
| | 4 | 2,4% | 3,5% | 4,4% | 2,8% | 1,6% | 1,4% |
| | 5 | 3,9% | 4,3% | 4,5% | 2,1% | 1,2% | 1,1% |
| 10 | 2 | 1,6% | 1,6% | 1,9% | 0,1% | 1,2% | 2,3% |
| | 3 | 0,7% | 1,1% | 1,8% | 0,9% | 0,1% | 0,2% |
| | 4 | 1,1% | 1,4% | 1,8% | 1,2% | 0,1% | 0,2% |
| | 5 | 1,2% | 1,5% | 1,9% | 0,4% | 0,0% | 0,8% |
| 20 | 2 | 0,7% | 0,9% | 0,5% | 0,6% | 0,4% | 0,4% |
| | 3 | 0,9% | 1,0% | 1,3% | 0,7% | 0,0% | 0,8% |
| | 4 | 0,8% | 0,9% | 1,0% | 1,5% | 0,2% | 0,7% |
| | 5 | 1,2% | 1,0% | 1,0% | 1,5% | 0,7% | 0,9% |
| 30 | 2 | 0,5% | 1,0% | 1,4% | 1,2% | 0,5% | 0,8% |
| | 3 | 0,1% | 0,2% | 0,6% | 0,4% | 1,3% | 0,7% |
| | 4 | 0,7% | 0,6% | 0,3% | 0,5% | 1,6% | 0,0% |
| | 5 | 0,1% | 0,0% | 0,2% | 0,5% | 0,4% | 0,8% |

Fonte: elaborada pelo autor

Para o modelo RFE, nota-se o efeito assintótico do estimador com erros de viés e de nível de confiança maiores quando da observação de um sistema em análise.

Tabela 4.9 – erros relativos da simulação do Modelo PPNH considerando fator de melhoria pelo método redução de intensidade de falha com fator de melhoria variável utilizando relação potência (modelo RFP) e parâmetros Lambda = 100, Beta = 2 e Theta = 4

| | | Erro relativo do Viés | | | Erro relativo do Nível de Confiança | | |
|---|---|---|---|---|---|---|---|
| **Número de sistemas** | **Ciclos** | **Lambda** | **Beta** | ***Theta b - Potência*** | **Lambda** | **Beta** | ***Theta b - Potência*** |
| 1 | 2 | 18,7% | 31,3% | 77,8% | 9,7% | 4,6% | 11,1% |
| | 3 | 9,6% | 12,5% | 106,0% | 5,7% | 1,6% | 0,6% |
| | 4 | 8,3% | 8,1% | 37,3% | 3,9% | 0,2% | 1,3% |
| | 5 | 7,5% | 7,4% | 4,0% | 3,8% | 0,7% | 1,9% |
| 5 | 2 | 5,1% | 9,7% | 84,0% | 1,3% | 1,9% | 12,7% |
| | 3 | 2,2% | 2,5% | 15,4% | 1,2% | 0,1% | 0,3% |
| | 4 | 0,9% | 1,1% | 2,9% | 0,5% | 0,5% | 1,5% |
| | 5 | 2,1% | 1,7% | 0,2% | 0,9% | 0,5% | 1,6% |
| 10 | 2 | 3,0% | 6,2% | 70,3% | 2,4% | 2,3% | 12,7% |
| | 3 | 0,6% | 0,7% | 7,7% | 0,6% | 0,2% | 0,9% |
| | 4 | 0,0% | 0,2% | 2,2% | 0,3% | 0,1% | 0,9% |
| | 5 | 1,1% | 0,8% | 0,0% | 0,0% | 0,8% | 1,2% |
| 20 | 2 | 2,2% | 4,4% | 69,7% | 1,8% | 2,3% | 13,7% |
| | 3 | 0,1% | 0,3% | 2,7% | 1,1% | 0,1% | 0,8% |
| | 4 | 0,4% | 0,4% | 0,3% | 0,6% | 0,7% | 0,6% |
| | 5 | 0,7% | 0,6% | 0,1% | 1,1% | 0,6% | 0,1% |
| 30 | 2 | 1,2% | 2,5% | 88,0% | 0,6% | 0,7% | 8,0% |
| | 3 | 0,5% | 0,4% | 1,2% | 0,2% | 0,6% | 1,2% |
| | 4 | 0,7% | 0,5% | 0,2% | 0,0% | 0,1% | 0,4% |
| | 5 | 0,2% | 0,2% | 0,1% | 0,4% | 1,6% | 1,2% |

Fonte: elaborada pelo autor

Esse modelo apresenta um erro relativo do viés significativamente alto quando se observa um sistema e, também, quando se observam, apenas, dois ciclos de MP para qualquer quantidade de sistemas para o parâmetro Theta.

### 4.4.3 **Conclusões sobre o estudo de simulação**

Neste estudo de simulação, pode-se avaliar que o método para estimação dos parâmetros e construção dos intervalos de confiança baseados na normalidade assintótica dos estimadores de máxima verossimilhança revelou convergência satisfatória (menores erros relativos) à medida que os tamanhos das amostras aumentaram.

Em algumas combinações, os resultados obtidos apresentaram erros com valores altos quando comparado ao modelo tradicional de manutenção perfeita e esse resultado indica que

as inferências baseadas em procedimentos assintóticos merecem cuidado especial nesses casos, particularmente quando amostras pequenas são consideradas na análise.

Destacam-se três pontos da análise dos resultados da simulação:

– quando analisar apenas um sistema, evitar os modelos que consideram fator de melhoria variável com as relações Potência e Exponencial;

– quando acompanhar apenas dois ciclos de MP, evitar os modelos que consideram fator de melhoria variável com a relação Potência;

– o modelo considerando fator de melhoria pelo método redução de idade (modelo RI) tem os menores valores de erros para amostras pequenas, portanto mais aconselhável sua utilização nessa condição.

Em suma, este estudo permitiu avaliar questões pontuais sobre as propriedades assintóticas dos estimadores dos modelos propostos nesta tese e sugerir a utilização dos modelos em algumas situações segundo as informações de falhas disponíveis.

# 5. **MÉTODO PARA AVALIAÇÃO DO AJUSTE DOS MODELOS**

A seleção do modelo que será utilizado nas análises subsequentes ao ajuste dos parâmetros é um tópico importante na análise paramétrica de dados, pois a escolha dos modelos que apresentaram melhores ajustes possibilita maior acurácia nas previsões realizadas e por consequência melhores decisões adotadas. Portanto, o desenvolvimento de testes de hipóteses e análises gráficos que possibilitem uma avaliação da qualidade de ajuste das estimativas é um ponto fundamental para utilização dos modelos no processo decisório.

Rigdon e Basu (2000) descrevem testes estatísticos e análises gráficas para avaliação dos modelos tradicionais de Processo de Poisson considerando manutenções perfeitas. Porém esse tema é pouco explorado na literatura para os modelos com manutenções imperfeitas conforme estudo exposto na Tabela 1.1 deste trabalho. Dessa forma, fazem-se necessárias adaptações nos testes estatísticos e análises gráficas existentes para que suportem o modelo de fator de melhoria fixo e variável.

Neste trabalho, serão utilizados dois testes de hipóteses para avaliação da qualidade de ajuste, o teste da razão de verossimilhança (TRV), que permite comparar o ajuste dos modelos propostos e o teste Cramér Von Mises (CVM) que possibilita avaliar a aderência do modelo proposto aos dados. Além disso, será apresentada uma proposta de construção gráfica para avaliação da qualidade do ajuste do modelo aos dados baseado no artigo de Toledo et al. (2015). No final do capitulo, será proposta uma metodologia para utilização conjunta dos testes e análise gráfica que possibilite selecionar o modelo mais adequado ao conjunto de dados.

## 5.1 **Teste de razão de verossimilhanças para comparar modelos**

O teste estatístico da razão de verossimilhança (TRV) é um teste estatístico de qualidade do ajuste entre dois modelos hierarquicamente encaixados (COLOSIMO; GIOLO, 2006). Pode, principalmente, ser utilizado para verificar a significância em se adicionar outros parâmetros em um modelo.

Suponha $\omega$ ser um parâmetro adicionado em um determinado modelo e deseja-se testar a hipótese $H_0$: $\omega = 0$, contra a hipótese alternativa $H_1$: $\omega \neq 0$. Este teste é baseado na função de verossimilhança e envolve a comparação dos valores dos logaritmos da função de verossimilhança nos pontos máximos obtidos para dois modelos:

(1) modelo com o parâmetro adicionado, considerando $\omega \neq 0$ ($\ln L(\widehat{\omega})$);

(2) modelo sem o parâmetro adicional, considerando manutenção preventiva sem efeito no processo ($\ln L(\widehat{\omega}_0)$), sob $H_0$: $\omega = 0$.

A estatística do teste é:

$$TRV = -2[\ln L(\widehat{\omega}_0) - \ln L(\widehat{\omega})] \tag{5.1}$$

Utilizando a função verossimilhança definida na seção 4.1 deste trabalho, a estatística do TRV para os modelos relacionados à redução de idade é dada por (5.2):

$$\begin{aligned} TRV = -2\Bigg\{ ln\Bigg[\prod_{k=1}^{m}\Bigg[\prod_{i=1}^{n_k}\frac{1}{{\lambda_0}^{\beta_0}}.\beta_0.{t_{ki}}^{(\beta_0-1)}.exp\left(-\frac{1}{{\lambda_0}^{\beta_0}}.{T_k}^{\beta_0}\right)\Bigg]\Bigg] - \\ - ln\Bigg[\prod_{k=1}^{m}\Bigg[\prod_{j=1}^{c_k}\Bigg[\prod_{i=1}^{n_{kj}}\frac{1}{\lambda^{\beta}}.\beta.\left(t_{kji}-\widehat{\omega}.T_{k\ j-1}\right)^{(\beta-1)}\Bigg]. \\ .exp\Bigg[-\sum_{j=1}^{c_k}\left(\frac{1}{\lambda^{\beta}}.\left(T_{k\ j}-\widehat{\omega}.T_{k\ j-1}\right)^{\beta}-\frac{1}{\lambda^{\beta}}.\left((1-\widehat{\omega})T_{k\ j-1}\right)^{\beta}\right)\Bigg]\Bigg]\Bigg]\Bigg\} \end{aligned} \tag{5.2}$$

A estatística do TRV para os modelos relacionados à redução de intensidade de falha, considerando a verossimilhança definida em 4.1, é dada por (5.3):

$$\begin{aligned} TRV = -2\Bigg\{ ln\Bigg[\prod_{k=1}^{m}\Bigg[\prod_{i=1}^{n_k}\frac{1}{{\lambda_0}^{\beta_0}}.\beta_0.{t_{ki}}^{(\beta_0-1)}.exp\left(-\frac{1}{{\lambda_0}^{\beta_0}}.{T_k}^{\beta_0}\right)\Bigg]\Bigg] - \\ - ln\Bigg[\prod_{k=1}^{m}\Bigg[\prod_{j=1}^{c_k}\Bigg[\prod_{i=1}^{n_{kj}}(1-\widehat{\omega})^{j-1}.\frac{1}{\lambda^{\beta}}.\beta.\left(t_{kji}\right)^{(\beta-1)}\Bigg]. \\ .exp\Bigg[-\sum_{j=1}^{c_k}(1-\widehat{\omega})^{j-1}.\left(\frac{1}{\lambda^{\beta}}.\left(T_{k\ j}\right)^{\beta}-\frac{1}{\lambda^{\beta}}.\left(T_{k\ j-1}\right)^{\beta}\right)\Bigg]\Bigg]\Bigg]\Bigg\} \end{aligned} \tag{5.3}$$

A estatística desse teste segue uma distribuição qui-quadrado ($\chi^2$) com $p$ graus de liberdade, em que $p$ é o número adicional de parâmetros associado a modelagem dos fatores de melhoria. No caso especifico desta tese, o TRV será aplicado para avaliar a adição do

parâmetro $a$ nos modelos de redução de idade, do parâmetro $b$ nos modelos de redução de intensidade e do parâmetro $\theta$ nos modelos de fator de melhoria variável. Dessa forma, $p$ será igual a um, referente ao parâmetro adicionado.

Definindo o valor de $\alpha$ como o nível de significância do teste, a hipótese nula é rejeitada se $TRV > \chi^2_{p,\alpha}$, em que $\chi^2_{p,\alpha}$ é o valor do percentil $\alpha$ da distribuição qui-quadrado com $p$ graus de liberdade. Portanto, a significância estatística da inclusão do parâmetro do modelo de fator de melhoria pode ser avaliada utilizando o TRV.

Para decisão do teste, também é possível calcular o nível descritivo ou *p-valor* do teste tomando a probabilidade da distribuição qui-quadrado com $p$ graus de liberdade da estatística do TRV. Dessa forma, quando *p-valor* é menor que o valor $\alpha$, a hipótese nula é rejeitada, ou seja, há significância estatística na inclusão do parâmetro do modelo de fator de melhoria.

## 5.2 **Teste e gráfico para avaliação da qualidade do ajuste do modelo**

O TRV verifica se há significância estatística em incluir um novo parâmetro no modelo, porém não verifica se o modelo selecionado está descrevendo adequadamente o comportamento de falha dos dados observados. Para verificar o ajuste do modelo será utilizado o teste Cramér Von Mises.

### 5.2.1 **Teste *Cramér Von Mises***

O teste estatístico Cramér Von Mises **(**CVM) é conhecido para o modelo tradicional de PPNH e apresentado em Rigdon e Basu (2000). Crow (1974) adaptou o teste CVM para múltiplos sistemas e sistemas reparáveis.

Crow (1974) fez a adaptação, apenas, para o caso em que a função intensidade de falha é a Lei da Potência. Para os casos de MP imperfeita, considerando os fatores de melhoria, há necessidade de utilizar uma generalização na estatística do teste de CVM para possibilitar testar os modelos por meio da função intensidade do processo.

A hipótese nula ($H_0$) é que a função intensidade utilizada é o modelo que representa o processo de falha e a hipótese alternativa ($H_1$) é que a função intensidade não é o modelo correto. A função intensidade considerada nas hipóteses do teste, é a mesma que deverá ser utilizada no cálculo da estatística do teste descrito a seguir.

Considere a observação de *n* falhas com reparação mínima com $0 < t_1 < t_2 < \ldots < t_n = T$. A estatística do teste de CVM generalizada é dada por:

$$CVM = \frac{1}{12M} + \sum_{i=1}^{M} \left( \frac{\mu(t_i)}{\mu(T)} - \frac{2i-1}{2M} \right)^2 \tag{5.4}$$

Se a última informação for uma falha (truncado na falha), então $M=n-1$, porém, se a última informação for uma preventiva (truncado no tempo), então $M=n$. Apesar da estatística do teste utilizar a função $\mu(t)$, esta possui relação direta com a função intensidade $u(t)$ a qual a hipótese do teste se refere.

Se houver mais de um sistema envolvido na análise que esteja sob as mesmas condições de uso, Crow (1974) propõe calcular todas as relações $\mu(t_i)/\mu(T)$ para cada sistema e depois tratá-las como um único grupo, ordenando-as da menor para a maior.

Este teste é aplicado aos PPNH considerando manutenções perfeitas, mas é adaptável para os modelos de fator de melhoria e fator de melhoria variável proposto nesta tese, pois para calcular o CVM sob os modelos propostos, basta utilizar as funções $\mu(t)$, definidas no capítulo 3, no cálculo da estatística do teste.

Uma tabela da distribuição da estatística do teste CVM é apresentada no apêndice de Rigdon e Basu (2000), de onde se pode obter o valor crítico para um determinado nível de significância $\alpha$. Então, se a estatística CVM for menor que o valor crítico extraído da tabela, a hipótese nula é aceita, indicando que os tempos de falha seguem um processo definida pela função intensidade utilizada.

A decisão do teste CVM também pode ser realizada pelo nível descritivo ou *p-valor* do teste, calculando a probabilidade da distribuição da estatística do teste CVM. Dessa forma, quando *p-valor* é maior que $\alpha$, a hipótese nula ($H_0$) é aceita, ou seja, a função intensidade utilizada no cálculo da estatística CVM é o modelo que representa o processo de falha.

### 5.2.2 **Gráfico para avaliação da qualidade do ajuste dos modelos**

Além de um teste estatístico para verificar o ajuste, a análise gráfica também é uma maneira interessante de análise do comportamento do modelo com os dados observados. No caso de um ajuste inadequado, a análise gráfica permite detectar pontos que podem estar

causando o mau ajuste ou problemas na função paramétrica selecionada para representar a função intensidade.

Toledo et al. (2015) propõem um gráfico para avaliação de ajuste de modelos para reparo imperfeito. A ideia geral é estimar a função valor médio (FVM) do processo para cada modelo e comparar com a estimação não paramétrica desta função proposta em Nelson (2003).

Os passos a seguir descrevem a construção do gráfico proposto:

1. calcular os EMV de um modelo específico com base nos dados de falhas dos *k* sistemas disponíveis;
2. utilizando o EMV obtido no passo anterior e os dados de falha do *k*-ésimo sistema, calcular a estimativa da função valor médio $\mu_k(t)$ do k-ésimo sistema. Calcular para os *k* sistemas (*k*=1,...,*m*). Neste passo deverá utilizar a função paramétrica $\mu_k(t)$ que se deseja avaliar o ajuste;
3. a função valor médio estimada para o processo no tempo *t* é calculado como
$$\widehat{FVM}(t) = \sum_{k=1}^{m} \mu_k(t)/m \tag{5.5}$$
4. a partir desse passo, será calculada a função valor médio não paramétrica. Ordenar os tempos de falha e de MP do menor para o maior, *i* tempos (*i*=1,...,*n*). Se o tempo de falha e de MP for o mesmo, a falha deve vir primeiro.
5. para cada tempo, calcular o número, $r_i$, de unidades que passaram da idade $t_i$ :
   – $r_1$ é igual a *m*.
   – Se $t_i$ é um tempo de falha, então $r_i = r_{i-1}$.
   – Se $t_i$ é um tempo de MP (censura), então $r_i = r_{i-1}-1$.
6. para cada tempo de falha, $t_i$, calcular a função valor médio não paramétrica da seguinte forma:
$$FVM_{np}(t_i) = 1/r_i + FVM_{np}(t_{i-1}) \tag{5.6}$$
em que $FVM_{np}(t_1) = 1/r_1$.
7. construir o gráfico $\widehat{FVM}(t)$ x $FVM_{np}(t_i)$.

Espera-se que o melhor ajuste faça com que a FVM estimada fique mais próxima da FVM não paramétrica. Portanto, ao comparar os pontos estimados por essas duas funções, em um gráfico, o modelo que apresenta melhor ajuste será cujos pontos se aproximam de uma reta.

## 5.3 **Metodologia para avaliação do melhor modelo**

Considerando-se que, para um determinado conjunto de dados, diferentes modelos podem levar a diferentes estimativas de confiabilidade, é importante escolher o modelo que melhor se ajusta aos dados disponíveis para que as previsões estejam dentro da realidade observada.

Existem diversos testes e gráficos para avaliar o ajuste de modelos para sistemas reparáveis conforme Rigdon e Basu (2000). Anteriormente, foram apresentados os testes e gráfico para PPNH com função de intensidade de potência, fazendo adaptações para os modelos propostos de fator de melhoria na MP.

A seguir será apresentada uma proposta de metodologia para escolha do modelo mais adequando utilizando os testes:

1. calcula-se TRV para todos os modelos propostos;
2. escolhe-se o modelo com maior valor TRV (ou menor *p-valor*). Essa escolha garante o modelo cuja inclusão do parâmetro do fator melhoria foi a mais significativa;
3. calcula-se o teste CVM para o modelo selecionado no passo 2 para avaliar se o modelo escolhido representa adequadamente o processo em análise;
4. construir o gráfico FVM estimada pela FVM não paramétrica para avaliar o ajuste e se há algum ponto discrepante;
5. se o modelo for aprovado no item 3 e 4, utiliza-se o modelo selecionado para os cálculos das métricas de confiabilidade e otimização de MP. Caso o modelo não seja aceito nos passos 3 e 4, deve-se retornar ao passo 2 e escolher o próximo modelo ordenado pelo valor de TRV;
6. caso esse processo se repita e nenhum modelo que utiliza o fator de melhoria da MP for selecionado, o provável modelo é o PPNH tradicional com efeito de MPP ou sem efeito da MP. Detalhes de como avaliar o ajuste desse modelo pode ser visto em Rigdon e Basu (2000).

# 6. **OTIMIZAÇÃO DA PERIODICIDADE DA MP IMPERFEITA**

No cenário industrial, a política de manutenção preventiva é essencial para redução do risco de falha dos equipamentos, diminuindo as potenciais perdas e aumentando a confiabilidade do sistema. Há extensa literatura sobre estratégias para especificar tais políticas, como Nakagawa (1986), Jayabalan (1992), Gilardoni e Colosimo (2007), Bartholomew-Biggs et al. (2009), Yeh et al. (2009), Wu e Zuo (2010), Do e Berenguer (2012) e Remy et al. (2013). Esses trabalhos têm como principal foco o estudo e otimização das políticas de periodicidade da MP por meio da minimização dos custos relacionados a manutenção preventiva e corretiva, porém utilizam a suposição que a manutenção preventiva é perfeita, ou seja, traz o sistema para a condição de operação igual ao seu início de vida.

Para descrever esses métodos, considere um sistema reparável colocado em operação no tempo $T_0$=0 e que a manutenção preventiva (MP) ocorre em tempos predefinidos $T_1$, $T_2$, ..., $T_k$ ($T_0=0 < T_1 < T_2 <, ..., < T_k$) e sujeito à Manutenção Corretiva (MC) mínima em caso de falha.

Em cada ponto de MP, uma ação de custo fixo $C_{MP}$ é executada e entre as ações de MP, um reparo mínimo é executado caso ocorra uma falha e o custo esperado para cada ação de reparo é $C_{MC}$.

A função de custo de manutenção é composta por uma parte determinística associada com a MP periódica e outra parte aleatória relacionada à MC, assumindo que os custos de reparo e tempos de falha são independentes e tempos de reparo são insignificantes (GILARDONI; COLOSIMO, 2007). O custo esperado de manutenção do j-ésimo ciclo de MP é dada por 6.1.

$$C_{MP} + E\left[N\left(T_j\right)\right].C_{MC} \tag{6.1}$$

Em que o primeiro termo é o custo MP no final do intervalo e segundo termo é o número esperado de falhas que ocorrem dentro do intervalo multiplicado pelo custo de MC.
Como os custos de manutenção se acumulam ao longo do tempo, faz sentido trabalhar com o custo esperado por unidade de tempo. Assim, o custo total esperado por unidade de tempo para o j- ésimo ($j = 1, 2, 3, ...,k$) ciclo de MP irá ser determinada conforme 6.2.

$$C_{TOT}\left(T_j\right) = \frac{C_{MP} + E\left[N\left(T_j\right)\right].C_{MC}}{T_j - T_{j-1}} \tag{6.2}$$

Uma vez que em cada ponto de verificação de MP o estado do sistema não é restaurado para uma condição como novo, a função de custo deve considerar a função valor médio que utiliza os fatores de melhoria definidos anteriormente. Portanto a função custo é definida conforme equação 6.3.

$$C_{TOT}(T_j) = \frac{C_{MP} + [\mu(T_j) - \mu(T_{j-1})].C_{MC}}{T_j - T_{j-1}} \tag{6.3}$$

O efeito no $C_{TOT}(T_j)$ será dependente dos valores da periodicidade da preventiva ($T_j$) e do número de falhas esperada ($E[N(T_j)]$). Considerando-se que o tempo entre as MPs é aumentado, os custos de MP terá uma redução no custo por unidade de tempo e, por outro lado, ocorrerá aumento do custo em MC, pois, provavelmente, ocorrerão mais falhas. Se o tempo entre MPs é reduzido, os custos de MP promoverá aumento nos custo total por unidade de tempo, porém acarretará em uma redução nos custos de MC. Portanto, o tempo $T_j$ que minimizar $C_{TOT}(T_j)$ é o tempo de manutenção preventiva ótimo para o ciclo *j* de manutenção preventiva.

Para encontrar esse valor, a função $C_{TOT}(T_j)$ é derivado e igualada a zero.

$$\frac{dC_{TOT}}{dT_j}(T_j) = 0 \tag{6.4}$$

Para otimização da periodicidade preventiva considerando os modelos de fator de melhoria e fator de melhoria variável, deve-se substituir a função $\mu(T)$ pelas correspondentes definidas na seção 3.1 e 3.2 desta tese. Devido à complexidade da função $\mu(T)$ envolvida na equação, a solução analítica torna-se complexa. Portanto $T_j$, deve ser resolvido numericamente e satisfazer a equação 6.4.

Nesse caso, se a função de custo tem um ponto mínimo, este representa o tempo entre MPs que minimiza os custos de manutenção. Assim, uma vez conhecido o valor da relação $C_{MP}/C_{MC}$, utilizando os parâmetros estimados do modelo e substituindo $\mu(T)$, o tempo que minimiza o custo de manutenção entre as MPs ao longo de diferentes ciclos pode ser obtido.

Em geral os modelos de otimização da periodicidade preventiva encontrados na literatura, citada no inicio deste capítulo, consideram a MP perfeita e/ou realizada em condições constantes ao longo do tempo. Portanto há dois principais avanços neste trabalho com relação à otimização da periodicidade da manutenção preventiva.

– O primeiro é a utilização da função $\mu(t)$ com o fator de melhoria no cálculo do custo total o que permita considerar a manutenção preventiva imperfeita nos cálculos de otimização da periodicidade preventiva.

– O segundo é o uso dos modelos propostos que relaciona as diferentes condições do plano de manutenção (fator de melhoria variável) na função custo total. Dessa forma é possível determinar a otimização da periodicidade para um novo plano de manutenção, ou seja, considerando uma manutenção com atividades distintas da que foi realizada nos ciclos anteriores.

Para o modelo de fator de melhoria variável apresentado neste trabalho, o custo $C_{MP}$ varia conforme a severidade ($s$) da ação preventiva muda, pois se há alteração na severidade (mais tarefas ou mais itens substituídos) o custo da ação preventiva aumentará também. Ou seja, $C_{MP}$ é uma função da severidade da intervenção de manutenção e para cada severidade de MP haverá um $C_{MP}$ diferente que deverá ser considerado no cálculo da otimização da periodicidade.

# 7. FUNÇÃO CONFIABILIDADE

A confiabilidade pode ser definida como a probabilidade de um sistema desempenhar sua função, sem falhar, por um determinado tempo *t*. Esta métrica é importante, principalmente, para prever o comportamento de falha de sistemas e desta forma auxiliar a equipe de engenharia na tomada de decisão de reparos e planos de manutenção.

A confiabilidade pode ser mensurada quantitativamente através da relação com as funções apresentadas anteriormente. Denomina-se como $t_n$ o maior valor entre o último tempo de falha observado e o último tempo de MP ($T_c$) do ciclo *c* para um dado sistema. O interesse é conhecer o comportamento do próximo tempo de falha $t_{n+1}$ - $t_n$ dada a história até $t_n$ ($\mathcal{H}_t$). A função de confiabilidade no tempo *t* descreve o comportamento de falha do processo e definida como a seguir:

$$R(t) = P(t_{n+1} - t_n > t | \mathcal{H}_t) = P(N(t_n, t_n + t) = 0 | \mathcal{H}_t)$$
$$= exp\left[-\int_{t_n}^{t_n+t} u(w)\, dw\right] \tag{7.1}$$

Esta função expressa à probabilidade de sobrevivência ou a probabilidade de que o sistema irá funcionar, sem falhas, durante um determinado tempo maior do que o *t* após $t_n$. Para o cálculo da função confiabilidade para os modelos definidos neste trabalho deve-se substituir a função intensidade *u(t)* de cada modelo na função confiabilidade (7.1).

Para o modelo RI tem-se a seguinte função confiabilidade:

$$R(t) = exp\left[-\int_{t_n}^{t_n+t} \frac{1}{\lambda^\beta}.\beta.(w - a.T_c)^{(\beta-1)}\, dw\right]$$
$$= exp\left\{-\left[\left[\frac{1}{\lambda^\beta}.(t_n + t - a.T_c)^\beta - \frac{1}{\lambda^\beta}.\left((1-a)T_c\right)^\beta\right] - \left[\frac{1}{\lambda^\beta}.(t_n - a.T_c)^\beta - \frac{1}{\lambda^\beta}.\left((1-a)T_c\right)^\beta\right]\right]\right\}$$
$$= exp\left\{-\left[\left[\frac{1}{\lambda^\beta}.(t_n + t - a.T_c)^\beta\right] - \left[\frac{1}{\lambda^\beta}.(t_n - a.T_c)^\beta\right]\right]\right\} \tag{7.2}$$

Para o modelo RF tem-se a seguinte função confiabilidade:

$$R(t) = exp\left[-\int_{t_n}^{t_n+t}(1-b)^{c-1}.\frac{1}{\lambda^{\beta}}.\beta.w^{(\beta-1)}\,dw\right]$$

$$= exp\left\{-\left[\left[(1-b)^{c-1}.\left[\frac{1}{\lambda^{\beta}}.(t_n+t)^{\beta}-\frac{1}{\lambda^{\beta}}.{T_{c-1}}^{\beta}\right]\right]-\left[(1-b)^{c-1}.\left[\frac{1}{\lambda^{\beta}}.(t_n)^{\beta}-\frac{1}{\lambda^{\beta}}.{T_{c-1}}^{\beta}\right]\right]\right]\right\}$$

$$= exp\left\{-\left[\left[(1-b)^{c-1}.\left[\frac{1}{\lambda^{\beta}}.(t_n+t)^{\beta}\right]\right]-\left[(1-b)^{c-1}.\left[\frac{1}{\lambda^{\beta}}.(t_n)^{\beta}\right]\right]\right]\right\} \qquad (7.3)$$

Para o modelo RIL, RIE e RIP tem-se a seguinte função confiabilidade:

$$R(t) = exp\left\{-\left[\left[\frac{1}{\lambda^{\beta}}.(t_n+t-a(s).T_c)^{\beta}\right]-\left[\frac{1}{\lambda^{\beta}}.(t_n-a(s).T_c)^{\beta}\right]\right]\right\} \qquad (7.4)$$

Para definir as funções confiabilidade dos modelos RIL, RIE e RIP deve-se substituir em (7.4) a função *a(s)* pelas três formas funcionais propostas. Para função confiabilidade do modelo RIL deverá substituir *a(s)* pela relação linear $a_L(s) = s\,.\,\theta$, para do modelo RIE deverá substituir pela relação exponencial $a_E(s) = 1 - Exp(-s.\theta)$ e para do modelo RIP substituir pela relação potência $a_P(s) = s^{\,\theta}$.

Para o modelo RFL, RFE e RFP tem-se a seguinte função confiabilidade:

$$R(t) = exp\left\{-\left[\left[\left(1-b(s)\right)^{c-1}.\left[\frac{1}{\lambda^{\beta}}.(t_n+t)^{\beta}\right]\right]-\left[\left(1-b(s)\right)^{c-1}.\left[\frac{1}{\lambda^{\beta}}.(t_n)^{\beta}\right]\right]\right]\right\} \qquad (7.5)$$

Para definir as funções confiabilidade dos modelos RFL, RFE e RFP deve-se substituir em (7.5) a função *b(s)* pelas três formas funcionais propostas. Para função confiabilidade do modelo RIL deverá substituir *b(s)* pela relação linear $b_L(s) = s\,.\,\theta$, para do modelo RIE deverá substituir pela relação exponencial $b_E(s) = 1 - Exp(-s.\theta)$ e para do modelo RIP substituir pela relação potência $b_P(s) = s^{\,\theta}$.

O EMV para $\beta$, $\eta$ e $\theta$ pode ser substituído nas funções confiabilidade definidas anteriormente, possibilitando calculo analítico das previsões relativas ao futuro comportamento de confiabilidade para um determinado sistema.

Vale ressaltar que a função confiabilidade dos modelos com fator de melhoria variável depende da severidade de execução da preventiva, desta forma é possível analisar o

comportamento da confiabilidade para diferentes planos de MP e decidir a estratégia preventiva mais adequada A previsão da função confiabilidade, utilizando os modelos propostos nesta tese, ganha uma dimensão adicional, pois considera à condição de execução e atividades da manutenção preventiva no sistema em análise.

# 8. **APLICAÇÃO DO MODELO**

O problema que motivou este trabalho foi o de mensurar o efeito da manutenção preventiva na vida de um sistema juntamente com o efeito de diferentes atividades de realização da manutenção preventiva e considerar esses dois efeitos na otimização da periodicidade de realização da manutenção preventiva.

Para ilustrar a aplicação dos modelos propostos, foram simulados dados de falha de um sistema de refrigeração de uma usina nuclear com base no comportamento de falha dos dados apresentados no artigo de Shin et al. (1996). A simulação foi realizada utilizando os passos 1, 2 e 3 do processo descrito na seção 4.4.1 desta tese e utilizando os parâmetros estimados no artigo citado. Os dados simulados representam o acompanhamento de cinco sistemas reparáveis colocado em operação no tempo 0 e observado por 600 dias. A ação preventiva desses sistemas foi realizada quatro vezes no período observado nos tempos 150, 250, 500 e 600 dias.

O artigo Shin et al. (1996) não apresenta as informações de severidade das manutenções preventivas realizadas em cada ação, o que é necessário para aplicação dos modelos propostos nesta tese. Portanto faz-se necessidade a definição das severidades das três primeiras ações preventivas realizadas nos sistemas. Assim foi realizada uma análise de sensibilidade, a fim de avaliar o impacto da escolha das severidades na qualidade ajuste dos modelos e se há alteração na determinação do melhor ajuste em decorrência da variação das severidades. Ou seja, o objetivo dessa análise de sensibilidade é avaliar se a escolha das severidades neste estudo altera a conclusão da escolha dos modelos.

Para o desenvolvimento da análise de sensibilidade, foram selecionadas três condições para severidades de MP:

- a primeira condição foi de uma severidade de 0,2 na primeira ação, 0,2 para a segunda MP e 0,5 para a terceira preventiva;
- a segunda condição considerada foi uma severidade de 0,4 para a primeira MP, 0,4 na segunda ação e 0,7 para a terceira ação preventiva;
- a terceira condição foi uma severidade de 0,6 para a primeira ação, 0,6 na segunda ação e 0,8 para a terceira MP.

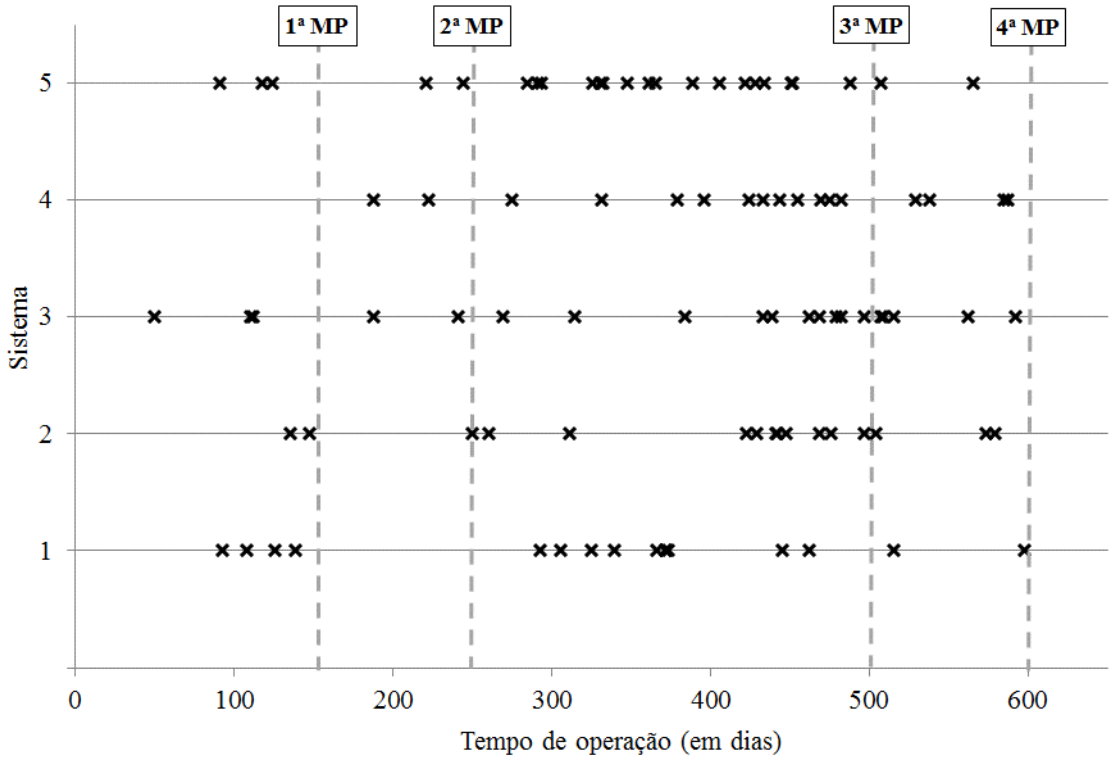
Foram ajustados os modelos propostos neste trabalho (RIL, RIE, RIP, RFL, RFE e RFP) e calculado o *p-valor* do TRV de cada modelo para cada uma das três condições de severidade. Em seguida, para cada modelo, foi calculado o desvio padrão dos *p-valores* do TRV. Se houver pouca variação nos *p-valores* (baixo desvio padrão) isto indica que a

alteração das condições não altera os *p-valores* e como consequência não altera a escolha do modelo. Desta forma, pode-se avaliar o impacto da alteração das severidades na escolha do modelo.

Verificou-se um desvio do *p-valor* para o modelo RIL de 0,0010, para o modelo RIE de 0,0004, para o modelo RIP 0,0025, para o modelo RFL de 0,0008, para o modelo RFE de 0,0005, para o modelo RIP 0,0028. Portanto, como há uma variação pequena dos *p-valores*, ou seja, abaixo de 0,01 que será o nível de significância adotado, pode-se concluir, nesse caso, que a variação nas severidades não afetará a escolha do modelo de acordo com TRV, pois não há uma variação relevante do *p-valor* que possa alterar a conclusão de seleção do modelo. Assim, será selecionado para esta análise a seguinte condição de severidade para as atividades de manutenção preventiva: 0,4 para a primeira ação, 0,4 na segunda ação e 0,7 para a terceira preventiva.

Os cinco sistemas simulados apresentaram 93 falhas no período observado entre 0 e 600 dias, todas seguidas por um reparo mínimo visando restabelecer a condição operacional do sistema. Na Figura 8.1, estão dispostas as falhas versus o tempo de operação, em que cada linha corresponde a um sistema e cada símbolo **x** representa um tempo de falha.

Figura 8.1 – tempo de falha em dias de operação para cada sistema




Fonte: elaborado pelo autor

## 8.1 **Estimativa e seleção dos modelos**

A seguir, são ajustados os modelos propostos com base no conjunto de dados anterior. As estimativas são obtidas pela maximização das funções verossimilhança derivados na seção 4.1 e os intervalos de confiança dos parâmetros são calculados conforme seção 4.3, utilizando os programas desenvolvidos em R (www.R-project.org, v.3.1.2) que estão apresentados no apêndice deste trabalho.

A Tabela 8.1 apresenta as estimativas pontuais dos parâmetros e os respectivos intervalos com 95% de coeficiente de confiança para o modelo de MPP (Manutenção Preventiva Perfeita) e dos modelos de redução de idade (RI) e redução de intensidade de falha (RF) com suas respectivas formas funcional para a severidade da MP. Nesta tabela também estão apresentados os *p-valores* dos TRV e do teste CVM dos modelos.

Tabela 8.1 – estimativas pontuais, intervalos (95% de confiança) e *p-valores* do TRV e CVM

| | **Parâmetros** | | | | ***p-valor*** **TRV** | ***p-valor*** **CVM** |
|---|---|---|---|---|---|---|
| **Modelo** | ***Lambda*** | ***Beta*** | ***a/b*** | ***Theta*** | | |
| MPP | 57,34 (45,25 - 72,65) | 1,53 (1,28 - 1,82) | - | - | - | < 0,0001 |
| Modelo RI - *a* | 102,04 (77,59 - 134,20) | 2,17 (1,76 - 2,66) | 0,76 (0,61 - 0,95) | - | 0,00052 | 0,20919 |
| Modelo RIE - *a Exponencial* | 113,34 (85,50 - 150,24) | 2,19 (1,78 - 2,70) | - | 2,06 (1,20 - 3,53) | 0,00051 | 0,78070 |
| Modelo RIL - *a Linear* | 119,99 (91,08 - 158,08) | 2,16 (1,74 - 2,67) | - | 1,02 (0,78 - 1,35) | 0,00107 | 0,76692 |
| Modelo RIP - *a Potência* | 111,10 (81,28 - 151,86) | 2,16 (1,76 - 2,65) | - | 0,64 (0,24 - 1,66) | 0,00070 | 0,79962 |
| Modelo RF - *b* | 123,02 (101,98 - 148,38) | 3,11 (2,24 - 4,30) | 0,61 (0,41 - 0.90) | - | 0,00149 | 0,46252 |
| Modelo RFE - *b Exponencial* | 126,24 (101,27 - 157,37) | 2,74 (2,06 - 3,63) | - | 1,48 (0,78 - 2,83) | 0,00142 | 0,59751 |
| Modelo RFL - *b Linear* | 126,71 (99,53 - 161,31) | 2,55 (2,01 - 3,24) | - | 0,89 (0,61 - 1,32) | 0,00184 | 0,56502 |
| Modelo RFP - *b Potência* | 128,43 (101,55 - 162,43) | 2,58 (1,84 - 3,63) | - | 1,14 (0,47 - 2,72) | 0,00225 | 0,51561 |

Fonte: elaborada pelo autor

Após estimar os parâmetros, o objetivo consiste em identificar qual o modelo que proporciona o melhor ajuste para os dados e utilizá-lo para os cálculos de confiabilidade e otimização da periodicidade de MP.

Seguindo a metodologia proposta na seção 5.3, será utilizado o TRV como critério para seleção de modelos. Pode ser observado na Tabela 8.1 que, definindo $\alpha$=1%, há significância estatística da inclusão do fator de melhoria para todos os modelos de MPI propostos.

O modelo com maior significância na inclusão do parâmetro (menor *p-valor*) é o modelo considerando o método redução de idade com fator de melhoria variável utilizando

relação exponencial (modelo RIE). Portanto, esse modelo apresenta um melhor ajuste quando comparado aos demais modelos propostos para o caso em análise.

O TRV permite, apenas, comparar a qualidade do ajuste entre dois modelos considerando a inclusão de um parâmetro. Na seção 5.2.1, foi apresentado o teste CVM que será utilizado para verificar a hipótese que os tempos de falha seguem um processo definido pela função intensidade do modelo RIE estimado. Para o conjunto de dados e para a função intensidade selecionada (modelo RIE), o teste CVM apresentou um *p-valor* de 0,78070. Considerando-se $\alpha$=1%, conclui-se que a hipótese nula pode ser aceita, ou seja, a função intensidade do modelo RIE é o modelo que representa o processo para os dados observados.

Nota-se que os modelos RI e RIE têm *p-valor* do TRV muito próximos, isso porque, neste caso, o fator de melhoria fixo do modelo RI ($a$=0,76) é aproximadamente uma média dos valores do fator de melhoria variável do modelo RIE para as condições de severidade adotadas. Porém o *p-valor* do CVM do modelo RIE (0,78070) é maior do que o *p-valor* do modelo RI (0,20919) o que ratifica a escolha pelo modelo RIE.

Para o modelo MPP não é possível aplicar o TRV, pois não há inclusão de parâmetros para este modelo. Porém nota-se que o *p-valor* do CVM do modelo MPP é < 0,0001, o que conduz a rejeição da hipótese nula, ou seja, o modelo de MPP não representa o processo de falha para os dados analisados. Isso demonstra a importância da avaliação do ajuste do modelo antes de sua utilização no cálculo de métricas de confiabilidade e tomada de decisão.

Anteriormente foi analisado o ajuste dos modelos através de testes estatísticos. Esta mesma avaliação pode ser realizada através da análise gráfica. Conforme descrito na seção 5.2.2, a Figura 8.2 apresenta a comparação entre a FVM estimada e FVM não paramétrica, que é utilizado para avaliar a qualidade do ajuste de dois modelos:

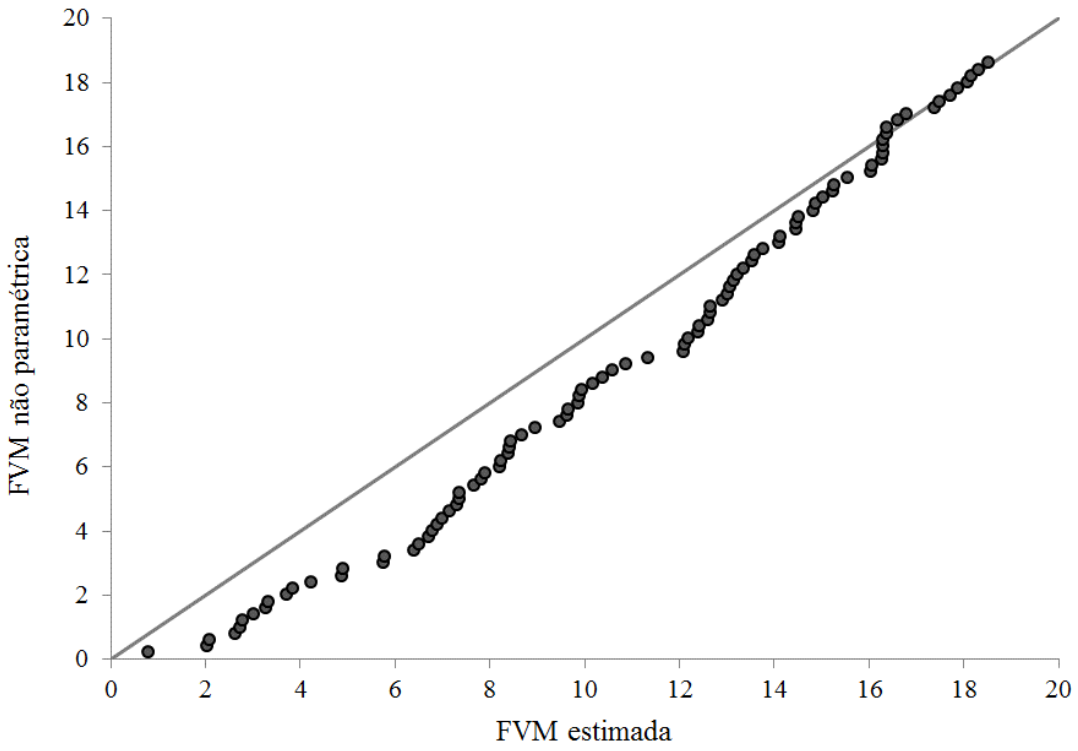
(a) modelo de manutenção preventiva perfeita (MPP);

(b) modelo redução de idade com relação exponencial (RIE).

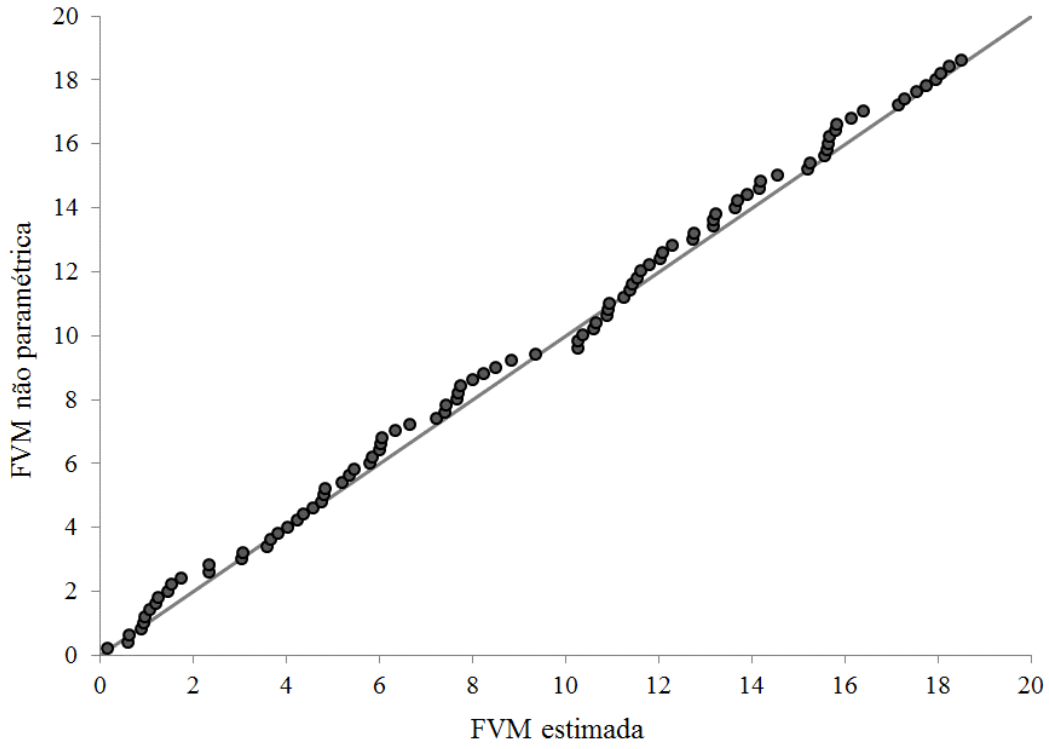
Na Figura 8.2 (a), pode-se verificar os pontos distantes da reta indicando que o modelo de MPP não seria adequado para os dados de falhas analisados.

Na Figura 8.2 (b), a FVM do modelo RIE está próximo da FVM não paramétrica, o que confirma o resultado obtido pelo teste estatístico CVM, ou seja, o modelo apresenta um ajuste adequando para o conjunto de dados. Com base nesses dois gráficos, pode-se concluir que a análise gráfica proposta captura o bom e o mau ajuste de um modelo aos dados analisados.

Figura 8.2 –comparação entre a Função Valor Médio (FVM) estimada e não paramétrica para o modelo de MPP **(a)** e para o modelo RIE **(b)**




**(a)**




**(b)**

Fonte: elaborado pelo autor

Após os testes estatísticos e análise gráfica, conclui-se que o modelo RIE é o que apresenta melhor ajuste para os dados analisados. E, desta forma, será utilizado nos próximos cálculos desta aplicação.

## 8.2 **Modelo selecionado: redução de idade com relação exponencial (RIE)**

Na Tabela 8.2, estão às estimativas pontuais para os parâmetros do modelo RIE selecionado e os limites dos intervalos de confiança com $\gamma$ = 95% construídos segundo descrito na seção 4.2 desta tese.

Tabela 8.2 – estimativa pontual e intervalo de confiança dos parâmetros do modelo RIE

| | **Intervalo de confiança 95%** | | |
|---|---|---|---|
| **Parâmetros** | **Limite inferior** | **Estimativa pontual** | **Limite superior** |
| Lambda | 85,50 | 113,34 | 150,24 |
| Beta | 1,78 | 2,19 | 2,70 |
| Theta | 1,20 | 2,06 | 3,53 |

Fonte: elaborada pelo autor

O parâmetro Beta estimado foi de 2,19, com intervalo de confiança de 95 % entre 1,78 e 2,70, indicando que a função intensidade de falha aumenta com tempo. Conforme Moubray (1997), isso significa que os sistemas estão na fase de envelhecimento, ou seja, os equipamentos tendem a falhar com mais frequência ao longo do tempo, o que justifica a necessidade de MP.

A estimativa do parâmetro Theta foi 2,06, com intervalo de confiança de 95 % entre 1,20 e 3,53, indicando que a função *a(s)* com relação exponencial terá um valor menor do que um para diferentes severidades. Ou seja, após uma ação de MP, retorna para um estado intermediário, não retornando ao estado similar ao inicio de vida (“tão bom quanto novo”). Portanto, deve-se estabelecer a política de MP levando em consideração o fato que a ação de manutenção não conduz o sistema a uma intensidade de falha similar ao inicio de operação. Caso contrário, pode levar a decisões equivocadas na gestão da manutenção do sistema.

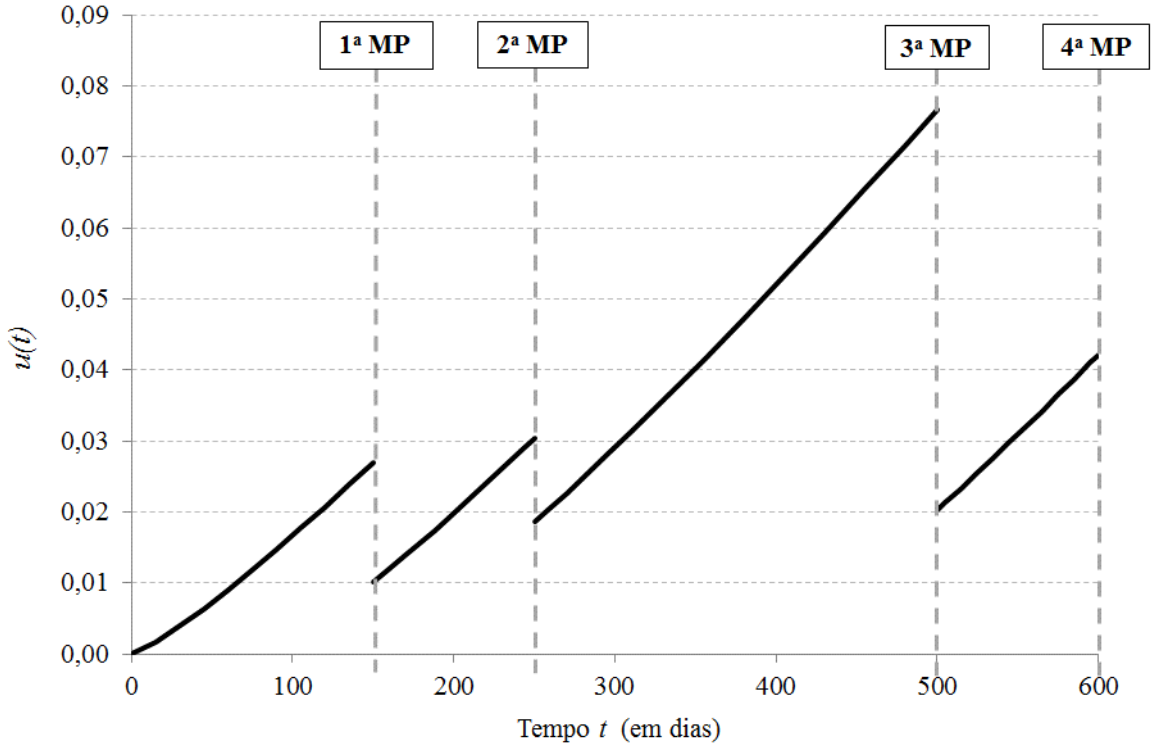
Com base nos valores estimados dos parâmetros do modelo RIE, a Figura 8.3 apresenta a função intensidade de falha considerando as ações de manutenção preventivas executadas durante a operação do sistema. A título de comparação, foi contruído a Figura 8.4

que apresenta a função intensidade de falha sob a suposição de MPP, ou seja, reconduzindo o sistema para a condição de vida inicial a cada ação preventiva.

Na Figura 8.3 pode-se notar que o modelo proposto captura o efeito da manutenção preventiva não ser perfeita ao longo dos ciclos de realização. Inclusive, demonstra as diferentes magnitudes de MP realizadas em virtude de a terceira ação preventiva ter sido realizada com maior severidade, proporcionado redução mais relevante da intensidade de falha do sistema. Na Figura 8.4, sob a suposição de MPP, a cada ação preventiva a função intensidade retorna para o valor zero, não possibilitando diferenciar o efeito da MP no comportamento de falha do sistema.

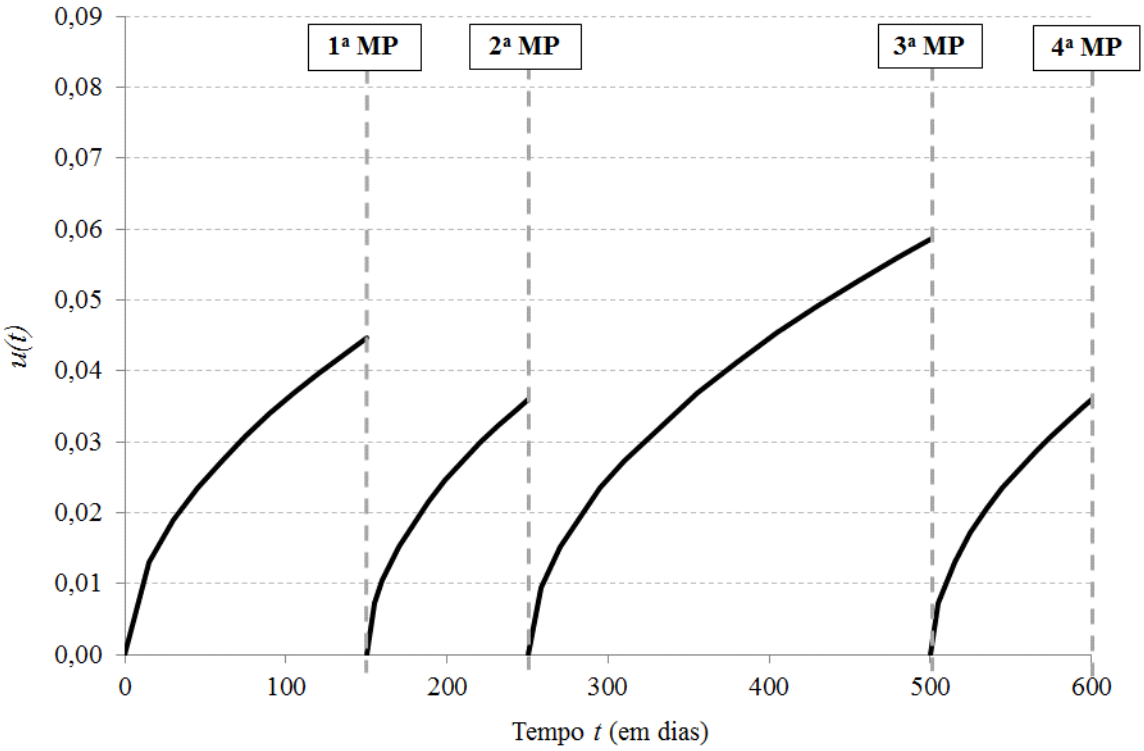
A inclusão do fator de melhoria variável possibilitou a mensuração do efeito da MP na função intensidade, bem como avaliar os diferentes graus desse efeito em função da severidade das atividades da ação.

Figura 8.3 – função intensidade estimada para o modelo RIE




Fonte: elaborado pelo autor

Figura 8.4 – função intensidade estimada para o modelo de MPP




Fonte: elaborado pelo autor

## 8.3 **Função do fator de melhoria variável**

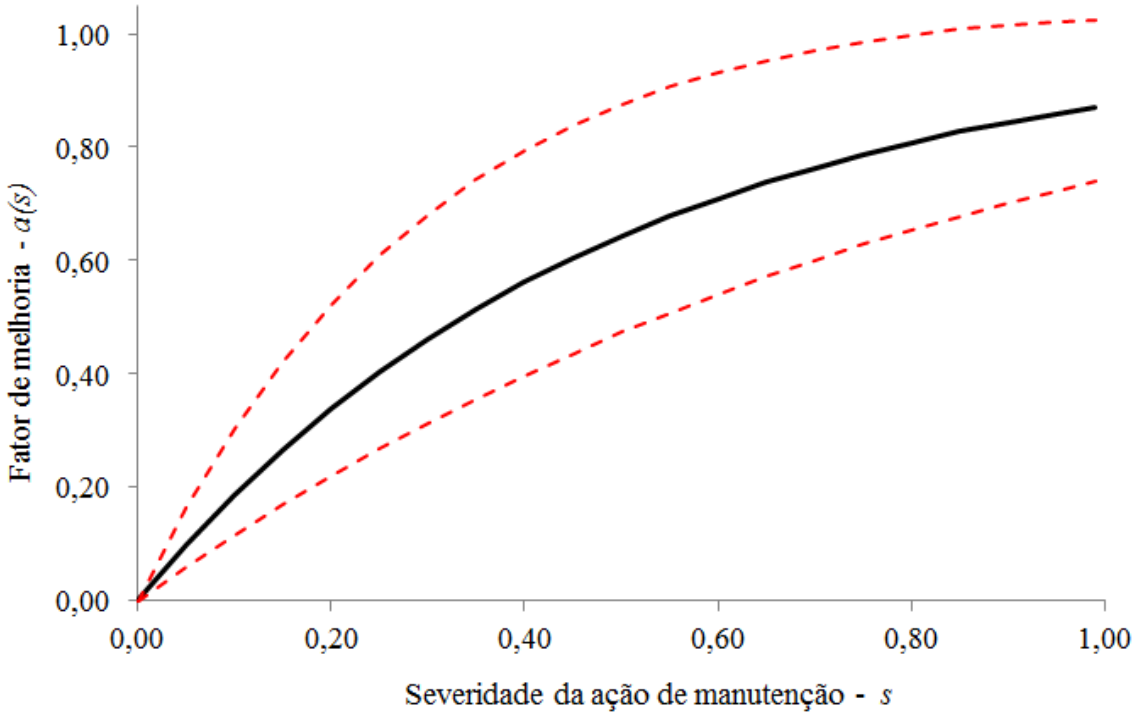
O principal elemento do fator de melhoria variável é a função *a(s)* que pode ser utilizada para medir o fator de melhoria para uma dada severidade da ação de manutenção preventiva. O modelo selecionado RIE possui fator de melhoria com relação exponencial e com base no parâmetro estimado, Theta igual a 2,06, tem-se a seguinte forma funcional:

$$a(s) = 1 - Exp(-2{,}06\,.\,s).$$

onde *s* é a severidade da ação preventiva realizada.

A Figura 8.5 é a representação da função *a(s)* (linha continua) com seu respectivo intervalo de confiança com $\gamma = 95\%$ (linha tracejada) que apresenta a relação entre o fator de melhoria da MP e a severidade da atividade de MP para o modelo RIE.

Figura 8.5 – relação entre fator de melhoria e severidade para o modelo RIE




Fonte: elaborado pelo autor

Por meio dessa figura, pode-se extrair a informação da eficiência da ação preventiva (fator de melhoria) para um dado grau de severidade da MP. Por exemplo, considerando que a última realização da ação preventiva possui severidade de 0,7 pontos (obtido utilizando a metodologia de *BSC* proposta na seção 3.3). Nesse caso o fator de melhoria correspondente é de 0,764 (com variação entre 0,601 e 0,971). Isso significa que, após a realização da ação preventiva, a idade do sistema será reduzida em 76,40% com relação à intensidade de falha. Vale ressaltar, que este valor é obtido como reflexo da ação de manutenção executada, pois basea-se na estimativa dos parâmetros realizada através dos dados de falhas observadas.

## 8.4 **Número de falhas considerando o fator de melhoria variável**

O número de falhas esperadas para um determinado período de tempo pode ser calculado através da função $\mu(t)$. No caso do modelo considerando fator de melhoria variável, o número de falhas esperadas depende da severidade da manutenção preventiva realizada.

Considerando o modelo estimado e a política de manutenção atual com *BSC* de 0,7 pontos de severidade (0,764 de fator de melhoria de redução da idade) e utilizando a função valor médio $\mu(t)$ do modelo RIE, o número de falhas esperadas para o próximo ano de um

sistema é de 25,04 falhas. Se os gestores do sistema definem uma redução de custos da ação preventiva com a diminuição de tarefas executadas e itens trocados, o que resulta em um *BSC* de severidade de 0,5 pontos, pode-se determinar então que o fator de melhoria será de 0,644. Em decorrência disso, o número de falhas esperadas para o próximo ano será de 31,71 falhas.

Portanto, alterando a política atual de manutenção preventiva de 0,7 pontos para 0,5 pontos de severidade, espera-se que ocorra um aumento de aproximadamente 7 falhas no período de um ano. Com base nessa informação, considerando-se o aumento de custos de ações corretivas e da indisponibilidade do sistema que esse aumento de falhas provocaria, os gestores poderiam avaliar a melhor decisão com relação à nova política de execução da ação preventiva do equipamento.

Quando utilizado o modelo de fator de melhoria constante ou MPP, o número de falhas esperadas (30,16 falhas para RI e 17,05 falhas para MPP) é independente do plano de ação preventivo realizado. O que pode tornar a previsão de falhas equivocada, no caso subestimando o número de falhas, pois não leva em consideração as características da atividade preventiva realizada.

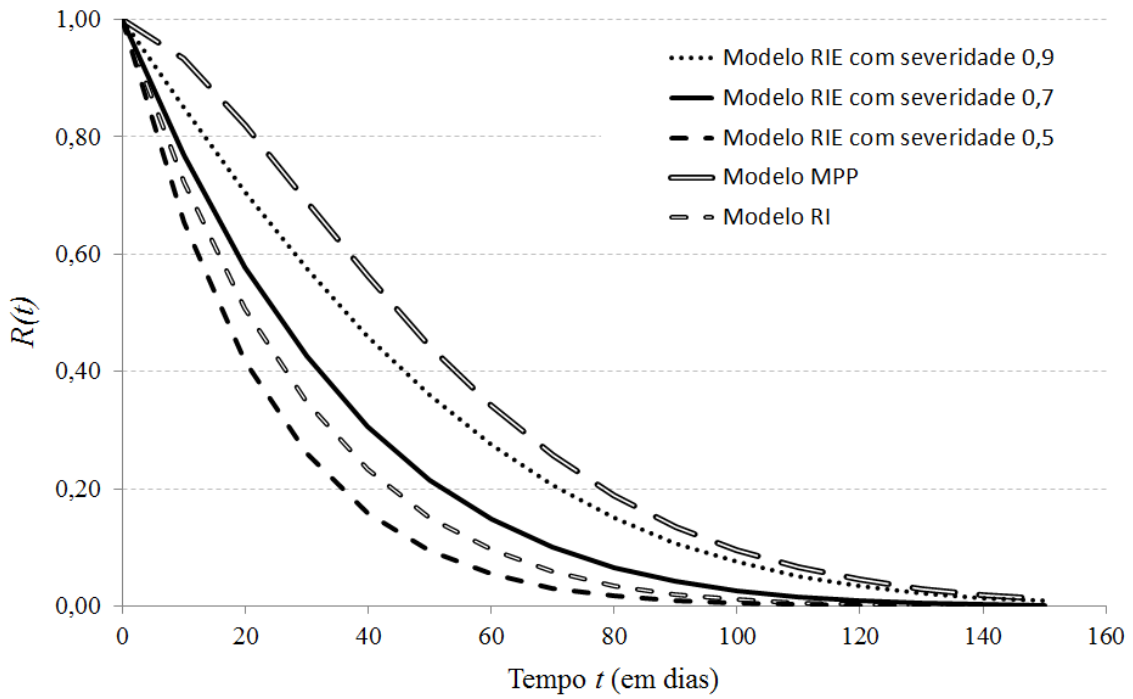
## 8.5 **Confiabilidade considerando o fator de melhoria variável**

Uma análise similar a anterior pode ser realizada utilizando o prisma de confiabilidade conforme a definição do capítulo 7. Para isso, foi calculada a previsão da função de confiabilidade para os tempos de operação após a última ação preventiva, no caso 600 dias.

O modelo com fator de melhoria variável realiza a previsão da confiabilidade considerando diferentes severidades de execução da MP. Esta situação é mais realista do que as consideradas pelos modelos de fator de melhoria constante e pelo modelo de MPP, pois estes consideram uma única condição de realização da ação preventiva, independente de alterações nos planos de manutenção e atividades realizadas.

A Figura 8.6 e a Tabela 8.3 apresentam a confiabilidade calculada após 600 dias para o modelo RI ($a$=0,76 constante), modelo MPP ($a$=1 constante) e o modelo RIE considerando três condições de severidades: 0,5, 0,7 e 0,9.

Figura 8.6 – função confiabilidade após *T*=600 dias para o modelo RI, modelo MPP e modelo RIE (0,5, 0,7 e 0,9)




Fonte: elaborado pelo autor

Tabela 8.3 – valores da confiabilidade após *T*=600 dias para o modelo RI, modelo MPP e modelo RIE (0,5; 0,7 e 0,9)

| t | R(t) | | | | |
|---|---|---|---|---|---|
| | Modelo RIE com severidade 0,5 | Modelo RIE com severidade 0,7 | Modelo RIE com severidade 0,9 | Modelo MPP | Modelo RI |
| 0 | 1,000 | 1,000 | 1,000 | 1,000 | 1,000 |
| 10 | 0,654 | 0,769 | 0,849 | 0,933 | 0,722 |
| 20 | 0,418 | 0,578 | 0,706 | 0,819 | 0,508 |
| 30 | 0,261 | 0,426 | 0,575 | 0,690 | 0,348 |
| 40 | 0,159 | 0,307 | 0,459 | 0,562 | 0,233 |
| 50 | 0,095 | 0,216 | 0,359 | 0,445 | 0,151 |
| 60 | 0,055 | 0,149 | 0,275 | 0,342 | 0,096 |
| 70 | 0,031 | 0,100 | 0,206 | 0,257 | 0,059 |
| 80 | 0,017 | 0,066 | 0,151 | 0,189 | 0,035 |
| 90 | 0,009 | 0,042 | 0,108 | 0,136 | 0,021 |
| 100 | 0,005 | 0,027 | 0,076 | 0,096 | 0,012 |
| 110 | 0,003 | 0,016 | 0,052 | 0,066 | 0,006 |
| 120 | 0,001 | 0,010 | 0,035 | 0,045 | 0,003 |
| 130 | 0,001 | 0,006 | 0,023 | 0,030 | 0,002 |
| 140 | 0,000 | 0,003 | 0,015 | 0,020 | 0,001 |
| 150 | 0,000 | 0,002 | 0,009 | 0,013 | 0,000 |

Fonte: elaborada pelo autor

Para o modelo RIE, a função confiabilidade é maior quando a severidade da ação preventiva é 0,9 quando comparada com as severidades de 0,7 ou 0,5. Isso ocorre porque o modelo reflete que a MP executada com 0,9 possui uma efetividade maior na confiabilidade do sistema quando comparada com as condições 0,7 e 0,5.

Além disso, é possível quantificar esta diferença. Por exemplo, da Tabela 8.3, em um tempo de 20 dias, considerando a severidade 0,9 a confiabilidade é 0,706, utilizando a severidade 0,7 a confiabilidade é de 0,578 e para a severidade 0,5 a confiabilidade é de 0,418. Estas informações podem ser interpretadas como uma visão do impacto na confiabilidade do sistema para uma dada alteração no plano de manutenção. Esse tipo de métrica é de alta relevância para a gestão de manutenção, pois permite mensurar a eficiência da ação preventiva que será executada e auxiliar em uma decisão mais assertiva por parte dos gestores do sistema.

A previsão do modelo RI (*a* constante) esta em penúltimo lugar, enquanto que a previsão do modelo MMP (*a*=1) é a maior entre os modelos analisados. Porém estes modelos não mensuram o impacto de mudanças de atividades preventivas na confiabilidade do sistema, ou seja, a previsão, utilizando esses modelos, considera a execução de MP fixa durante o período de operação do sistema. O que pode promover uma previsão otimista ou pessimista da confiabilidade como no caso analisado.

## 8.6 **Otimização da periodicidade de MP considerando o fator de melhoria variável**

Outro ponto importante no processo decisório dos gestores responsáveis pela manutenção é a determinação da periodicidade de execução da preventiva. No capítulo 6, foi desenvolvido um método para otimização da periodicidade de MP, considerando-se a relação de custos de manutenção e utilizando os modelos propostos que possibilitam realizar a otimização considerando que a ação de preventiva não é perfeita, ou seja, não leva o sistema para uma condição igual à nova, mas uma condição intermediária. Uma aplicação da metodologia de otimização proposta pode ser realizada utilizando o modelo RIE estimado anteriormente.

Considerando que com 600 dias de operação foi realizada uma MP no equipamento com uma severidade de 0,7 pontos. Supondo-se que o custo da ação preventiva é de 3 unidades monetárias e da ação corretiva é de 2 unidades monetárias, lembrando que o importante não são os valores absolutos dos custos, mas a relação entre custos preventivo e corretivo. Assim, utilizando o procedimento descrito no capítulo 6 desta tese, o tempo ótimo

para execução da próxima ação preventiva que minimiza os custos de manutenção nessas condições é de 714 dias (ou mais 114 dias de operação).

Utilizando o modelo de fator de melhoria variável, a otimização pode ser influenciada pela severidade da ação de MP, ou seja, se o plano de manutenção preventivo for alterado, o tempo da periodicidade da preventiva também será alterado. Deve-se considerar, também, o fato que, à medida que a severidade da ação preventiva aumenta (mais tarefas e itens substituídos), há um aumento dos custos preventivos e a relação dos custos de manutenção se altera. A Tabela 8.4 foi construída para avaliar o comportamento do tempo ótimo de ação preventiva para diferentes condições de realização da manutenção preventiva, considerando que os custos desta última aumentam e os custos da ação corretiva são fixos.

Tabela 8.4 – tempos ótimos de MP para diferentes severidades da MP

| Severidade MP | Custo MP | Eficiência MP | Tempo Ótimo |
|---|---|---|---|
| 0,4 | 1,0 | 56,2% | 664 |
| 0,5 | 2,0 | 64,4% | 691 |
| 0,7 | 3,0 | 76,4% | 714 |
| 0,8 | 4,0 | 80,8% | 733 |

Fonte: elaborada pelo autor

Observa-se que à medida que a severidade da MP aumenta, ocorre um aumento na periodicidade da realização da preventiva. Isso ocorre devido ao fato de que o aumento da severidade proporciona aumento da confiabilidade do sistema, ou seja, maior eficiência da ação de MP permite um espaçamento maior entre as manutenções preventivas.

Diferente do modelo tradicional de MP perfeita, que não permite a variação da ação de manutenção, a incorporação dos modelos propostos na otimização permite que sejam considerados diferentes cenários de realização do plano de manutenção na otimização da periodicidade de ações preventivas e mensurado o impacto da eficiência na confiabilidade do sistema.

# 9. **CONCLUSÃO**

Neste trabalho foram apresentados modelos estatísticos de confiabilidade para sistemas reparáveis que considerem ação de manutenção preventiva imperfeita utilizando o fator de melhoria. Foi proposta uma nova classe de modelos que considera as diferentes condições e atividades dos planos de manutenção realizados denominados de modelos com fator de melhoria variável.

Por meio da construção da função verossimilhança dos modelos propostos foi desenvolvida a metodologia que permitiu estimar os parâmetros com base em dados de falha, calcular seus respectivos limites de confiança e avaliar a qualidade do ajuste dos modelos por meio de testes estatísticos e análise gráfica para os modelos de manutenção imperfeita já existentes e para a nova classe de modelos propostos.

A metodologia foi utilizada em um conjunto de dados com a finalidade de avaliar a aplicação dos modelos propostos. De acordo com os critérios para a seleção do modelo, o modelo mais adequado foi o fator de melhoria pelo método redução de idade com fator de melhoria variável utilizando relação exponencial, denominado modelo RIE. Com base nos parâmetros estimados desse modelo, obteve-se o Beta de 2,19, com intervalo de confiança de 95 % entre 1,78 e 2,70 e o parâmetro Theta de 2,06, com intervalo de confiança de 95 % entre 1,20 e 3,53. Esses valores revelam que os sistemas analisados tendem a falhar com mais frequência ao longo do tempo, necessitando de ações preventivas e, também, que as ações preventivas tendem a deixar o equipamento em um estado diferente da condição inicial de operação, ou seja, as ações não são perfeitas, pois não levam o sistema para o estado de novo. Dessa forma pode-se verificar que, quando há execução de ação preventiva imperfeita, os modelos propostos apresentam melhor ajuste quando comparado ao modelo tradicional de manutenção perfeita.

A nova classe de modelos com fator de melhoria variável representa uma possibilidade de ajuste mais eficiente aos dados por ter uma parametrização mais flexível que captura as diferentes condições da MP e por generalizar os demais modelos de fator de melhoria propostos na literatura. Além disso, nesta tese, não foi, apenas, proposta a nova classe de modelos como desenvolvido todo seu arcabouço inferencial, por meio do desenvolvimento de métodos de estimação, intervalos de confiança e avaliação da qualidade de ajuste dos modelos.

Além de permitir uma modelagem dos dados com maior acurácia, os novos modelos fornecem informações importantes para o gerenciamento de decisões de manutenção, tais

como mensurar o impacto de novos planos de ações preventivas na confiabilidade do sistema e considerar uma nova condição de manutenção na otimização da periodicidade das ações de MP, proporcionando informações relevantes para definição de novos planos de manutenção.

Outro ponto importante desenvolvido neste trabalho foram os programas computacionais desenvolvidos, pois, como não há implantado em nenhum software comercial, a metodologia proposta dos modelos de manutenção imperfeita e dos novos modelos de fator de melhoria variável, foi desenvolvido código computacional, na linguagem de programação R, que possibilita a transferência do conhecimento desenvolvido neste trabalho e a aplicação da metodologia proposta para outros conjuntos de dados de falhas de sistemas reparáveis.

Uma dificuldade para utilização do modelo proposto é a obtenção dos dados utilizados para estimação dos seus parâmetros. Além dos dados de tempo de falha, deve ser feito um levantamento da descrição das atividades de manutenção preventivas realizadas ao longo da operação do sistema, o que é necessário para propor um valor de severidade das ações preventivas. Essas informações algumas vezes não são registradas ou são de difícil recuperação o que poderá inviabilizar a utilização do modelo.

O estudo de simulação realizado, apesar de não ser um estudo conclusivo sobre as questões assintóticas dos estimadores propostos, permitiu avaliar questões pontuais sobre essa propriedade dos estimadores e fazer sugestões de utilização mais adequada dos modelos para condições de dados disponíveis.

Vale ressaltar que o modelo proposto também pode ser utilizado em outras situações como para mensurar o efeito de alterações de projeto na confiabilidade do sistema. Para isso deve-se considerar a alteração de projeto como uma ação de manutenção preventiva e avaliar se o efeito do fator de melhoria é significativo ou não na intensidade de falha do sistema e, desta forma, avaliar o efeito da alteração executada no sistema.

Como sugestão para trabalhos futuros propõe-se alguns avanços listados a seguir:

– avaliação da metodologia bayesiana para estimação dos parâmetros;

– inclusão do fator de melhoria conjuntamente para ação preventiva e corretiva;

– estudo detalhado das propriedades assintóticas dos estimadores propostos;

– utilização da metodologia proposta com a função de intensidade de falha log-linear.

**REFERÊNCIAS**

ASCHER, H.; FEINGOLD, H. **Repairable systems reliability modeling, inference, misconceptions and their causes.** Vol. 7, New York: Marcel Dekker, 1984.

BARLOW, R. E.; HUNTER, L. C. **Optimum preventive maintenance policies**. Operations Research, vol. 8, p.90-100, 1960.

BARTHOLOMEW-BIGGS, M.; ZUO, M. J.; LI, X. **Modelling and optimizing sequential imperfect preventive maintenance.** Reliability Engineering and System Safety, vol. 94, p. 53–62, 2009.

BROWN, M.; PROSCHAN, F. **Imperfect repair**. Journal of Applied Probability, vol. 20, n. 4, p. 851-859, 1983.

BUSSAB, W.O.; HAZZAN, S.; MORETTIN, P.A. **Cálculo – Funções de uma e várias variáveis**. Ed. Saraiva, 2003.

CASELLA, G; BERGER, R.L. **Inferência Estatística.** Cengage Learning, 2010.

COLOSIMO, E.A; GIOLO, S.R. **Análise de sobrevivência aplicada**. Ed Blucher, 2006.

CORSET, F.; DOYEN, L., **Bayesian Analysis of ARA Imperfect Repair Models**. Communications in Statistics – Theory and Methods, vol. 41, p. 3915-3941, 2012.

CROW, L. R., **Reliability Analysis for Complex, Repairable Systems in Reliability and Biometry.** SIAM, ed. Proschan e R. J. Serfling, Philadelphia, Pennsylvania, p. 379-410, 1974.

CUI, L., KOU, W., LOH, H. T., e XIE, M., **Optimal allocation of minimal and perfect repairs under resource constraints**. IEEE Transactions on Reliability, vol.53, p. 193–199, 2004.

DO, V.P.; BERENGUER, C. **Condition-based maintenance with imperfect preventive repairs for a deteriorating production system.** Quality and Reliability Engineering International, vol. 28, p. 624–633, 2012.

DOYEN, L. **Asymptotic properties of imperfect repair models and estimation of repair efficiency.** Naval Research Logistics, vol. 57 p. 296-307, 2010.

DOYEN, L.; GAUDOIN, O. **Classes of imperfect repair models based on reduction of failure intensity or virtual age**. Reliability Engineering System Safety, vol. 84, n. 1, p. 45-56, 2004.

DOYEN, L.; GAUDOIN, O. **Modeling and assessment of aging and efficiency of corrective and planned preventive maintenance**. IEEE Trans Reliability, vol. 60, n. 4, p. 759-769, 2011.

GILARDONI, G.; COLOSIMO, E. A. **Optimal Maintenance Time for Repairable Systems.** Journal of Quality and Technology, vol. 39, n. 1, p. 48-54, 2007.

JACK, N. **Analysing event data from repairable machine subject to imperfect preventive maintenance.** Quality and Reliability Engineering International, vol. 13, p.183-186, 1997.

JACK, N. **Age-reduction models for imperfect maintenance**. IMA Journal Math Applied Business Industry, vol. 9, p. 347-354, 1998.

JATURONNATEE, J.; MURTHY, D.; BOONDISKULCHOK, R. **Optimal preventive maintenance of leased equipment with corrective minimal repairs**. European Journal of Operational Research, vol. 174, p. 201-215, 2006.

JAYABALAN, V.; CHAUDHURI, D. **Optimal maintenance and replacement policy for a deteriorating system with increased mean downtime**. Naval Research Logistics, vol. 39, p. 67–78, 1992.

JIANG, R. **Life restoration degree following minimal repair and its application for maintenance decision optimization**. Journal of Quality in Maintenance Engineering, vol. 19, p. 413–428, 2013.

KIJIMA, M.; MORIMURA, H.; SUZUKI, Y. **Periodical replacement problem without assuming minimal repair.** European Journal of Operational Research, vol.37, p.194–203, 1988.

KIJIMA, M. **Some results for repairable systems with general repair**. Journal of Applied Probability, vol. 26, p. 89–102, 1989.

LIN, D.; ZUO, M.J.; YAM, R.C.M. **General sequential imperfect preventive maintenance model.** International Journal of Reliability, Quality and Safety Engineering, vol. 7, n. 3, p. 253-266, 2000.

MALIK, M. **Reliable preventive maintenance scheduling**. AIIE Transactions, vol. 11, n. 3, p. 221–228, 1979.

MEEKER, W.Q.; ESCOBAR, L.A. **Statistical Methods for Reliability Data**. John Wiley & Sons, 1998.

MIL-STD-721C – Military Standard – Department of Defense - **Definition of terms for reliability and maintainability - Revision C**, Waschington, DC, USA, 1981.

MOUBRAY, J. **Reliability-Centered Maintenance**. 2ª. ed. New York: Industrial Press Inc., 1997.

NAKAGAWA, T. **Periodic and sequential preventive maintenance policies**. Journal of Applied Probability, vol. 23, p. 536–542, 1986.

NAKAGAWA, T. **Sequential imperfect preventive maintenance policies**. IEEE Transactions Reliability, vol. 37, n. 3, p. 295-298, 1988.

NAKAGAWA, T.; ZHAO, X. **Asymptotic Replacement and Preventive Maintenance Policies**. IEEE Conference Maintenance and Safety Engineering, p. 417-420, 2011.

NELDER, J. A.; MEAD, R. **A simplex algorithm for function minimization**. Computer Journal, vol. 7, p. 308–313, 1965.

NELSON, W. **Recurrent Events Data Analysis for Product Repairs, Disease Recurrences, and Other Applications**, ASA-SIAM, 2003.

PANDEY, M.; ZUO, M.J.; MOGHADDASS, R.; TIWARI, M.K. **Selective maintenance for binary systems under imperfect repair**. Reliability Engineering and System Safety, vol.113, p.42–51, 2013.

PARK, D. H., JUNG, G. M., E YUM, J. K. **Cost minimization for periodic maintenance policy of a system subject to slow degradation**. Reliability Engineering and System Safety, vol. 68, p. 105–112, 2000.

PHAM, H.; WANG H.Z. **Imperfect maintenance**. European Journal of Operational Research, vol. 94, n.3, p. 425-438, 1996.

PHELPS, R. I. **Replacement policies under minimal repair**. The Journal of Operational Research Society, vol. 32, p. 549–554, 1981.

R (2014). **A language and environment for statistical computing**. The R Foundation for Statistical Computing, ISBN 3-900051-07-0.

RAMÍREZ, P.A.P; UTNE, I.B. **Decision support for life extension of technical systems through virtual age modelling.** Reliability Engineering System Safety. vol. 115, p. 55–69, 2013.

REMY, E.; CORSET, F.; DESPRÉAUX, S.; DOYEN, L.; GAUDOIN, O. **An example of integrated approach to technical and economic optimization of maintenance**. Reliability Engineering and System Safety, vol. 116, p. 8–19, 2013.

RIGDON, S.E; BASU, A.P. **Statistical Methods for the Reliability of Repairable Systems. John Wiley**, New York, 2000.

ROSS, S. **A First Course in Probability** - 9th Edition – Pearson, New York, 2012.

SHEU, S. e GRIFFITH, W., **Multivariate imperfect repair**. Journal of Applied Probability, vol. 29, p. 947–956, 1992.

SHIN, I.; LIM, T.; LIE, C. **Estimating parameters of intensity function and maintenance effect for repairable unit.** Reliability Engineering System Safety, vol. 54, p. 1-10, 1996.

TAGHIPOUR S.; BANJEVIC, D. **Trend analysis of the power law process using expectation-maximization algorithm for data censored by inspection intervals**. Reliability Engineering System Safety, vol. 96, p. 1340–1348, 2011.

TANWAR, M; RAI, R.N.; BOLIA, N. **Imperfect repair modeling using Kijima type generalized renewal process**. Reliability Engineering System Safety, vol. 124, p. 24-31, 2014.

TOLEDO, M.L.G; FREITAS, M.A.; COLOSIMO, E.A.; GILARDONI, G.L. **ARA and ARI imperfect repair models: Estimation, goodness-of-fit and reliability prediction**. Reliability Engineering System Safety, vol. 140, p. 107-115, 2015.

TOLEDO, M.L.G, **Determination of the optimal periodic maintenance policy under imperfect repair assumption**. 2014. Tese de Doutorado - Universidade Federal de Minas Gerais, Belo Horizonte, Março de 2014.

WANG, H. **A survey of maintenance policies of deteriorating systems**. European Journal of Operational Research, vol. 139, p.469–489, 2002.

WU, S.; ZUO, M.J. **Linear and Nonlinear Preventive Maintenance Models**. IEEE Transactions on Reliability, vol. 59, n. 1, p. 242-249, 2010.

YANEZ, M.; JOGLAR, F.; MODARRES, M. **Generalized renewal process for analysis of repairable systems with limited failure experience.** Reliability Engineering System Safety, vol. 77, p.167–180, 2002.

YEH, R.H.; KAO, K-C, CHANG, W.L. **Optimal preventive maintenance policy for leased equipment using failure rate reduction**. Computers & Industrial Engineering, vol. 57, p. 304–309, 2009.

YEVKIN, O.; KRIVTSOV, V. **Comparative analysis of optimal maintenance policies under general repair with underlying Weibull distributions**. IEEE Transactions on Reliability, vol. 62, p.82–91, 2013.

ZHAO, M.; XIE, M. **On maximum likelihood estimation for a general non-homogeneous Poisson process**. Scandinavian Journal of Statistics, vol. 23, p. 597–607, 1996.

ZHOU, X.; XI, L.; LEE, J. **Reliability-centered predictive maintenance scheduling for a continuously monitored system subject to degradation**. Reliability Engineering and System Safety, vol 92, p. 530‑534, 2007.

# APÊNDICE – programas em R

==========
Os programas desenvolvidos neste trabalho podem ser encontrados em:
http://sites.google.com/site/doutoradomarcoscoquejr/
==========

## A.1 - Programa referente ao modelo de Manutenção Preventiva Perfeita (MPP)

```
################################################################################
# Formato do arquivo de entrada dados.txt
# 1a coluna - Tempos de falha acumulados (o ultimo valor do ciclo deve ser a MP)
# 2a Coluna - Número do ciclo de MP ordenado
# 3a coluna - BSC da MP realizada que influênciará o próximo ciclo de MP
# 4a coluna - Número do sistema ordenado
################################################################################
library(nlme)
################################################################################
# Função da LogVerossimilhança para o Modelos de Confiabilidade de Sistemas
# Reparáveis considerando MP Imperfeita
################################################################################
logvero <- function(x){
	# Entrada dos Parâmetros
	lambda<-x[1]
	beta<-x[2]
	a<-x[3]

	lv<-0

	# Entrada de Dados
	dados_bruto<-read.table(file="dados.txt", sep = " ")

	# Número de Sistemas analisados
	m<-max(dados_bruto[,4])

	tam<-vector(mode="integer",length=m)

	# Define a quantidade de dados para cada sistema
	for (k in 1:nrow(dados_bruto)) {
		tam[dados_bruto[k,4]]<-tam[dados_bruto[k,4]]+1
	}

	linha<-1
	for (sistema in 1:m) {
		dados<-matrix(0,tam[sistema],3)
		# Entrada de dados do sistema k
		for (k in 1:tam[sistema]) {
			dados[k,1]<-dados_bruto[linha,1]
			dados[k,2]<-dados_bruto[linha,2]
			dados[k,3]<-dados_bruto[linha,3]
			linha<-linha+1
		}

		total<-nrow(dados)
		ciclos<-dados[total,2]

		n<-vector(mode="integer",length=ciclos)
		T<-vector(mode="integer",length=ciclos)
		S<-vector(mode="integer",length=ciclos)
```

```
		# Cria vetor com os tempos e severidade da MP em cada ciclo
		# e o número de falhas em cada cilco
		for (i in 1:(total-1)){
			if (dados[i,2]<dados[i+1,2]) {T[dados[i,2]]<-dados[i,1]
								S[dados[i,2]]<-dados[i,3]}
			else {n[dados[i,2]]<-n[dados[i,2]]+1}
		}

		T[dados[i+1,2]]<-dados[i+1,1]
		S[dados[i+1,2]]<-dados[i+1,3]

		# Número total de falhas
		N<-sum(n)

		# Cria vetor com os tempos de falha e severidades da MP
		t<-matrix(0,ciclos,max(n))

		j<-1
		for (i in 1:(total-1)){
			if (dados[i,2]<dados[i+1,2]) { j<-1 }
			else {t[dados[i,2],j]<-dados[i,1]
			     j<-j+1}
		}
		#######################################################################
		# Cálculo da LogVerossimilhança
		#######################################################################
		lv<-lv+(N*log(beta))+(N*log(((1/lambda)^beta)))

		Va<-0 #Idade Virtual inicial

		j<-1
		aux<-0
		for (i in 1:n[j]){
			if(n[j]>0) {aux<-aux+log(t[j,i])}
		}

		lv<-lv+((beta-1)*aux)
		lv<-lv-(((1/lambda)^beta)*(T[j]^beta))
		Va<-T[j]

		if(ciclos>1){
			for (j in 2:ciclos){

				aux<-0
				for (i in 1:n[j]){

					if(n[j]>0) {aux<-aux+log(t[j,i]-(a*Va))}

				}

				lv<-lv+((beta-1)*aux)

				lv<-lv-(((1/lambda)^beta)*(((T[j]-(a*Va))^beta)-(((1-a)*Va)^beta)))

				Va<-T[j]
			}
		}
		#######################################################################
	}
	restricao<-0
	for (j in 1:ciclos){
		#if( a < 0 ) {restricao<-1}
		#if( a > 1 ) {restricao<-1}
	}
	if(restricao==1) {return(-Inf)} else{return(lv)}
}
```

```
##########################################################################################
# O método da função optim de otimização encontra o mínimo, portanto para determinar
# o máximo da função LogVerossimilhança foi utilizado seu valor negativo
##########################################################################################
logvero_opt <- function(x){

        lambda<-x[1]
        beta<-x[2]
        a<-1
        return(-logvero(c(lambda,beta,a)))
}
##########################################################################################
# Otimização da LogVerossimilhança
##########################################################################################
mle<-optim(c(100, 1.8), logvero_opt, method = "Nelder-Mead")
##########################################################################################
# PARÂMETROS ESTIMADOS
lambda.hat<-mle$par[1]
beta.hat<-mle$par[2]
a.hat<-1 # considerando MPP

param<-c(mle$par[1], mle$par[2])

# Deve ser zero para indicar que o algoritmo convergiu
convergencia<-mle$convergence

mat<-fdHess(mle$par, logvero_opt)
hessiano<-(-mat$Hessian)

# Para ser ponto de máximo devem ser alternados os sinais

# VERIFICAÇÃO 1 (Negativo)
detver1<-hessiano[1,1]

#VERIFICAÇÃO 2 (Positivo)
ver2<-matrix(0,2,2)
ver2[1,1]<-hessiano[1,1]
ver2[1,2]<-hessiano[1,2]
ver2[2,1]<-hessiano[2,1]
ver2[2,2]<-hessiano[2,2]

detver2<-det(ver2)

# Outra verificação: se os Autovalores forem todos negativos, é ponto de máximo.
# VERIFICAÇÃO AUTOVALORES
av<-eigen(hessiano)

##########################################################################################
# Resumo Verificações
##########################################################################################

if(detver1<0){"OK"}else{"Verificar"}
if(detver2>0){"OK"}else{"Verificar"}

if(av$values[1]<0){"OK"}else{"Verificar"}
if(av$values[2]<0){"OK"}else{"Verificar"}

##########################################################################################
# Desvios Padrãoes das estimativas
##########################################################################################
dp.lambda.hat<-sqrt(solve(-hessiano)[1,1])
dp.beta.hat<-sqrt(solve(-hessiano)[2,2])
```

```
# Nível de Confiança
NC<-0.95

IC_lambda_Inf<-lambda.hat/(exp(qnorm((1+NC)/2)*dp.lambda.hat/lambda.hat))
IC_lambda_Sup<-lambda.hat*(exp(qnorm((1+NC)/2)*dp.lambda.hat/lambda.hat))

IC_beta_Inf<-beta.hat/(exp(qnorm((1+NC)/2)*dp.beta.hat/beta.hat))
IC_beta_Sup<-beta.hat*(exp(qnorm((1+NC)/2)*dp.beta.hat/beta.hat))

##################################################################################
# Verossimilhança da Manutenção Sem Efeito (MSE)
##################################################################################
logvero_mse <- function(x){

        lambda<-x[1]
        beta<-x[2]
        a<-0
        return(-logvero(c(lambda,beta,a)))
}

mle_mse<-optim(c(100, 1.5), logvero_mse, method = "Nelder-Mead")

param_mse<-c(mle_mse$par[1], mle_mse$par[2])
##################################################################################
# Teste de Razão de Verossimilhanças
##################################################################################
TRV<-(-2*(logvero(c(param_mse,0))-(logvero(c(param,1)))))

# P-valor (H0: a=0)
1-pchisq(TRV, 1)

# Estatística TRV
TRV

# Valor Crítico (VC)
qchisq(0.95,1)

# Se TRV < VC, então aceita H0 (não há efeito entre as MP)
##################################################################################
# Teste Cramer-von Mises
##################################################################################
# Considera o último tempo de cada ciclo como a MP
# É considerado nos cálculos se o último tempo do sistema for falha ou MP
##################################################################################
# Entrada de Dados
dados_bruto<-read.table(file="dados.txt", sep = " ")

# Número de sistemas analisados
m<-max(dados_bruto[,4])

# tam: quantidade de dados para cada sistema
# ciclo: quantidade de ciclos para cada sistema
tam<-vector(mode="integer",length=m)
ciclo<-vector(mode="integer",length=m)

for (k in 1:nrow(dados_bruto)) {
        tam[dados_bruto[k,4]]<-tam[dados_bruto[k,4]]+1
        if ( k==nrow(dados_bruto) ) { ciclo[dados_bruto[k,4]]<-dados_bruto[k,2] }
        else { if (dados_bruto[k,4]<dados_bruto[k+1,4]) { ciclo[dados_bruto[k,4]]<-dados_bruto[k,2] }}
}

#Número Total de falhas considerando todos os sistemas
M<-sum(tam)-sum(ciclo)

CVM_aux<-matrix(0,M,2)
```

```
##################################################################################
# Definição das funções mi para os 5 possíveis ciclos
##################################################################################

mi1<-function(t){((t^beta.hat)*((1/lambda.hat)^beta.hat))}

mi2<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                        (((t-(a.hat*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))}

mi3<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                        (((T[2]-(a.hat*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))+
                        (((t-(a.hat*T[2]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[2])^beta.hat)*((1/lambda.hat)^beta.hat))}

mi4<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                        (((T[2]-(a.hat*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))+
                        (((T[3]-(a.hat*T[2]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[2])^beta.hat)*((1/lambda.hat)^beta.hat))+
                        (((t-(a.hat*T[3]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[3])^beta.hat)*((1/lambda.hat)^beta.hat))}

mi5<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                        (((T[2]-(a.hat*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))+
                        (((T[3]-(a.hat*T[2]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[2])^beta.hat)*((1/lambda.hat)^beta.hat))+
                        (((T[4]-(a.hat*T[3]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[3])^beta.hat)*((1/lambda.hat)^beta.hat))+
                        (((t-(a.hat*T[4]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[4])^beta.hat)*((1/lambda.hat)^beta.hat))}

##################################################################################
linha<-1
linha2<-1
for (sistema in 1:m) {

        dados<-matrix(0,tam[sistema],3)

        # Entrada de dados do sistema
        for (k in 1:tam[sistema]) {
                dados[k,1]<-dados_bruto[linha,1]
                dados[k,2]<-dados_bruto[linha,2]
                dados[k,3]<-dados_bruto[linha,3]
                linha<-linha+1}
        total<-nrow(dados)
        ciclos<-dados[total,2]

        # Cria vetor com os tempos de falha e severidades da MP
        falhas<-matrix(0,(total-ciclos),2)

        # Cria vetor com os tempos de MP e Severidades
        T<-vector(mode="integer",length=ciclos)
        S<-vector(mode="integer",length=ciclos)

        j<-1
        k<-1
        for (i in 1:(total-1)){

                if (dados[i,2]<dados[i+1,2]) {T[k]<-dados[i,1]
                                                S[k]<-dados[i,3]
                                                k<-k+1 }
                else {  falhas[j,1]<-dados[i,1]
                        falhas[j,2]<-dados[i,2]
                        j<-j+1}

                if (i+1==total)         {T[k]<-dados[i+1,1]
                                         S[k]<-dados[i+1,3]}
        }
        if (ciclos==1) {mit<-mi1(T[ciclos])}
        if (ciclos==2) {mit<-mi2(T[ciclos])}
        if (ciclos==3) {mit<-mi3(T[ciclos])}
        if (ciclos==4) {mit<-mi4(T[ciclos])}
        if (ciclos==5) {mit<-mi5(T[ciclos])}
```

```
		tot_falhas<-nrow(falhas)

		for (i in 1:tot_falhas){
			if (falhas[i,2]==1) {CVM_aux[linha2,1]<-mi1(falhas[i,1])
						CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
						linha2<-linha2+1}
			if (falhas[i,2]==2) {CVM_aux[linha2,1]<-mi2(falhas[i,1])
						CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
						linha2<-linha2+1}
			if (falhas[i,2]==3) {CVM_aux[linha2,1]<-mi3(falhas[i,1])
						CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
						linha2<-linha2+1}
			if (falhas[i,2]==4) {CVM_aux[linha2,1]<-mi4(falhas[i,1])
						CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
						linha2<-linha2+1}
			if (falhas[i,2]==5) {CVM_aux[linha2,1]<-mi5(falhas[i,1])
						CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
						linha2<-linha2+1}
		# a coluna 1 da matriz CVM_aux tem as estimativas de mi(t)
		}
}
##########################################################################################
# Formato da matriz do Teste Cramer-von Mises (CVM)
# 1a coluna - Ordem da falha
# 2a Coluna - Ri estimado pela Função mi
# 3a coluna - Ri estimado pela Ordem da falha
# 4a coluna - Diferença ao quadrado 2a e 3a colunas
##########################################################################################
CVM<-matrix(0,M,4)
ord<-order(CVM_aux[,2])

for (i in 1:M) {
		CVM[i,1]<-i
		CVM[i,2]<-CVM_aux[ord[i],2]
		CVM[i,3]<-(((2*CVM[i,1])-1)/(2*M))
		CVM[i,4]<-(CVM[i,2]-CVM[i,3])^2
}

#Estatística do Teste CVM
CVM2<-(1/(12*M))+sum(CVM[,4])

# H0: Modelo segue NHPP com u(t)
# Se CVM2 < Valor Crítico, então aceita H0.

##########################################################################################
# Gráfico para avaliação da Bondade do Ajuste do Modelo
##########################################################################################
# Considera o último tempo de cada ciclo como a MP
# Se o tempo da MP for falha, deve replicar esse valor nos dados_bruto
# Essa duplicação deve ser feita apenas para construção deste gráfico
##########################################################################################
# Entrada de Dados
dados_bruto<-read.table(file="dados.txt", sep = " ")

# Número de sistemas analisados
m<-max(dados_bruto[,4])

# tam: quantidade de dados para cada sistema
# ciclo: quantidade de ciclos para cada sistema
tam<-vector(mode="integer",length=m)
ciclo<-vector(mode="integer",length=m)

# tempo_sist: ultimo tempo do sistema
tempo_sist<-vector(mode="integer",length=m)
```

```
for (k in 1:nrow(dados_bruto)) {
        tam[dados_bruto[k,4]]<-tam[dados_bruto[k,4]]+1
        if ( k==nrow(dados_bruto) ) { ciclo[dados_bruto[k,4]]<-dados_bruto[k,2]
                                        tempo_sist[dados_bruto[k,4]]<-dados_bruto[k,1]}
        else { if (dados_bruto[k,4]<dados_bruto[k+1,4]) {
                ciclo[dados_bruto[k,4]]<-dados_bruto[k,2]
                tempo_sist[dados_bruto[k,4]]<-dados_bruto[k,1]}
        }
}
#Número Total de falhas considerando todos os sistemas
M<-sum(tam)-sum(ciclo)

MCF_aux<-matrix(0,M,m+1)
MCF<-matrix(0,M,4)

info_ciclo<-matrix(0,sum(ciclo),3)
k<-1
linha2<-1
auxMP<-0

# Retirando os tempos de MP dos ciclos ou do sistema e ficando somente as falhas em MCF_aux
# info_ciclo tem a informação dos tempos finais (coluna 1) de cada ciclo (coluna 2) dos sistemas (coluna 3)
for (linha in 1:sum(tam)) {
        if ( linha==nrow(dados_bruto) ) { info_ciclo[linha2,1]<-dados_bruto[linha,1]
                                          info_ciclo[linha2,2]<-dados_bruto[linha,2]
                                          info_ciclo[linha2,3]<-dados_bruto[linha,4]
        }
        else {      if (dados_bruto[linha,2]<dados_bruto[linha+1,2]) {info_ciclo[linha2,1]<-dados_bruto[linha,1]
                                                                       info_ciclo[linha2,2]<-dados_bruto[linha,2]
                                                                       info_ciclo[linha2,3]<-dados_bruto[linha,4]
                                                                       linha2<-linha2+1
                                                                       auxMP<-1
                    }
                    if (dados_bruto[linha,4]<dados_bruto[linha+1,4]) {info_ciclo[linha2,1]<-dados_bruto[linha,1]
                                                                       info_ciclo[linha2,2]<-dados_bruto[linha,2]
                                                                       info_ciclo[linha2,3]<-dados_bruto[linha,4]
                                                                       linha2<-linha2+1
                                                                       auxMP<-1
                    }
                    if (auxMP==0) {MCF_aux[k,1]<-dados_bruto[linha,1]
                                   k<-k+1
                    }
                    auxMP<-0
          }
}
# Cada coluna da matriz MCF_aux (a partir da 2a) representa um sistema
# Determina em que ciclo esta cada tempo de falha da matriz MCF_aux para os diferentes sistemas
for (i in 1:M) {
        linha<-1
        for (sistema in 1:m) {

                    while (info_ciclo[linha,3]==sistema){
                            if(MCF_aux[i,1]>=info_ciclo[linha,1]) {
                                MCF_aux[i,sistema+1]<-info_ciclo[linha,2]+1
                                linha<-linha+1
                            }
                            else { MCF_aux[i,sistema+1]<-info_ciclo[linha,2]
                                while (info_ciclo[linha,3]==sistema){ linha<-linha+1
                                                                      if(linha==sum(ciclo)+1){
                                                                                  sistema<-sistema+1
                                                                                  linha<-1}
                                }
                            }
                    }
        }
}
```

```
##################################################################################
# Definição das funções mi para os 5 possíveis ciclos
##################################################################################
mi1<-function(t){((t^beta.hat)*((1/lambda.hat)^beta.hat))}
mi2<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                    (((t-(a.hat*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))}
mi3<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                    (((T[2]-(a.hat*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))+
                    (((t-(a.hat*T[2]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[2])^beta.hat)*((1/lambda.hat)^beta.hat))}
mi4<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                    (((T[2]-(a.hat*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))+
                    (((T[3]-(a.hat*T[2]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[2])^beta.hat)*((1/lambda.hat)^beta.hat))+
                    (((t-(a.hat*T[3]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[3])^beta.hat)*((1/lambda.hat)^beta.hat))}
mi5<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                    (((T[2]-(a.hat*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))+
                    (((T[3]-(a.hat*T[2]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[2])^beta.hat)*((1/lambda.hat)^beta.hat))+
                    (((T[4]-(a.hat*T[3]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[3])^beta.hat)*((1/lambda.hat)^beta.hat))+
                    (((t-(a.hat*T[4]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[4])^beta.hat)*((1/lambda.hat)^beta.hat))}
##################################################################################
# Cálculo da estimativa do MCF do processo
for (i in 1:M) {

        MCF[i,1]<-MCF_aux[i,1] #Recebe o tempo de falha
        mit<-0
        linha<-1
        for (sistema in 1:m) {

                dados<-matrix(0,tam[sistema],3)

                # Entrada de dados do sistema
                for (k in 1:tam[sistema]) {
                        dados[k,1]<-dados_bruto[linha,1]
                        dados[k,2]<-dados_bruto[linha,2]
                        dados[k,3]<-dados_bruto[linha,3]
                        linha<-linha+1}

                total<-nrow(dados)
                ciclos<-dados[total,2]

                # Cria vetor com os tempos de falha e severidades da MP
                falhas<-matrix(0,(total-ciclos),2)

                # Cria vetor com os tempos de MP e Severidades
                T<-vector(mode="integer",length=ciclos)
                S<-vector(mode="integer",length=ciclos)

                j<-1
                k<-1
                for (q in 1:(total-1)){
                        if (dados[q,2]<dados[q+1,2]) {T[k]<-dados[q,1]
                                                        S[k]<-dados[q,3]
                                                        k<-k+1 }
                        else {  falhas[j,1]<-dados[q,1]
                                falhas[j,2]<-dados[q,2]
                                j<-j+1}

                        if (q+1==total)         {T[k]<-dados[q+1,1]
                                                 S[k]<-dados[q+1,3]}
                }
                # Verficar em que ciclo esta cada sistema, para utilizar a mi correspondente
                if (MCF_aux[i,sistema+1]==1) {mit<-mit+mi1(MCF[i,1])}
                if (MCF_aux[i,sistema+1]==2) {mit<-mit+mi2(MCF[i,1])}
                if (MCF_aux[i,sistema+1]==3) {mit<-mit+mi3(MCF[i,1])}
                if (MCF_aux[i,sistema+1]==4) {mit<-mit+mi4(MCF[i,1])}
                if (MCF_aux[i,sistema+1]==5) {mit<-mit+mi5(MCF[i,1])}
        }
```

```
		MCF[i,2]<-mit/m #Estimativa do MCF do processo
}
ord<-order(MCF[,1])
MCF<-MCF[ord,]

# Cálculo da MCF empírica do processo
for (i in 1:M) {
		ri<-0
		for (sistema in 1:m) {
			if(MCF[i,1]<=tempo_sist[sistema]){ri<-ri+1}
		}
		if(i==1) {MCF[i,3]<-(1/ri)}
		else {MCF[i,3]<-(1/ri)+MCF[i-1,3]}
		MCF[i,4]<-ri
}
# Gráfico MCF estimada X MCF empírica
MCF_estimado<-MCF[,2]
MCF_empirico<-MCF[,3]
plot(MCF_estimado,MCF_empirico)
abline(0,1)

##########################################################################################
# RELATÓRIO FINAL
##########################################################################################

##########################################################################################
# Resumo Verificações
##########################################################################################

# Deve ser zero para indicar que o algoritmo convergiu
convergencia

if(detver1>=0){"Verificar"}
if(detver2<=0){"Verificar"}

if(av$values[1]>=0){"Verificar"}
if(av$values[2]>=0){"Verificar"}

##########################################################################################
# Estimativas
##########################################################################################

lambda.hat
IC_lambda_Inf
IC_lambda_Sup

beta.hat
IC_beta_Inf
IC_beta_Sup

##########################################################################################
# Teste de Razão de Verossimilhanças
##########################################################################################

# P-valor (H0: a=0)
1-pchisq(TRV, 1)

# Estatística TRV
TRV

# Valor Crítico (VC)
qchisq(0.95,1)

# Se TRV < VC, então aceita H0
```

```
##########################################################################################
# Teste Cramer-von Mises
##########################################################################################
#Estatística do Teste CVM
CVM2

# H0: Modelo segue NHPP com u(t)
# Se CVM2 < Valor Crítico, então aceita H0

##########################################################################################
# Otimização Tempo de MP
##########################################################################################
# Custo Manutenção Corretiva
CMC<-1

# custo Manutenção Preventiva
CMP<-5

# Tempo da realização da MP
MP<-T[4]

# Severidade da MP realizada
SMP<-S[4]

# mi=Número de falhas esperadas ultima MP até t
mi<-function(t){((((t-(a.hat*MP))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*MP)^beta.hat)*((1/lambda.hat)^beta.hat))}

# Função Custo Médio
FCM<-function(t) { ((CMC*mi(t+MP))+(CMP))/t }

# Otimização do Tempo para realização de MP
TO<-optimize(FCM, c(0,500))

TO$minimum+MP

TO$objective

##########################################################################################
# Função Confiabilidade
##########################################################################################

R<-function(t){exp(-mi(MP+t))}

##########################################################################################
```

## A.2 - Programa referente ao modelo de Redução de Idade (modelo RI)

```
##################################################################################
# Formato do arquivo de entrada dados.txt
# 1a coluna - Tempos de falha acumulados (o ultimo valor do ciclo deve ser a MP)
# 2a Coluna - Número do ciclo de MP ordenado
# 3a coluna - BSC da MP realizada que influênciará o próximo ciclo de MP
# 4a coluna - Número do sistema ordenado
##################################################################################
library(nlme)
##################################################################################
# Função da LogVerossimilhança para o Modelos de Confiabilidade de Sistemas
# Reparáveis considerando MP Imperfeita
##################################################################################
logvero <- function(x){

	# Entrada dos Parâmetros
	lambda<-x[1]
	beta<-x[2]
	a<-x[3]

	lv<-0

	# Entrada de Dados
	dados_bruto<-read.table(file="dados.txt", sep = " ")

	# Número de Sistemas analisados
	m<-max(dados_bruto[,4])

	tam<-vector(mode="integer",length=m)

	# Define a quantidade de dados para cada sistema
	for (k in 1:nrow(dados_bruto)) {
		tam[dados_bruto[k,4]]<-tam[dados_bruto[k,4]]+1
	}

	linha<-1
	for (sistema in 1:m) {
		dados<-matrix(0,tam[sistema],3)

		# Entrada de dados do sistema k
		for (k in 1:tam[sistema]) {
			dados[k,1]<-dados_bruto[linha,1]
			dados[k,2]<-dados_bruto[linha,2]
			dados[k,3]<-dados_bruto[linha,3]
			linha<-linha+1
		}
		total<-nrow(dados)
		ciclos<-dados[total,2]

		n<-vector(mode="integer",length=ciclos)
		T<-vector(mode="integer",length=ciclos)
		S<-vector(mode="integer",length=ciclos)

		# Cria vetor com os tempos e severidade da MP em cada ciclo
		# e o número de falhas em cada cilco
		for (i in 1:(total-1)){

			if (dados[i,2]<dados[i+1,2]) {T[dados[i,2]]<-dados[i,1]
								S[dados[i,2]]<-dados[i,3]}

			else {n[dados[i,2]]<-n[dados[i,2]]+1}

		}
```

```
		T[dados[i+1,2]]<-dados[i+1,1]
		S[dados[i+1,2]]<-dados[i+1,3]

		# Número total de falhas
		N<-sum(n)

		# Cria vetor com os tempos de falha e severidades da MP
		t<-matrix(0,ciclos,max(n))

		j<-1
		for (i in 1:(total-1)){
			if (dados[i,2]<dados[i+1,2]) { j<-1 }

			else {t[dados[i,2],j]<-dados[i,1]
			     j<-j+1}
		}

		#######################################################################
		# Cálculo da LogVerossimilhança
		#######################################################################
		lv<-lv+(N*log(beta))+(N*log(((1/lambda)^beta)))

		Va<-0 #Idade Virtual inicial

		j<-1
		aux<-0
		for (i in 1:n[j]){
			if(n[j]>0) {aux<-aux+log(t[j,i])}
		}

		lv<-lv+((beta-1)*aux)
		lv<-lv-(((1/lambda)^beta)*(T[j]^beta))

		Va<-T[j]

		for (j in 2:ciclos){
			aux<-0
			for (i in 1:n[j]){
				if(n[j]>0) {aux<-aux+log(t[j,i]-(a*Va))}
			}

			lv<-lv+((beta-1)*aux)
			lv<-lv-(((1/lambda)^beta)*(((T[j]-(a*Va))^beta)-(((1-a)*Va)^beta)))

			Va<-T[j]
		}
		#######################################################################
	}
	restricao<-0
	for (j in 1:ciclos){
		if( a < 0 ) {restricao<-1}
		if( a > 1 ) {restricao<-1}

	}

	if(restricao==1) {return(-Inf)} else{return(lv)}
}
```

```
##########################################################################################
# O método da função optim de otimização encontra o mínimo, portanto para determinar
# o máximo da função LogVerossimilhança foi utilizado seu valor negativo
##########################################################################################
logvero_opt <- function(x){

        lambda<-x[1]
        beta<-x[2]
        a<-x[3]
        return(-logvero(c(lambda,beta,a)))
}

##########################################################################################
# Otimização da LogVerossimilhança
##########################################################################################
mle<-optim(c(100, 1.8, 0.5), logvero_opt, method = "Nelder-Mead")
##########################################################################################
# PARÂMETROS ESTIMADOS
lambda.hat<-mle$par[1]
beta.hat<-mle$par[2]
a.hat<-mle$par[3]

param<-c(mle$par[1], mle$par[2], mle$par[3])

# Deve ser zero para indicar que o algoritmo convergiu
convergencia<-mle$convergence

mat<-fdHess(mle$par, logvero)
hessiano<-mat$Hessian

# Para ser ponto de máximo devem ser alternados os sinais

# VERIFICAÇÃO 1 (Negativo)
detver1<-hessiano[1,1]

#VERIFICAÇÃO 2 (Positivo)
ver2<-matrix(0,2,2)
ver2[1,1]<-hessiano[1,1]
ver2[1,2]<-hessiano[1,2]
ver2[2,1]<-hessiano[2,1]
ver2[2,2]<-hessiano[2,2]

detver2<-det(ver2)

#VERIFICAÇÃO 3 (Negativo)
ver3<-matrix(0,3,3)
ver3[1,1]<-hessiano[1,1]
ver3[1,2]<-hessiano[1,2]
ver3[1,3]<-hessiano[1,3]
ver3[2,1]<-hessiano[2,1]
ver3[2,2]<-hessiano[2,2]
ver3[2,3]<-hessiano[2,3]
ver3[3,1]<-hessiano[3,1]
ver3[3,2]<-hessiano[3,2]
ver3[3,3]<-hessiano[3,3]

detver3<-det(ver3)

# Outra verificação: se os Autovalores forem todos negativos, é ponto de máximo.
# VERIFICAÇÃO AUTOVALORES
av<-eigen(hessiano)
```

```
##########################################################################################
# Resumo Verificações
##########################################################################################

if(detver1<0){"OK"}else{"Verificar"}
if(detver2>0){"OK"}else{"Verificar"}
if(detver3<0){"OK"}else{"Verificar"}

if(av$values[1]<0){"OK"}else{"Verificar"}
if(av$values[2]<0){"OK"}else{"Verificar"}
if(av$values[3]<0){"OK"}else{"Verificar"}

##########################################################################################
# Desvios Padrãoes das estimativas
##########################################################################################
dp.lambda.hat<-sqrt(solve(-hessiano)[1,1])
dp.beta.hat<-sqrt(solve(-hessiano)[2,2])
dp.a.hat<-sqrt(solve(-hessiano)[3,3])

# Nível de Confiança
NC<-0.95

IC_lambda_Inf<-lambda.hat/(exp(qnorm((1+NC)/2)*dp.lambda.hat/lambda.hat))
IC_lambda_Sup<-lambda.hat*(exp(qnorm((1+NC)/2)*dp.lambda.hat/lambda.hat))

IC_beta_Inf<-beta.hat/(exp(qnorm((1+NC)/2)*dp.beta.hat/beta.hat))
IC_beta_Sup<-beta.hat*(exp(qnorm((1+NC)/2)*dp.beta.hat/beta.hat))

IC_a_Inf<-a.hat/(exp(qnorm((1+NC)/2)*dp.a.hat/a.hat))
IC_a_Sup<-a.hat*(exp(qnorm((1+NC)/2)*dp.a.hat/a.hat))

##########################################################################################
# Verossimilhança da Manutenção Sem Efeito (MSE)
##########################################################################################

logvero_mse <- function(x){

        lambda<-x[1]
        beta<-x[2]
        a<-0
        return(-logvero(c(lambda,beta,a)))
}

mle_mse<-optim(c(100, 1.5), logvero_mse, method = "Nelder-Mead")
param_mse<-c(mle_mse$par[1], mle_mse$par[2])

##########################################################################################
# Teste de Razão de Verossimilhanças
##########################################################################################
TRV<-(-2*(logvero(c(param_mse,0))-(logvero(param))))

# P-valor (H0: a=0)
1-pchisq(TRV, 1)

# Estatística TRV
TRV

# Valor Crítico (VC)
qchisq(0.95,1)

# Se TRV < VC, então aceita H0 (não há efeito entre as MP)
```

```
##########################################################################################
# Teste Cramer-von Mises
##########################################################################################
# Considera o último tempo de cada ciclo como a MP
# É considerado nos cálculos se o último tempo do sistema for falha ou MP
##########################################################################################
# Entrada de Dados
dados_bruto<-read.table(file="dados.txt", sep = " ")

# Número de sistemas analisados
m<-max(dados_bruto[,4])

# tam: quantidade de dados para cada sistema
# ciclo: quantidade de ciclos para cada sistema
tam<-vector(mode="integer",length=m)
ciclo<-vector(mode="integer",length=m)

for (k in 1:nrow(dados_bruto)) {
	tam[dados_bruto[k,4]]<-tam[dados_bruto[k,4]]+1
	if ( k==nrow(dados_bruto) ) { ciclo[dados_bruto[k,4]]<-dados_bruto[k,2] }
	else { if (dados_bruto[k,4]<dados_bruto[k+1,4]) { ciclo[dados_bruto[k,4]]<-dados_bruto[k,2] }}
}

#Número Total de falhas considerando todos os sistemas
M<-sum(tam)-sum(ciclo)

##########################################################################################

CVM_aux<-matrix(0,M,2)

##########################################################################################
# Definição das funções mi para os 5 possíveis ciclos
##########################################################################################

mi1<-function(t){((t^beta.hat)*((1/lambda.hat)^beta.hat))}
mi2<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
		(((t-(a.hat*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))}
mi3<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
		(((T[2]-(a.hat*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))+
		(((t-(a.hat*T[2]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[2])^beta.hat)*((1/lambda.hat)^beta.hat))}
mi4<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
		(((T[2]-(a.hat*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))+
		(((T[3]-(a.hat*T[2]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[2])^beta.hat)*((1/lambda.hat)^beta.hat))+
		(((t-(a.hat*T[3]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[3])^beta.hat)*((1/lambda.hat)^beta.hat))}
mi5<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
		(((T[2]-(a.hat*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))+
		(((T[3]-(a.hat*T[2]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[2])^beta.hat)*((1/lambda.hat)^beta.hat))+
		(((T[4]-(a.hat*T[3]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[3])^beta.hat)*((1/lambda.hat)^beta.hat))+
		(((t-(a.hat*T[4]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[4])^beta.hat)*((1/lambda.hat)^beta.hat))}
##########################################################################################
linha<-1
linha2<-1

for (sistema in 1:m) {
	dados<-matrix(0,tam[sistema],3)

	# Entrada de dados do sistema
	for (k in 1:tam[sistema]) {
		dados[k,1]<-dados_bruto[linha,1]
		dados[k,2]<-dados_bruto[linha,2]
		dados[k,3]<-dados_bruto[linha,3]
		linha<-linha+1}

	total<-nrow(dados)
	ciclos<-dados[total,2]
```

```
        # Cria vetor com os tempos de falha e severidades da MP
        falhas<-matrix(0,(total-ciclos),2)

        # Cria vetor com os tempos de MP e Severidades
        T<-vector(mode="integer",length=ciclos)
        S<-vector(mode="integer",length=ciclos)

        j<-1
        k<-1

        for (i in 1:(total-1)){
                if (dados[i,2]<dados[i+1,2]) {T[k]<-dados[i,1]
                                              S[k]<-dados[i,3]
                                              k<-k+1 }
                else {  falhas[j,1]<-dados[i,1]
                        falhas[j,2]<-dados[i,2]
                        j<-j+1}

                if (i+1==total)         {T[k]<-dados[i+1,1]
                                         S[k]<-dados[i+1,3]}
        }

        if (ciclos==1) {mit<-mi1(T[ciclos])}
        if (ciclos==2) {mit<-mi2(T[ciclos])}
        if (ciclos==3) {mit<-mi3(T[ciclos])}
        if (ciclos==4) {mit<-mi4(T[ciclos])}
        if (ciclos==5) {mit<-mi5(T[ciclos])}

        tot_falhas<-nrow(falhas)

        for (i in 1:tot_falhas){
                if (falhas[i,2]==1) {CVM_aux[linha2,1]<-mi1(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}
                if (falhas[i,2]==2) {CVM_aux[linha2,1]<-mi2(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}
                if (falhas[i,2]==3) {CVM_aux[linha2,1]<-mi3(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}
                if (falhas[i,2]==4) {CVM_aux[linha2,1]<-mi4(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}
                if (falhas[i,2]==5) {CVM_aux[linha2,1]<-mi5(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}

        # a coluna 1 da matriz CVM_aux tem as estimativas de mi(t)
        }
}
##########################################################################################
# Formato da matriz do Teste Cramer-von Mises (CVM)
# 1a coluna - Ordem da falha
# 2a Coluna - Ri estimado pela Função mi
# 3a coluna - Ri estimado pela Ordem da falha
# 4a coluna - Diferença ao quadrado 2a e 3a colunas
##########################################################################################
CVM<-matrix(0,M,4)
ord<-order(CVM_aux[,2])

for (i in 1:M) {
        CVM[i,1]<-i
        CVM[i,2]<-CVM_aux[ord[i],2]
        CVM[i,3]<-(((2*CVM[i,1])-1)/(2*M))
        CVM[i,4]<-(CVM[i,2]-CVM[i,3])^2
}
```

```
#Estatística do Teste CVM
CVM2<-(1/(12*M))+sum(CVM[,4])

# H0: Modelo segue NHPP com u(t)
# Se CVM2 < Valor Crítico, então aceita H0.

##################################################################################
# Gráfico para avaliação da Bondade do Ajuste do Modelo
##################################################################################
# Considera o último tempo de cada ciclo como a MP
# Se o tempo da MP for falha, deve replicar esse valor nos dados_bruto
# Essa duplicação deve ser feita apenas para construção deste gráfico
##################################################################################
# Entrada de Dados
dados_bruto<-read.table(file="dados.txt", sep = " ")

# Número de sistemas analisados
m<-max(dados_bruto[,4])

# tam: quantidade de dados para cada sistema
# ciclo: quantidade de ciclos para cada sistema
tam<-vector(mode="integer",length=m)
ciclo<-vector(mode="integer",length=m)

# tempo_sist: ultimo tempo do sistema
tempo_sist<-vector(mode="integer",length=m)

for (k in 1:nrow(dados_bruto)) {
	tam[dados_bruto[k,4]]<-tam[dados_bruto[k,4]]+1
	if ( k==nrow(dados_bruto) ) { ciclo[dados_bruto[k,4]]<-dados_bruto[k,2]
						tempo_sist[dados_bruto[k,4]]<-dados_bruto[k,1]}
	else { if (dados_bruto[k,4]<dados_bruto[k+1,4]) {
			ciclo[dados_bruto[k,4]]<-dados_bruto[k,2]
			tempo_sist[dados_bruto[k,4]]<-dados_bruto[k,1]}
	}
}
#Número Total de falhas considerando todos os sistemas
M<-sum(tam)-sum(ciclo)

MCF_aux<-matrix(0,M,m+1)
MCF<-matrix(0,M,4)

info_ciclo<-matrix(0,sum(ciclo),3)
k<-1
linha2<-1
auxMP<-0

# Retirando os tempos de MP dos ciclos ou do sistema e ficando somente as falhas em MCF_aux
# info_ciclo tem a informação dos tempos finais (coluna 1) de cada ciclo (coluna 2) dos sistemas (coluna 3)
for (linha in 1:sum(tam)) {
	if ( linha==nrow(dados_bruto) ) { info_ciclo[linha2,1]<-dados_bruto[linha,1]
						info_ciclo[linha2,2]<-dados_bruto[linha,2]
						info_ciclo[linha2,3]<-dados_bruto[linha,4]
	}
	else {	if (dados_bruto[linha,2]<dados_bruto[linha+1,2]) {info_ciclo[linha2,1]<-dados_bruto[linha,1]
								info_ciclo[linha2,2]<-dados_bruto[linha,2]
								info_ciclo[linha2,3]<-dados_bruto[linha,4]
								linha2<-linha2+1
								auxMP<-1
		}
		if (dados_bruto[linha,4]<dados_bruto[linha+1,4]) {info_ciclo[linha2,1]<-dados_bruto[linha,1]
								info_ciclo[linha2,2]<-dados_bruto[linha,2]
								info_ciclo[linha2,3]<-dados_bruto[linha,4]
								linha2<-linha2+1
								auxMP<-1
		}
```

```
                    if (auxMP==0) {MCF_aux[k,1]<-dados_bruto[linha,1]
                                k<-k+1
                    }
                    auxMP<-0
          }
}

# Cada coluna da matriz MCF_aux (a partir da 2a) representa um sistema
# Determina em que ciclo esta cada tempo de falha da matriz MCF_aux para os diferentes sistemas
for (i in 1:M) {
          linha<-1
          for (sistema in 1:m) {
                    while (info_ciclo[linha,3]==sistema){
                              if(MCF_aux[i,1]>=info_ciclo[linha,1]) {
                                   MCF_aux[i,sistema+1]<-info_ciclo[linha,2]+1
                                   linha<-linha+1
                              }
                              else { MCF_aux[i,sistema+1]<-info_ciclo[linha,2]
                                   while (info_ciclo[linha,3]==sistema){ linha<-linha+1
                                                                        if(linha==sum(ciclo)+1){
                                                                                  sistema<-sistema+1
                                                                                  linha<-1}
                                   }
                              }
                    }
          }
}

##########################################################################################
# Definição das funções mi para os 5 possíveis ciclos
##########################################################################################

mi1<-function(t){((t^beta.hat)*((1/lambda.hat)^beta.hat))}

mi2<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                    (((t-(a.hat*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))}
mi3<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                    (((T[2]-(a.hat*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))+
                    (((t-(a.hat*T[2]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[2])^beta.hat)*((1/lambda.hat)^beta.hat))}
mi4<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                    (((T[2]-(a.hat*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))+
                    (((T[3]-(a.hat*T[2]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[2])^beta.hat)*((1/lambda.hat)^beta.hat))+
                    (((t-(a.hat*T[3]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[3])^beta.hat)*((1/lambda.hat)^beta.hat))}
mi5<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                    (((T[2]-(a.hat*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))+
                    (((T[3]-(a.hat*T[2]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[2])^beta.hat)*((1/lambda.hat)^beta.hat))+
                    (((T[4]-(a.hat*T[3]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[3])^beta.hat)*((1/lambda.hat)^beta.hat))+
                    (((t-(a.hat*T[4]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*T[4])^beta.hat)*((1/lambda.hat)^beta.hat))}

##########################################################################################
# Cálculo da estimativa do MCF do processo
for (i in 1:M) {
          MCF[i,1]<-MCF_aux[i,1] #Recebe o tempo de falha
          mit<-0
          linha<-1

          for (sistema in 1:m) {
                    dados<-matrix(0,tam[sistema],3)

                    # Entrada de dados do sistema
                    for (k in 1:tam[sistema]) {
                              dados[k,1]<-dados_bruto[linha,1]
                              dados[k,2]<-dados_bruto[linha,2]
                              dados[k,3]<-dados_bruto[linha,3]
                              linha<-linha+1}
```

```
                    total<-nrow(dados)
                    ciclos<-dados[total,2]

                    # Cria vetor com os tempos de falha e severidades da MP
                    falhas<-matrix(0,(total-ciclos),2)

                    # Cria vetor com os tempos de MP e Severidades
                    T<-vector(mode="integer",length=ciclos)
                    S<-vector(mode="integer",length=ciclos)

                    j<-1
                    k<-1

                    for (q in 1:(total-1)){
                              if (dados[q,2]<dados[q+1,2]) {T[k]<-dados[q,1]
                                                            S[k]<-dados[q,3]
                                                            k<-k+1 }
                              else {  falhas[j,1]<-dados[q,1]
                                      falhas[j,2]<-dados[q,2]
                                      j<-j+1}
                              if (q+1==total)        {T[k]<-dados[q+1,1]
                                                      S[k]<-dados[q+1,3]}
                    }

                    # Verficar em que ciclo esta cada sistema, para utilizar a mi correspondente
                    if (MCF_aux[i,sistema+1]==1) {mit<-mit+mi1(MCF[i,1])}
                    if (MCF_aux[i,sistema+1]==2) {mit<-mit+mi2(MCF[i,1])}
                    if (MCF_aux[i,sistema+1]==3) {mit<-mit+mi3(MCF[i,1])}
                    if (MCF_aux[i,sistema+1]==4) {mit<-mit+mi4(MCF[i,1])}
                    if (MCF_aux[i,sistema+1]==5) {mit<-mit+mi5(MCF[i,1])}
          }
          MCF[i,2]<-mit/m #Estimativa do MCF do processo
}
ord<-order(MCF[,1])
MCF<-MCF[ord,]

# Cálculo da MCF empírica do processo
for (i in 1:M) {

          ri<-0

          for (sistema in 1:m) {
                    if(MCF[i,1]<=tempo_sist[sistema]){ri<-ri+1}
          }
          if(i==1) {MCF[i,3]<-(1/ri)}
          else {MCF[i,3]<-(1/ri)+MCF[i-1,3]}
          MCF[i,4]<-ri
}
# Gráfico MCF estimada X MCF empírica
MCF_estimado<-MCF[,2]
MCF_empirico<-MCF[,3]
plot(MCF_estimado,MCF_empirico)
abline(0,1)

##########################################################################################
# RELATÓRIO FINAL
##########################################################################################
##########################################################################################
# Resumo Verificações
##########################################################################################
# Deve ser zero para indicar que o algoritmo convergiu
convergencia

if(detver1>=0){"Verificar"}
if(detver2<=0){"Verificar"}
if(detver3>=0){"Verificar"}
```

```
if(av$values[1]>=0){"Verificar"}
if(av$values[2]>=0){"Verificar"}
if(av$values[3]>=0){"Verificar"}

##################################################################################
# Estimativas
##################################################################################

lambda.hat
IC_lambda_Inf
IC_lambda_Sup

beta.hat
IC_beta_Inf
IC_beta_Sup

a.hat
IC_a_Inf
IC_a_Sup

##################################################################################
# Teste de Razão de Verossimilhanças
##################################################################################

# P-valor (H0: a=0)
1-pchisq(TRV, 1)

# Estatística TRV
TRV

# Valor Crítico (VC)
qchisq(0.95,1)

# Se TRV < VC, então aceita H0

##################################################################################
# Teste Cramer-von Mises
##################################################################################

#Estatística do Teste CVM
CVM2

# H0: Modelo segue NHPP com u(t)
# Se CVM2 < Valor Crítico, então aceita H0

##################################################################################
# Otimização Tempo de MP
##################################################################################

# Custo Manutenção Corretiva
CMC<-1

# custo Manutenção Preventiva
CMP<-5

# Tempo da realização da MP
MP<-T[4]

# Severidade da MP realizada
SMP<-S[4]

# mi=Número de falhas esperadas ultima MP até t
mi<-function(t){((((t-(a.hat*MP))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-a.hat)*MP)^beta.hat)*((1/lambda.hat)^beta.hat))}
```

```
# Função Custo Médio
FCM<-function(t) { ((CMC*mi(t+MP))+(CMP))/t }

# Otimização do Tempo para realização de MP
TO<-optimize(FCM, c(0,500))

TO$minimum+MP

TO$objective

##########################################################################################
# Função Confiabilidade
##########################################################################################

R<-function(t){exp(-mi(MP+t))}

##########################################################################################
```

## A.3 - Programa referente ao modelo de Redução de Idade com relação Exponencial (modelo RIE)

```
##################################################################################
# Formato do arquivo de entrada dados.txt
# 1a coluna - Tempos de falha acumulados (o ultimo valor do ciclo deve ser a MP)
# 2a Coluna - Número do ciclo de MP ordenado
# 3a coluna - BSC da MP realizada que influênciará o próximo ciclo de MP
# 4a coluna - Número do sistema ordenado
##################################################################################
library(nlme)
##################################################################################
# Função da LogVerossimilhança para o Modelos de Confiabilidade de Sistemas
# Reparáveis considerando MP Imperfeita
##################################################################################
logvero <- function(x){

	# Entrada dos Parâmetros
	lambda<-x[1]
	beta<-x[2]
	theta<-x[3]

	lv<-0

	# Entrada de Dados
	dados_bruto<-read.table(file="dados.txt", sep = " ")

	# Número de Sistemas analisados
	m<-max(dados_bruto[,4])

	tam<-vector(mode="integer",length=m)

	# Define a quantidade de dados para cada sistema
	for (k in 1:nrow(dados_bruto)) {
		tam[dados_bruto[k,4]]<-tam[dados_bruto[k,4]]+1
	}

	linha<-1

	for (sistema in 1:m) {

		dados<-matrix(0,tam[sistema],3)

		# Entrada de dados do sistema k
		for (k in 1:tam[sistema]) {
			dados[k,1]<-dados_bruto[linha,1]
			dados[k,2]<-dados_bruto[linha,2]
			dados[k,3]<-dados_bruto[linha,3]
			linha<-linha+1		}

		total<-nrow(dados)

		ciclos<-dados[total,2]

		n<-vector(mode="integer",length=ciclos)
		T<-vector(mode="integer",length=ciclos)
		S<-vector(mode="integer",length=ciclos)

		# Cria vetor com os tempos e severidade da MP em cada ciclo
		# e o número de falhas em cada cilco
		for (i in 1:(total-1)){

			if (dados[i,2]<dados[i+1,2]) {T[dados[i,2]]<-dados[i,1]
								S[dados[i,2]]<-dados[i,3]}
```

```
                else {n[dados[i,2]]<-n[dados[i,2]]+1}

        }

        T[dados[i+1,2]]<-dados[i+1,1]
        S[dados[i+1,2]]<-dados[i+1,3]

        # Número total de falhas
        N<-sum(n)

        # Cria vetor com os tempos de falha e severidades da MP
        t<-matrix(0,ciclos,max(n))

        j<-1
        for (i in 1:(total-1)){

                if (dados[i,2]<dados[i+1,2]) { j<-1 }

                else {t[dados[i,2],j]<-dados[i,1]
                    j<-j+1}
        }
        ##########################################################################
        # Cálculo da LogVerossimilhança
        ##########################################################################
        lv<-lv+(N*log(beta))+(N*log(((1/lambda)^beta)))

        Va<-0 #Idade Virtual inicial

        j<-1
        aux<-0
        for (i in 1:n[j]){
                if(n[j]>0) {aux<-aux+log(t[j,i])}
        }

        lv<-lv+((beta-1)*aux)

        lv<-lv-(((1/lambda)^beta)*(T[j]^beta))

        Va<-T[j]

        for (j in 2:ciclos){
                aux<-0
                for (i in 1:n[j]){
                        if(n[j]>0) {aux<-aux+log(t[j,i]-((1-exp(-S[j-1]*theta))*Va))}
                }

                lv<-lv+((beta-1)*aux)
                lv<-lv-(((1/lambda)^beta)*(((T[j]-((1-exp(-S[j-1]*theta))*Va))^beta)-(((1-(1-exp(-S[j-
1]*theta)))*Va)^beta)))

                Va<-T[j]
        }
        ##########################################################################
    }

    restricao<-0
    for (j in 1:ciclos){
        if( (1-exp(-S[j]*theta)) < 0 ) {restricao<-1}
        if( (1-exp(-S[j]*theta)) > 1 ) {restricao<-1}

    }

    if(restricao==1) {return(-Inf)} else{return(lv)}

}
```

```
##########################################################################################
# O método da função optim de otimização encontra o mínimo, portanto para determinar
# o máximo da função LogVerossimilhança foi utilizado seu valor negativo
##########################################################################################
logvero_opt <- function(x){
        lambda<-x[1]
        beta<-x[2]
        theta<-x[3]
        return(-logvero(c(lambda,beta,theta)))
}

##########################################################################################
# Otimização da LogVerossimilhança
##########################################################################################
mle<-optim(c(100, 1.8, 0.8), logvero_opt, method = "Nelder-Mead")
##########################################################################################
# PARÂMETROS ESTIMADOS
lambda.hat<-mle$par[1]
beta.hat<-mle$par[2]
theta.hat<-mle$par[3]

param<-c(mle$par[1], mle$par[2], mle$par[3])

# Deve ser zero para indicar que o algoritmo convergiu
convergencia<-mle$convergence

mat<-fdHess(mle$par, logvero)
hessiano<-mat$Hessian

# Para ser ponto de máximo devem ser alternados os sinais

# VERIFICAÇÃO 1 (Negativo)
detver1<-hessiano[1,1]

#VERIFICAÇÃO 2 (Positivo)
ver2<-matrix(0,2,2)
ver2[1,1]<-hessiano[1,1]
ver2[1,2]<-hessiano[1,2]
ver2[2,1]<-hessiano[2,1]
ver2[2,2]<-hessiano[2,2]

detver2<-det(ver2)

#VERIFICAÇÃO 3 (Negativo)
ver3<-matrix(0,3,3)
ver3[1,1]<-hessiano[1,1]
ver3[1,2]<-hessiano[1,2]
ver3[1,3]<-hessiano[1,3]
ver3[2,1]<-hessiano[2,1]
ver3[2,2]<-hessiano[2,2]
ver3[2,3]<-hessiano[2,3]
ver3[3,1]<-hessiano[3,1]
ver3[3,2]<-hessiano[3,2]
ver3[3,3]<-hessiano[3,3]

detver3<-det(ver3)

# Outra verificação: se os Autovalores forem todos negativos, é ponto de máximo.
# VERIFICAÇÃO AUTOVALORES
av<-eigen(hessiano)
```

```
##########################################################################################
# Resumo Verificações
##########################################################################################

if(detver1<0){"OK"}else{"Verificar"}
if(detver2>0){"OK"}else{"Verificar"}
if(detver3<0){"OK"}else{"Verificar"}

if(av$values[1]<0){"OK"}else{"Verificar"}
if(av$values[2]<0){"OK"}else{"Verificar"}
if(av$values[3]<0){"OK"}else{"Verificar"}

##########################################################################################
# Desvios Padrãoes das estimativas
##########################################################################################
dp.lambda.hat<-sqrt(solve(-hessiano)[1,1])
dp.beta.hat<-sqrt(solve(-hessiano)[2,2])
dp.theta.hat<-sqrt(solve(-hessiano)[3,3])

# Nível de Confiança
NC<-0.95

IC_lambda_Inf<-lambda.hat/(exp(qnorm((1+NC)/2)*dp.lambda.hat/lambda.hat))
IC_lambda_Sup<-lambda.hat*(exp(qnorm((1+NC)/2)*dp.lambda.hat/lambda.hat))

IC_beta_Inf<-beta.hat/(exp(qnorm((1+NC)/2)*dp.beta.hat/beta.hat))
IC_beta_Sup<-beta.hat*(exp(qnorm((1+NC)/2)*dp.beta.hat/beta.hat))

IC_theta_Inf<-theta.hat/(exp(qnorm((1+NC)/2)*dp.theta.hat/theta.hat))
IC_theta_Sup<-theta.hat*(exp(qnorm((1+NC)/2)*dp.theta.hat/theta.hat))

##########################################################################################
# Verossimilhança da Manutenção Sem Efeito (MSE)
##########################################################################################
logvero_mse <- function(x){
        lambda<-x[1]
        beta<-x[2]
        theta<-0
        return(-logvero(c(lambda,beta,theta)))
}

mle_mse<-optim(c(100, 1.5), logvero_mse, method = "Nelder-Mead")
param_mse<-c(mle_mse$par[1], mle_mse$par[2])

##########################################################################################
# Teste de Razão de Verossimilhanças
##########################################################################################
TRV<-(-2*(logvero(c(param_mse,0))-(logvero(param))))

# P-valor (H0: theta=0)
1-pchisq(TRV, 1)

# Estatística TRV
TRV

# Valor Crítico (VC)
qchisq(0.95,1)

# Se TRV < VC, então aceita H0 (não há efeito entre as MP)
```

```
##########################################################################################
# Teste Cramer-von Mises
##########################################################################################
# Considera o último tempo de cada ciclo como a MP
# É considerado nos cálculos se o último tempo do sistema for falha ou MP
##########################################################################################
# Entrada de Dados
dados_bruto<-read.table(file="dados.txt", sep = " ")

# Número de sistemas analisados
m<-max(dados_bruto[,4])

# tam: quantidade de dados para cada sistema
# ciclo: quantidade de ciclos para cada sistema
tam<-vector(mode="integer",length=m)
ciclo<-vector(mode="integer",length=m)

for (k in 1:nrow(dados_bruto)) {
		tam[dados_bruto[k,4]]<-tam[dados_bruto[k,4]]+1
		if ( k==nrow(dados_bruto) ) { ciclo[dados_bruto[k,4]]<-dados_bruto[k,2] }
		else { if (dados_bruto[k,4]<dados_bruto[k+1,4]) { ciclo[dados_bruto[k,4]]<-dados_bruto[k,2] }}
}

#Número Total de falhas considerando todos os sistemas
M<-sum(tam)-sum(ciclo)

##########################################################################################

CVM_aux<-matrix(0,M,2)

##########################################################################################
# Definição das funções mi para os 5 possíveis ciclos
##########################################################################################

mi1<-function(t){((t^beta.hat)*((1/lambda.hat)^beta.hat))}

mi2<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
				(((t-((1-exp(-S[1]*theta.hat))*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-(1-exp(-
S[1]*theta.hat)))*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))}

mi3<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
				(((T[2]-((1-exp(-S[1]*theta.hat))*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-(1-exp(-
S[1]*theta.hat)))*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))+
				(((t-((1-exp(-S[2]*theta.hat))*T[2]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-(1-exp(-
S[2]*theta.hat)))*T[2])^beta.hat)*((1/lambda.hat)^beta.hat))}

mi4<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
				(((T[2]-((1-exp(-S[1]*theta.hat))*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-(1-exp(-
S[1]*theta.hat)))*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))+
				(((T[3]-((1-exp(-S[2]*theta.hat))*T[2]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-(1-exp(-
S[2]*theta.hat)))*T[2])^beta.hat)*((1/lambda.hat)^beta.hat))+
				(((t-((1-exp(-S[3]*theta.hat))*T[3]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-(1-exp(-
S[3]*theta.hat)))*T[3])^beta.hat)*((1/lambda.hat)^beta.hat))}

mi5<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
				(((T[2]-((1-exp(-S[1]*theta.hat))*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-(1-exp(-
S[1]*theta.hat)))*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))+
				(((T[3]-((1-exp(-S[2]*theta.hat))*T[2]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-(1-exp(-
S[2]*theta.hat)))*T[2])^beta.hat)*((1/lambda.hat)^beta.hat))+
				(((T[4]-((1-exp(-S[3]*theta.hat))*T[3]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-(1-exp(-
S[3]*theta.hat)))*T[3])^beta.hat)*((1/lambda.hat)^beta.hat))+
				(((t-((1-exp(-S[4]*theta.hat))*T[4]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-(1-exp(-
S[4]*theta.hat)))*T[4])^beta.hat)*((1/lambda.hat)^beta.hat))}

##########################################################################################
```

```
linha<-1
linha2<-1

for (sistema in 1:m) {
        dados<-matrix(0,tam[sistema],3)
        # Entrada de dados do sistema
        for (k in 1:tam[sistema]) {
                dados[k,1]<-dados_bruto[linha,1]
                dados[k,2]<-dados_bruto[linha,2]
                dados[k,3]<-dados_bruto[linha,3]
                linha<-linha+1}
        total<-nrow(dados)
        ciclos<-dados[total,2]

        # Cria vetor com os tempos de falha e severidades da MP
        falhas<-matrix(0,(total-ciclos),2)

        # Cria vetor com os tempos de MP e Severidades
        T<-vector(mode="integer",length=ciclos)
        S<-vector(mode="integer",length=ciclos)

        j<-1
        k<-1

        for (i in 1:(total-1)){
                if (dados[i,2]<dados[i+1,2]) {T[k]<-dados[i,1]
                                              S[k]<-dados[i,3]
                                              k<-k+1 }
                else {  falhas[j,1]<-dados[i,1]
                        falhas[j,2]<-dados[i,2]
                        j<-j+1}

                if (i+1==total)         {T[k]<-dados[i+1,1]
                                         S[k]<-dados[i+1,3]}
        }

        if (ciclos==1) {mit<-mi1(T[ciclos])}
        if (ciclos==2) {mit<-mi2(T[ciclos])}
        if (ciclos==3) {mit<-mi3(T[ciclos])}
        if (ciclos==4) {mit<-mi4(T[ciclos])}
        if (ciclos==5) {mit<-mi5(T[ciclos])}

        tot_falhas<-nrow(falhas)

        for (i in 1:tot_falhas){

                if (falhas[i,2]==1) {CVM_aux[linha2,1]<-mi1(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}
                if (falhas[i,2]==2) {CVM_aux[linha2,1]<-mi2(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}
                if (falhas[i,2]==3) {CVM_aux[linha2,1]<-mi3(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}
                if (falhas[i,2]==4) {CVM_aux[linha2,1]<-mi4(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}
                if (falhas[i,2]==5) {CVM_aux[linha2,1]<-mi5(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}

        # a coluna 1 da matriz CVM_aux tem as estimativas de mi(t)

        }
}
```

```
##########################################################################################
# Formato da matriz do Teste Cramer-von Mises (CVM)
# 1a coluna - Ordem da falha
# 2a Coluna - Ri estimado pela Função mi
# 3a coluna - Ri estimado pela Ordem da falha
# 4a coluna - Diferença ao quadrado 2a e 3a colunas
##########################################################################################

CVM<-matrix(0,M,4)

ord<-order(CVM_aux[,2])

for (i in 1:M) {
        CVM[i,1]<-i
        CVM[i,2]<-CVM_aux[ord[i],2]
        CVM[i,3]<-(((2*CVM[i,1])-1)/(2*M))
        CVM[i,4]<-(CVM[i,2]-CVM[i,3])^2
}

#Estatística do Teste CVM
CVM2<-(1/(12*M))+sum(CVM[,4])

# H0: Modelo segue NHPP com u(t)
# Se CVM2 < Valor Crítico, então aceita H0.

##########################################################################################
# Gráfico para avaliação da Bondade do Ajuste do Modelo
##########################################################################################
# Considera o último tempo de cada ciclo como a MP
# Se o tempo da MP for falha, deve replicar esse valor nos dados_bruto
# Essa duplicação deve ser feita apenas para construção deste gráfico
##########################################################################################

# Entrada de Dados
dados_bruto<-read.table(file="dados.txt", sep = " ")

# Número de sistemas analisados
m<-max(dados_bruto[,4])

# tam: quantidade de dados para cada sistema
# ciclo: quantidade de ciclos para cada sistema
tam<-vector(mode="integer",length=m)
ciclo<-vector(mode="integer",length=m)

# tempo_sist: ultimo tempo do sistema
tempo_sist<-vector(mode="integer",length=m)

for (k in 1:nrow(dados_bruto)) {
        tam[dados_bruto[k,4]]<-tam[dados_bruto[k,4]]+1
        if ( k==nrow(dados_bruto) ) { ciclo[dados_bruto[k,4]]<-dados_bruto[k,2]
                                      tempo_sist[dados_bruto[k,4]]<-dados_bruto[k,1]}
        else { if (dados_bruto[k,4]<dados_bruto[k+1,4]) {
                ciclo[dados_bruto[k,4]]<-dados_bruto[k,2]
                tempo_sist[dados_bruto[k,4]]<-dados_bruto[k,1]}
        }
}

#Número Total de falhas considerando todos os sistemas
M<-sum(tam)-sum(ciclo)

MCF_aux<-matrix(0,M,m+1)

MCF<-matrix(0,M,4)

info_ciclo<-matrix(0,sum(ciclo),3)
```

```
k<-1
linha2<-1
auxMP<-0

# Retirando os tempos de MP dos ciclos ou do sistema e ficando somente as falhas em MCF_aux
# info_ciclo tem a informação dos tempos finais (coluna 1) de cada ciclo (coluna 2) dos sistemas (coluna 3)
for (linha in 1:sum(tam)) {
        if ( linha==nrow(dados_bruto) ) { info_ciclo[linha2,1]<-dados_bruto[linha,1]
                                          info_ciclo[linha2,2]<-dados_bruto[linha,2]
                                          info_ciclo[linha2,3]<-dados_bruto[linha,4]
        }
        else {    if (dados_bruto[linha,2]<dados_bruto[linha+1,2]) {info_ciclo[linha2,1]<-dados_bruto[linha,1]
                                                                     info_ciclo[linha2,2]<-dados_bruto[linha,2]
                                                                     info_ciclo[linha2,3]<-dados_bruto[linha,4]
                                                                     linha2<-linha2+1
                                                                     auxMP<-1
                  }


                  if (dados_bruto[linha,4]<dados_bruto[linha+1,4]) {info_ciclo[linha2,1]<-dados_bruto[linha,1]
                                                                     info_ciclo[linha2,2]<-dados_bruto[linha,2]
                                                                     info_ciclo[linha2,3]<-dados_bruto[linha,4]
                                                                     linha2<-linha2+1
                                                                     auxMP<-1
                  }

                  if (auxMP==0) {MCF_aux[k,1]<-dados_bruto[linha,1]
                                 k<-k+1
                  }
                  auxMP<-0
            }
}

# Cada coluna da matriz MCF_aux (a partir da 2a) representa um sistema
# Determina em que ciclo esta cada tempo de falha da matriz MCF_aux para os diferentes sistemas
for (i in 1:M) {

        linha<-1

        for (sistema in 1:m) {

                  while (info_ciclo[linha,3]==sistema){
                            if(MCF_aux[i,1]>=info_ciclo[linha,1]) {
                                 MCF_aux[i,sistema+1]<-info_ciclo[linha,2]+1
                                 linha<-linha+1
                            }
                            else { MCF_aux[i,sistema+1]<-info_ciclo[linha,2]
                                 while (info_ciclo[linha,3]==sistema){ linha<-linha+1
                                                                       if(linha==sum(ciclo)+1){
                                                                                  sistema<-sistema+1
                                                                                  linha<-1}
                                 }
                            }
                  }
        }
}

##########################################################################################
# Definição das funções mi para os 5 possíveis ciclos
##########################################################################################

mi1<-function(t){((t^beta.hat)*((1/lambda.hat)^beta.hat))}

mi2<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                  (((t-((1-exp(-S[1]*theta.hat))*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-(1-exp(-
S[1]*theta.hat)))*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))}
```

```
mi3<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                        (((T[2]-((1-exp(-S[1]*theta.hat))*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-(1-exp(-
S[1]*theta.hat)))*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))+
                        (((t-((1-exp(-S[2]*theta.hat))*T[2]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-(1-exp(-
S[2]*theta.hat)))*T[2])^beta.hat)*((1/lambda.hat)^beta.hat))}

mi4<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                        (((T[2]-((1-exp(-S[1]*theta.hat))*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-(1-exp(-
S[1]*theta.hat)))*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))+
                        (((T[3]-((1-exp(-S[2]*theta.hat))*T[2]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-(1-exp(-
S[2]*theta.hat)))*T[2])^beta.hat)*((1/lambda.hat)^beta.hat))+
                        (((t-((1-exp(-S[3]*theta.hat))*T[3]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-(1-exp(-
S[3]*theta.hat)))*T[3])^beta.hat)*((1/lambda.hat)^beta.hat))}

mi5<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                        (((T[2]-((1-exp(-S[1]*theta.hat))*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-(1-exp(-
S[1]*theta.hat)))*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))+
                        (((T[3]-((1-exp(-S[2]*theta.hat))*T[2]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-(1-exp(-
S[2]*theta.hat)))*T[2])^beta.hat)*((1/lambda.hat)^beta.hat))+
                        (((T[4]-((1-exp(-S[3]*theta.hat))*T[3]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-(1-exp(-
S[3]*theta.hat)))*T[3])^beta.hat)*((1/lambda.hat)^beta.hat))+
                        (((t-((1-exp(-S[4]*theta.hat))*T[4]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-(1-exp(-
S[4]*theta.hat)))*T[4])^beta.hat)*((1/lambda.hat)^beta.hat))}

##########################################################################################
# Cálculo da estimativa do MCF do processo
for (i in 1:M) {

        MCF[i,1]<-MCF_aux[i,1] #Recebe o tempo de falha
        mit<-0

        linha<-1

        for (sistema in 1:m) {

                dados<-matrix(0,tam[sistema],3)

                # Entrada de dados do sistema
                for (k in 1:tam[sistema]) {
                        dados[k,1]<-dados_bruto[linha,1]
                        dados[k,2]<-dados_bruto[linha,2]
                        dados[k,3]<-dados_bruto[linha,3]
                        linha<-linha+1}

                total<-nrow(dados)
                ciclos<-dados[total,2]

                # Cria vetor com os tempos de falha e severidades da MP
                falhas<-matrix(0,(total-ciclos),2)

                # Cria vetor com os tempos de MP e Severidades
                T<-vector(mode="integer",length=ciclos)
                S<-vector(mode="integer",length=ciclos)

                j<-1
                k<-1

                for (q in 1:(total-1)){

                        if (dados[q,2]<dados[q+1,2]) {T[k]<-dados[q,1]
                                                        S[k]<-dados[q,3]
                                                        k<-k+1 }
                        else {  falhas[j,1]<-dados[q,1]
                                falhas[j,2]<-dados[q,2]
                                j<-j+1}
```

```
                    if (q+1==total)        {T[k]<-dados[q+1,1]
                                            S[k]<-dados[q+1,3]}
            }
            # Verficar em que ciclo esta cada sistema, para utilizar a mi correspondente
            if (MCF_aux[i,sistema+1]==1) {mit<-mit+mi1(MCF[i,1])}
            if (MCF_aux[i,sistema+1]==2) {mit<-mit+mi2(MCF[i,1])}
            if (MCF_aux[i,sistema+1]==3) {mit<-mit+mi3(MCF[i,1])}
            if (MCF_aux[i,sistema+1]==4) {mit<-mit+mi4(MCF[i,1])}
            if (MCF_aux[i,sistema+1]==5) {mit<-mit+mi5(MCF[i,1])}
        }
        MCF[i,2]<-mit/m #Estimativa do MCF do processo
}
ord<-order(MCF[,1])
MCF<-MCF[ord,]

# Cálculo da MCF empírica do processo
for (i in 1:M) {
        ri<-0
        for (sistema in 1:m) {
            if(MCF[i,1]<=tempo_sist[sistema]){ri<-ri+1}

        }
        if(i==1) {MCF[i,3]<-(1/ri)}
        else {MCF[i,3]<-(1/ri)+MCF[i-1,3]}
        MCF[i,4]<-ri
}

# Gráfico MCF estimada X MCF empírica
MCF_estimado<-MCF[,2]
MCF_empirico<-MCF[,3]
plot(MCF_estimado,MCF_empirico)
abline(0,1)

##################################################################################
# RELATÓRIO FINAL
##################################################################################
##################################################################################
# Resumo Verificações
##################################################################################
# Deve ser zero para indicar que o algoritmo convergiu
convergencia

if(detver1>=0){"Verificar"}
if(detver2<=0){"Verificar"}
if(detver3>=0){"Verificar"}

if(av$values[1]>=0){"Verificar"}
if(av$values[2]>=0){"Verificar"}
if(av$values[3]>=0){"Verificar"}

##################################################################################
# Estimativas
##################################################################################
lambda.hat
IC_lambda_Inf
IC_lambda_Sup

beta.hat
IC_beta_Inf
IC_beta_Sup

theta.hat
IC_theta_Inf
IC_theta_Sup
```

```
##########################################################################################
# Teste de Razão de Verossimilhanças
##########################################################################################
# P-valor (H0: theta=0)
1-pchisq(TRV, 1)

# Estatística TRV
TRV

# Valor Crítico (VC)
qchisq(0.95,1)

# Se TRV < VC, então aceita H0

##########################################################################################
# Teste Cramer-von Mises
##########################################################################################
#Estatística do Teste CVM
CVM2

# H0: Modelo segue NHPP com u(t)
# Se CVM2 < Valor Crítico, então aceita H0

##########################################################################################
# Otimização Tempo de MP
##########################################################################################
# Custo Manutenção Corretiva
CMC<-1

# custo Manutenção Preventiva
CMP<-5

# Tempo da realização da MP
MP<-T[4]

# Severidade da MP realizada
SMP<-S[4]

# mi=Número de falhas esperadas ultima MP até t
mi<-function(t){((((t-((1-exp(-SMP*theta.hat))*MP))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-(1-exp(-
SMP*theta.hat)))*MP)^beta.hat)*((1/lambda.hat)^beta.hat))}

# Função Custo Médio
FCM<-function(t) { ((CMC*mi(t+MP))+(CMP))/t }

# Otimização do Tempo para realização de MP
TO<-optimize(FCM, c(0,500))

TO$minimum+MP

TO$objective

##########################################################################################
# Função Confiabilidade
##########################################################################################
R<-function(t){exp(-mi(MP+t))}
##########################################################################################
```

## A.4 - Programa referente ao modelo de Redução de Idade com relação Linear (modelo RIL)

```
##################################################################################
# Formato do arquivo de entrada dados.txt
# 1a coluna - Tempos de falha acumulados (o ultimo valor do ciclo deve ser a MP)
# 2a Coluna - Número do ciclo de MP ordenado
# 3a coluna - BSC da MP realizada que influênciará o próximo ciclo de MP
# 4a coluna - Número do sistema ordenado
##################################################################################
library(nlme)
##################################################################################
# Função da LogVerossimilhança para o Modelos de Confiabilidade de Sistemas
# Reparáveis considerando MP Imperfeita
##################################################################################
logvero <- function(x){
	# Entrada dos Parâmetros
	lambda<-x[1]
	beta<-x[2]
	theta<-x[3]

	lv<-0

	# Entrada de Dados
	dados_bruto<-read.table(file="dados.txt", sep = " ")

	# Número de Sistemas analisados
	m<-max(dados_bruto[,4])

	tam<-vector(mode="integer",length=m)

	# Define a quantidade de dados para cada sistema
	for (k in 1:nrow(dados_bruto)) {
		tam[dados_bruto[k,4]]<-tam[dados_bruto[k,4]]+1
	}

	linha<-1

	for (sistema in 1:m) {
		dados<-matrix(0,tam[sistema],3)

		# Entrada de dados do sistema k
		for (k in 1:tam[sistema]) {
			dados[k,1]<-dados_bruto[linha,1]
			dados[k,2]<-dados_bruto[linha,2]
			dados[k,3]<-dados_bruto[linha,3]
			linha<-linha+1		}

		total<-nrow(dados)

		ciclos<-dados[total,2]

		n<-vector(mode="integer",length=ciclos)
		T<-vector(mode="integer",length=ciclos)
		S<-vector(mode="integer",length=ciclos)

		# Cria vetor com os tempos e severidade da MP em cada ciclo
		# e o número de falhas em cada cilco
		for (i in 1:(total-1)){

			if (dados[i,2]<dados[i+1,2]) {T[dados[i,2]]<-dados[i,1]
								S[dados[i,2]]<-dados[i,3]}
```

```
                else {n[dados[i,2]]<-n[dados[i,2]]+1}

        }

        T[dados[i+1,2]]<-dados[i+1,1]
        S[dados[i+1,2]]<-dados[i+1,3]

        # Número total de falhas
        N<-sum(n)

        # Cria vetor com os tempos de falha e severidades da MP
        t<-matrix(0,ciclos,max(n))

        j<-1
        for (i in 1:(total-1)){

                if (dados[i,2]<dados[i+1,2]) { j<-1 }

                else {t[dados[i,2],j]<-dados[i,1]
                    j<-j+1}
        }

        ######################################################################
        # Cálculo da LogVerossimilhança
        ######################################################################
        lv<-lv+(N*log(beta))+(N*log(((1/lambda)^beta)))

        Va<-0 #Idade Virtual inicial
        j<-1
        aux<-0
        for (i in 1:n[j]){
                if(n[j]>0) {aux<-aux+log(t[j,i])}
        }

        lv<-lv+((beta-1)*aux)
        lv<-lv-(((1/lambda)^beta)*(T[j]^beta))

        Va<-T[j]

        for (j in 2:ciclos){

                aux<-0
                for (i in 1:n[j]){
                        if(n[j]>0) {aux<-aux+log(t[j,i]-((S[j-1]*theta)*Va))}
                }

                lv<-lv+((beta-1)*aux)
                lv<-lv-(((1/lambda)^beta)*(((T[j]-((S[j-1]*theta)*Va))^beta)-(((1-(S[j-1]*theta))*Va)^beta)))

                Va<-T[j]

        }
        ######################################################################
    }

    restricao<-0
    for (j in 1:ciclos){
        if( (S[j]*theta) < 0 ) {restricao<-1}
        if( (S[j]*theta) > 1 ) {restricao<-1}

    }

    if(restricao==1) {return(-Inf)} else{return(lv)}


}
```

```
##########################################################################################
# O método da função optim de otimização encontra o mínimo, portanto para determinar
# o máximo da função LogVerossimilhança foi utilizado seu valor negativo
##########################################################################################
logvero_opt <- function(x){

        lambda<-x[1]
        beta<-x[2]
        theta<-x[3]
        return(-logvero(c(lambda,beta,theta)))
}

##########################################################################################
# Otimização da LogVerossimilhança
##########################################################################################
mle<-optim(c(100, 2, 1), logvero_opt, method = "Nelder-Mead")

##########################################################################################
# PARÂMETROS ESTIMADOS
lambda.hat<-mle$par[1]
beta.hat<-mle$par[2]
theta.hat<-mle$par[3]

param<-c(mle$par[1], mle$par[2], mle$par[3])

# Deve ser zero para indicar que o algoritmo convergiu
convergencia<-mle$convergence

mat<-fdHess(mle$par, logvero)
hessiano<-mat$Hessian

# Para ser ponto de máximo devem ser alternados os sinais

# VERIFICAÇÃO 1 (Negativo)
detver1<-hessiano[1,1]

#VERIFICAÇÃO 2 (Positivo)
ver2<-matrix(0,2,2)
ver2[1,1]<-hessiano[1,1]
ver2[1,2]<-hessiano[1,2]
ver2[2,1]<-hessiano[2,1]
ver2[2,2]<-hessiano[2,2]

detver2<-det(ver2)

#VERIFICAÇÃO 3 (Negativo)
ver3<-matrix(0,3,3)
ver3[1,1]<-hessiano[1,1]
ver3[1,2]<-hessiano[1,2]
ver3[1,3]<-hessiano[1,3]
ver3[2,1]<-hessiano[2,1]
ver3[2,2]<-hessiano[2,2]
ver3[2,3]<-hessiano[2,3]
ver3[3,1]<-hessiano[3,1]
ver3[3,2]<-hessiano[3,2]
ver3[3,3]<-hessiano[3,3]

detver3<-det(ver3)

# Outra verificação: se os Autovalores forem todos negativos, é ponto de máximo.
# VERIFICAÇÃO AUTOVALORES
av<-eigen(hessiano)
```

```
##########################################################################################
# Resumo Verificações
##########################################################################################
if(detver1<0){"OK"}else{"Verificar"}
if(detver2>0){"OK"}else{"Verificar"}
if(detver3<0){"OK"}else{"Verificar"}

if(av$values[1]<0){"OK"}else{"Verificar"}
if(av$values[2]<0){"OK"}else{"Verificar"}
if(av$values[3]<0){"OK"}else{"Verificar"}

##########################################################################################
# Desvios Padrãoes das estimativas
##########################################################################################
dp.lambda.hat<-sqrt(solve(-hessiano)[1,1])
dp.beta.hat<-sqrt(solve(-hessiano)[2,2])
dp.theta.hat<-sqrt(solve(-hessiano)[3,3])

# Nível de Confiança
NC<-0.95

IC_lambda_Inf<-lambda.hat/(exp(qnorm((1+NC)/2)*dp.lambda.hat/lambda.hat))
IC_lambda_Sup<-lambda.hat*(exp(qnorm((1+NC)/2)*dp.lambda.hat/lambda.hat))

IC_beta_Inf<-beta.hat/(exp(qnorm((1+NC)/2)*dp.beta.hat/beta.hat))
IC_beta_Sup<-beta.hat*(exp(qnorm((1+NC)/2)*dp.beta.hat/beta.hat))

IC_theta_Inf<-theta.hat/(exp(qnorm((1+NC)/2)*dp.theta.hat/theta.hat))
IC_theta_Sup<-theta.hat*(exp(qnorm((1+NC)/2)*dp.theta.hat/theta.hat))

##########################################################################################
# Verossimilhança da Manutenção Sem Efeito (MSE)
##########################################################################################
logvero_mse <- function(x){

        lambda<-x[1]
        beta<-x[2]
        theta<-0
        return(-logvero(c(lambda,beta,theta)))
}

mle_mse<-optim(c(100, 1.5), logvero_mse, method = "Nelder-Mead")

param_mse<-c(mle_mse$par[1], mle_mse$par[2])

##########################################################################################
# Teste de Razão de Verossimilhanças
##########################################################################################
TRV<-(-2*(logvero(c(param_mse,0))-(logvero(param))))

# P-valor (H0: theta=0)
1-pchisq(TRV, 1)

# Estatística TRV
TRV

# Valor Crítico (VC)
qchisq(0.95,1)

# Se TRV < VC, então aceita H0 (não há efeito entre as MP)
```

```
##########################################################################################
# Teste Cramer-von Mises
##########################################################################################
# Considera o último tempo de cada ciclo como a MP
# É considerado nos cálculos se o último tempo do sistema for falha ou MP
##########################################################################################
# Entrada de Dados
dados_bruto<-read.table(file="dados.txt", sep = " ")

# Número de sistemas analisados
m<-max(dados_bruto[,4])

# tam: quantidade de dados para cada sistema
# ciclo: quantidade de ciclos para cada sistema
tam<-vector(mode="integer",length=m)
ciclo<-vector(mode="integer",length=m)

for (k in 1:nrow(dados_bruto)) {
	tam[dados_bruto[k,4]]<-tam[dados_bruto[k,4]]+1
	if ( k==nrow(dados_bruto) ) { ciclo[dados_bruto[k,4]]<-dados_bruto[k,2] }
	else { if (dados_bruto[k,4]<dados_bruto[k+1,4]) { ciclo[dados_bruto[k,4]]<-dados_bruto[k,2] }}
}

#Número Total de falhas considerando todos os sistemas
M<-sum(tam)-sum(ciclo)

##########################################################################################

CVM_aux<-matrix(0,M,2)

##########################################################################################
# Definição das funções mi para os 5 possíveis ciclos
##########################################################################################
mi1<-function(t){((t^beta.hat)*((1/lambda.hat)^beta.hat))}

mi2<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
		(((t-((S[1]*theta.hat)*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[1]*theta.hat))*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))}

mi3<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
		(((T[2]-((S[1]*theta.hat)*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[1]*theta.hat))*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))+
		(((t-((S[2]*theta.hat)*T[2]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[2]*theta.hat))*T[2])^beta.hat)*((1/lambda.hat)^beta.hat))}

mi4<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
		(((T[2]-((S[1]*theta.hat)*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[1]*theta.hat))*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))+
		(((T[3]-((S[2]*theta.hat)*T[2]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[2]*theta.hat))*T[2])^beta.hat)*((1/lambda.hat)^beta.hat))+
		(((t-((S[3]*theta.hat)*T[3]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[3]*theta.hat))*T[3])^beta.hat)*((1/lambda.hat)^beta.hat))}

mi5<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
		(((T[2]-((S[1]*theta.hat)*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[1]*theta.hat))*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))+
		(((T[3]-((S[2]*theta.hat)*T[2]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[2]*theta.hat))*T[2])^beta.hat)*((1/lambda.hat)^beta.hat))+
		(((T[4]-((S[3]*theta.hat)*T[3]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[3]*theta.hat))*T[3])^beta.hat)*((1/lambda.hat)^beta.hat))+
		(((t-((S[4]*theta.hat)*T[4]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[4]*theta.hat))*T[4])^beta.hat)*((1/lambda.hat)^beta.hat))}

##########################################################################################
```

```
linha<-1
linha2<-1
for (sistema in 1:m) {
        dados<-matrix(0,tam[sistema],3)

        # Entrada de dados do sistema
        for (k in 1:tam[sistema]) {
                dados[k,1]<-dados_bruto[linha,1]
                dados[k,2]<-dados_bruto[linha,2]
                dados[k,3]<-dados_bruto[linha,3]
                linha<-linha+1}

        total<-nrow(dados)
        ciclos<-dados[total,2]

        # Cria vetor com os tempos de falha e severidades da MP
        falhas<-matrix(0,(total-ciclos),2)

        # Cria vetor com os tempos de MP e Severidades
        T<-vector(mode="integer",length=ciclos)
        S<-vector(mode="integer",length=ciclos)

        j<-1
        k<-1

        for (i in 1:(total-1)){

                if (dados[i,2]<dados[i+1,2]) {T[k]<-dados[i,1]
                                              S[k]<-dados[i,3]
                                              k<-k+1 }
                else {  falhas[j,1]<-dados[i,1]
                        falhas[j,2]<-dados[i,2]
                        j<-j+1}

                if (i+1==total)         {T[k]<-dados[i+1,1]
                                         S[k]<-dados[i+1,3]}
        }
        if (ciclos==1) {mit<-mi1(T[ciclos])}
        if (ciclos==2) {mit<-mi2(T[ciclos])}
        if (ciclos==3) {mit<-mi3(T[ciclos])}
        if (ciclos==4) {mit<-mi4(T[ciclos])}
        if (ciclos==5) {mit<-mi5(T[ciclos])}

        tot_falhas<-nrow(falhas)

        for (i in 1:tot_falhas){
                if (falhas[i,2]==1) {CVM_aux[linha2,1]<-mi1(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}
                if (falhas[i,2]==2) {CVM_aux[linha2,1]<-mi2(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}
                if (falhas[i,2]==3) {CVM_aux[linha2,1]<-mi3(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}
                if (falhas[i,2]==4) {CVM_aux[linha2,1]<-mi4(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}
                if (falhas[i,2]==5) {CVM_aux[linha2,1]<-mi5(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}
        # a coluna 1 da matriz CVM_aux tem as estimativas de mi(t)

        }
```

```
}

##################################################################################
# Formato da matriz do Teste Cramer-von Mises (CVM)
# 1a coluna - Ordem da falha
# 2a Coluna - Ri estimado pela Função mi
# 3a coluna - Ri estimado pela Ordem da falha
# 4a coluna - Diferença ao quadrado 2a e 3a colunas
##################################################################################
CVM<-matrix(0,M,4)

ord<-order(CVM_aux[,2])

for (i in 1:M) {
        CVM[i,1]<-i
        CVM[i,2]<-CVM_aux[ord[i],2]
        CVM[i,3]<-(((2*CVM[i,1])-1)/(2*M))
        CVM[i,4]<-(CVM[i,2]-CVM[i,3])^2
}

#Estatística do Teste CVM
CVM2<-(1/(12*M))+sum(CVM[,4])

# H0: Modelo segue NHPP com u(t)
# Se CVM2 < Valor Crítico, então aceita H0.

##################################################################################
# Gráfico para avaliação da Bondade do Ajuste do Modelo
##################################################################################
# Considera o último tempo de cada ciclo como a MP
# Se o tempo da MP for falha, deve replicar esse valor nos dados_bruto
# Essa duplicação deve ser feita apenas para construção deste gráfico
##################################################################################
# Entrada de Dados
dados_bruto<-read.table(file="dados.txt", sep = " ")

# Número de sistemas analisados
m<-max(dados_bruto[,4])

# tam: quantidade de dados para cada sistema
# ciclo: quantidade de ciclos para cada sistema
tam<-vector(mode="integer",length=m)
ciclo<-vector(mode="integer",length=m)

# tempo_sist: ultimo tempo do sistema
tempo_sist<-vector(mode="integer",length=m)

for (k in 1:nrow(dados_bruto)) {
        tam[dados_bruto[k,4]]<-tam[dados_bruto[k,4]]+1
        if ( k==nrow(dados_bruto) ) { ciclo[dados_bruto[k,4]]<-dados_bruto[k,2]
                                      tempo_sist[dados_bruto[k,4]]<-dados_bruto[k,1]}
        else { if (dados_bruto[k,4]<dados_bruto[k+1,4]) {
                ciclo[dados_bruto[k,4]]<-dados_bruto[k,2]
                tempo_sist[dados_bruto[k,4]]<-dados_bruto[k,1]}
        }
}

#Número Total de falhas considerando todos os sistemas
M<-sum(tam)-sum(ciclo)

MCF_aux<-matrix(0,M,m+1)

MCF<-matrix(0,M,4)

info_ciclo<-matrix(0,sum(ciclo),3)
k<-1
```

```
linha2<-1
auxMP<-0
# Retirando os tempos de MP dos ciclos ou do sistema e ficando somente as falhas em MCF_aux
# info_ciclo tem a informação dos tempos finais (coluna 1) de cada ciclo (coluna 2) dos sistemas (coluna 3)
for (linha in 1:sum(tam)) {
        if ( linha==nrow(dados_bruto) ) { info_ciclo[linha2,1]<-dados_bruto[linha,1]
                                          info_ciclo[linha2,2]<-dados_bruto[linha,2]
                                          info_ciclo[linha2,3]<-dados_bruto[linha,4]
        }
        else {    if (dados_bruto[linha,2]<dados_bruto[linha+1,2]) {info_ciclo[linha2,1]<-dados_bruto[linha,1]
                                                                     info_ciclo[linha2,2]<-dados_bruto[linha,2]
                                                                     info_ciclo[linha2,3]<-dados_bruto[linha,4]
                                                                     linha2<-linha2+1
                                                                     auxMP<-1
                  }
                  if (dados_bruto[linha,4]<dados_bruto[linha+1,4]) {info_ciclo[linha2,1]<-dados_bruto[linha,1]
                                                                     info_ciclo[linha2,2]<-dados_bruto[linha,2]
                                                                     info_ciclo[linha2,3]<-dados_bruto[linha,4]
                                                                     linha2<-linha2+1
                                                                     auxMP<-1
                  }
                  if (auxMP==0) {MCF_aux[k,1]<-dados_bruto[linha,1]
                                 k<-k+1
                  }
                  auxMP<-0
          }
}

# Cada coluna da matriz MCF_aux (a partir da 2a) representa um sistema
# Determina em que ciclo esta cada tempo de falha da matriz MCF_aux para os diferentes sistemas
for (i in 1:M) {

        linha<-1

        for (sistema in 1:m) {

                  while (info_ciclo[linha,3]==sistema){
                          if(MCF_aux[i,1]>=info_ciclo[linha,1]) {
                              MCF_aux[i,sistema+1]<-info_ciclo[linha,2]+1
                              linha<-linha+1
                          }
                          else { MCF_aux[i,sistema+1]<-info_ciclo[linha,2]
                              while (info_ciclo[linha,3]==sistema){ linha<-linha+1
                                                                    if(linha==sum(ciclo)+1){
                                                                                sistema<-sistema+1
                                                                                linha<-1}
                              }
                          }
                  }
        }
}

##########################################################################################
# Definição das funções mi para os 5 possíveis ciclos
##########################################################################################
mi1<-function(t){((t^beta.hat)*((1/lambda.hat)^beta.hat))}

mi2<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                  (((t-((S[1]*theta.hat)*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[1]*theta.hat))*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))}

mi3<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                   (((T[2]-((S[1]*theta.hat)*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[1]*theta.hat))*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))+
                   (((t-((S[2]*theta.hat)*T[2]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[2]*theta.hat))*T[2])^beta.hat)*((1/lambda.hat)^beta.hat))}
```

```
mi4<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                        (((T[2]-((S[1]*theta.hat)*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[1]*theta.hat))*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))+
                        (((T[3]-((S[2]*theta.hat)*T[2]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[2]*theta.hat))*T[2])^beta.hat)*((1/lambda.hat)^beta.hat))+
                        (((t-((S[3]*theta.hat)*T[3]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[3]*theta.hat))*T[3])^beta.hat)*((1/lambda.hat)^beta.hat))}

mi5<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                        (((T[2]-((S[1]*theta.hat)*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[1]*theta.hat))*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))+
                        (((T[3]-((S[2]*theta.hat)*T[2]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[2]*theta.hat))*T[2])^beta.hat)*((1/lambda.hat)^beta.hat))+
                        (((T[4]-((S[3]*theta.hat)*T[3]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[3]*theta.hat))*T[3])^beta.hat)*((1/lambda.hat)^beta.hat))+
                        (((t-((S[4]*theta.hat)*T[4]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[4]*theta.hat))*T[4])^beta.hat)*((1/lambda.hat)^beta.hat))}

##################################################################################
# Cálculo da estimativa do MCF do processo
for (i in 1:M) {

        MCF[i,1]<-MCF_aux[i,1] #Recebe o tempo de falha
        mit<-0

        linha<-1

        for (sistema in 1:m) {

                dados<-matrix(0,tam[sistema],3)

                # Entrada de dados do sistema
                for (k in 1:tam[sistema]) {
                        dados[k,1]<-dados_bruto[linha,1]
                        dados[k,2]<-dados_bruto[linha,2]
                        dados[k,3]<-dados_bruto[linha,3]
                        linha<-linha+1}

                total<-nrow(dados)
                ciclos<-dados[total,2]

                # Cria vetor com os tempos de falha e severidades da MP
                falhas<-matrix(0,(total-ciclos),2)

                # Cria vetor com os tempos de MP e Severidades
                T<-vector(mode="integer",length=ciclos)
                S<-vector(mode="integer",length=ciclos)

                j<-1
                k<-1

                for (q in 1:(total-1)){

                        if (dados[q,2]<dados[q+1,2]) {T[k]<-dados[q,1]
                                                        S[k]<-dados[q,3]
                                                        k<-k+1 }
                        else {  falhas[j,1]<-dados[q,1]
                                falhas[j,2]<-dados[q,2]
                                j<-j+1}

                        if (q+1==total)         {T[k]<-dados[q+1,1]
                                                 S[k]<-dados[q+1,3]}
                }

                # Verficar em que ciclo esta cada sistema, para utilizar a mi correspondente
```

```
                    if (MCF_aux[i,sistema+1]==1) {mit<-mit+mi1(MCF[i,1])}

                    if (MCF_aux[i,sistema+1]==2) {mit<-mit+mi2(MCF[i,1])}
                    if (MCF_aux[i,sistema+1]==3) {mit<-mit+mi3(MCF[i,1])}
                    if (MCF_aux[i,sistema+1]==4) {mit<-mit+mi4(MCF[i,1])}
                    if (MCF_aux[i,sistema+1]==5) {mit<-mit+mi5(MCF[i,1])}
          }

          MCF[i,2]<-mit/m #Estimativa do MCF do processo
}

ord<-order(MCF[,1])
MCF<-MCF[ord,]

# Cálculo da MCF empírica do processo
for (i in 1:M) {

          ri<-0
          for (sistema in 1:m) {
                    if(MCF[i,1]<=tempo_sist[sistema]){ri<-ri+1}
          }

          if(i==1) {MCF[i,3]<-(1/ri)}
          else {MCF[i,3]<-(1/ri)+MCF[i-1,3]}
          MCF[i,4]<-ri
}

# Gráfico MCF estimada X MCF empírica
MCF_estimado<-MCF[,2]
MCF_empirico<-MCF[,3]
plot(MCF_estimado,MCF_empirico)
abline(0,1)

##################################################################################
# RELATÓRIO FINAL
##################################################################################
##################################################################################
# Resumo Verificações
##################################################################################

# Deve ser zero para indicar que o algoritmo convergiu
convergencia

if(detver1>=0){"Verificar"}
if(detver2<=0){"Verificar"}
if(detver3>=0){"Verificar"}

if(av$values[1]>=0){"Verificar"}
if(av$values[2]>=0){"Verificar"}
if(av$values[3]>=0){"Verificar"}

##################################################################################
# Estimativas
##################################################################################

lambda.hat
IC_lambda_Inf
IC_lambda_Sup

beta.hat
IC_beta_Inf
IC_beta_Sup

theta.hat
IC_theta_Inf
IC_theta_Sup
```

```
##########################################################################################
# Teste de Razão de Verossimilhanças
##########################################################################################

# P-valor (H0: theta=0)
1-pchisq(TRV, 1)

# Estatística TRV
TRV

# Valor Crítico (VC)
qchisq(0.95,1)

# Se TRV < VC, então aceita H0

##########################################################################################
# Teste Cramer-von Mises
##########################################################################################

#Estatística do Teste CVM
CVM2

# H0: Modelo segue NHPP com u(t)
# Se CVM2 < Valor Crítico, então aceita H0

##########################################################################################
# Otimização Tempo de MP
##########################################################################################

# Custo Manutenção Corretiva
CMC<-1

# custo Manutenção Preventiva
CMP<-5

# Tempo da realização da MP
MP<-T[4]

# Severidade da MP realizada
SMP<-S[4]

# mi=Número de falhas esperadas ultima MP até t
mi<-function(t){((((t-((SMP*theta.hat)*MP))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(SMP*theta.hat))*MP)^beta.hat)*((1/lambda.hat)^beta.hat))}

# Função Custo Médio
FCM<-function(t) { ((CMC*mi(t+MP))+(CMP))/t }

# Otimização do Tempo para realização de MP
TO<-optimize(FCM, c(0,500))

TO$minimum+MP

TO$objective

##########################################################################################
# Função Confiabilidade
##########################################################################################
R<-function(t){exp(-mi(MP+t))}
##########################################################################################
```

## A.5 - Programa referente ao modelo de Redução de Idade com relação Potência (modelo RIP)

```
##################################################################################
# Formato do arquivo de entrada dados.txt
# 1a coluna - Tempos de falha acumulados (o ultimo valor do ciclo deve ser a MP)
# 2a Coluna - Número do ciclo de MP ordenado
# 3a coluna - BSC da MP realizada que influênciará o próximo ciclo de MP
# 4a coluna - Número do sistema ordenado
##################################################################################
library(nlme)
##################################################################################
# Função da LogVerossimilhança para o Modelos de Confiabilidade de Sistemas
# Reparáveis considerando MP Imperfeita
##################################################################################
logvero <- function(x){
	# Entrada dos Parâmetros
	lambda<-x[1]
	beta<-x[2]
	theta<-x[3]

	lv<-0

	# Entrada de Dados
	dados_bruto<-read.table(file="dados.txt", sep = " ")

	# Número de Sistemas analisados
	m<-max(dados_bruto[,4])

	tam<-vector(mode="integer",length=m)

	# Define a quantidade de dados para cada sistema
	for (k in 1:nrow(dados_bruto)) {
		tam[dados_bruto[k,4]]<-tam[dados_bruto[k,4]]+1
	}

	linha<-1

	for (sistema in 1:m) {
		dados<-matrix(0,tam[sistema],3)

		# Entrada de dados do sistema k
		for (k in 1:tam[sistema]) {
			dados[k,1]<-dados_bruto[linha,1]
			dados[k,2]<-dados_bruto[linha,2]
			dados[k,3]<-dados_bruto[linha,3]
			linha<-linha+1		}

		total<-nrow(dados)

		ciclos<-dados[total,2]

		n<-vector(mode="integer",length=ciclos)
		T<-vector(mode="integer",length=ciclos)
		S<-vector(mode="integer",length=ciclos)

		# Cria vetor com os tempos e severidade da MP em cada ciclo
		# e o número de falhas em cada cilco
		for (i in 1:(total-1)){

			if (dados[i,2]<dados[i+1,2]) {T[dados[i,2]]<-dados[i,1]
								S[dados[i,2]]<-dados[i,3]}

			else {n[dados[i,2]]<-n[dados[i,2]]+1}
```

```
                }
                T[dados[i+1,2]]<-dados[i+1,1]
                S[dados[i+1,2]]<-dados[i+1,3]

                # Número total de falhas
                N<-sum(n)

                # Cria vetor com os tempos de falha e severidades da MP
                t<-matrix(0,ciclos,max(n))

                j<-1
                for (i in 1:(total-1)){

                        if (dados[i,2]<dados[i+1,2]) { j<-1 }

                        else {t[dados[i,2],j]<-dados[i,1]
                            j<-j+1}
                }

                ######################################################################
                # Cálculo da LogVerossimilhança
                ######################################################################
                lv<-lv+(N*log(beta))+(N*log(((1/lambda)^beta)))

                Va<-0 #Idade Virtual inicial

                j<-1
                aux<-0
                for (i in 1:n[j]){
                        if(n[j]>0) {aux<-aux+log(t[j,i])}
                }

                lv<-lv+((beta-1)*aux)
                lv<-lv-(((1/lambda)^beta)*(T[j]^beta))

                Va<-T[j]

                for (j in 2:ciclos){
                        aux<-0
                        for (i in 1:n[j]){

                                if(n[j]>0) {aux<-aux+log(t[j,i]-((S[j-1]^theta)*Va))}
                        }

                        lv<-lv+((beta-1)*aux)
                        lv<-lv-(((1/lambda)^beta)*(((T[j]-((S[j-1]^theta)*Va))^beta)-(((1-(S[j-1]^theta))*Va)^beta)))

                        Va<-T[j]
                }
                ######################################################################
        }

        restricao<-0
        for (j in 1:ciclos){
                if( (S[j]^theta) < 0 ) {restricao<-1}
                if( (S[j]^theta) > 1 ) {restricao<-1}

        }
        if(restricao==1) {return(-Inf)} else{return(lv)}

}
```

```
##########################################################################################
# O método da função optim de otimização encontra o mínimo, portanto para determinar
# o máximo da função LogVerossimilhança foi utilizado seu valor negativo
##########################################################################################
logvero_opt <- function(x){

        lambda<-x[1]
        beta<-x[2]
        theta<-x[3]
        return(-logvero(c(lambda,beta,theta)))
}
##########################################################################################
# Otimização da LogVerossimilhança
##########################################################################################
mle<-optim(c(100, 1.8, 0.8), logvero_opt, method = "Nelder-Mead")
##########################################################################################
# PARÂMETROS ESTIMADOS
lambda.hat<-mle$par[1]
beta.hat<-mle$par[2]
theta.hat<-mle$par[3]

param<-c(mle$par[1], mle$par[2], mle$par[3])

# Deve ser zero para indicar que o algoritmo convergiu
convergencia<-mle$convergence

mat<-fdHess(mle$par, logvero)
hessiano<-mat$Hessian

# Para ser ponto de máximo devem ser alternados os sinais

# VERIFICAÇÃO 1 (Negativo)
detver1<-hessiano[1,1]

#VERIFICAÇÃO 2 (Positivo)
ver2<-matrix(0,2,2)
ver2[1,1]<-hessiano[1,1]
ver2[1,2]<-hessiano[1,2]
ver2[2,1]<-hessiano[2,1]
ver2[2,2]<-hessiano[2,2]

detver2<-det(ver2)

#VERIFICAÇÃO 3 (Negativo)
ver3<-matrix(0,3,3)
ver3[1,1]<-hessiano[1,1]
ver3[1,2]<-hessiano[1,2]
ver3[1,3]<-hessiano[1,3]
ver3[2,1]<-hessiano[2,1]
ver3[2,2]<-hessiano[2,2]
ver3[2,3]<-hessiano[2,3]
ver3[3,1]<-hessiano[3,1]
ver3[3,2]<-hessiano[3,2]
ver3[3,3]<-hessiano[3,3]

detver3<-det(ver3)

# Outra verificação: se os Autovalores forem todos negativos, é ponto de máximo.
# VERIFICAÇÃO AUTOVALORES
av<-eigen(hessiano)
```

```
##########################################################################################
# Resumo Verificações
##########################################################################################
if(detver1<0){"OK"}else{"Verificar"}
if(detver2>0){"OK"}else{"Verificar"}
if(detver3<0){"OK"}else{"Verificar"}

if(av$values[1]<0){"OK"}else{"Verificar"}
if(av$values[2]<0){"OK"}else{"Verificar"}
if(av$values[3]<0){"OK"}else{"Verificar"}

##########################################################################################
# Desvios Padrãoes das estimativas
##########################################################################################
dp.lambda.hat<-sqrt(solve(-hessiano)[1,1])
dp.beta.hat<-sqrt(solve(-hessiano)[2,2])
dp.theta.hat<-sqrt(solve(-hessiano)[3,3])

# Nível de Confiança
NC<-0.95

IC_lambda_Inf<-lambda.hat/(exp(qnorm((1+NC)/2)*dp.lambda.hat/lambda.hat))
IC_lambda_Sup<-lambda.hat*(exp(qnorm((1+NC)/2)*dp.lambda.hat/lambda.hat))

IC_beta_Inf<-beta.hat/(exp(qnorm((1+NC)/2)*dp.beta.hat/beta.hat))
IC_beta_Sup<-beta.hat*(exp(qnorm((1+NC)/2)*dp.beta.hat/beta.hat))

IC_theta_Inf<-theta.hat/(exp(qnorm((1+NC)/2)*dp.theta.hat/theta.hat))
IC_theta_Sup<-theta.hat*(exp(qnorm((1+NC)/2)*dp.theta.hat/theta.hat))

##########################################################################################
# Verossimilhança da Manutenção Sem Efeito (MSE)
##########################################################################################
logvero_mse <- function(x){

        lambda<-x[1]
        beta<-x[2]
        theta<-10000000
        return(-logvero(c(lambda,beta,theta)))
}

mle_mse<-optim(c(100, 1.5), logvero_mse, method = "Nelder-Mead")

param_mse<-c(mle_mse$par[1], mle_mse$par[2])

##########################################################################################
# Teste de Razão de Verossimilhanças
##########################################################################################
TRV<-(-2*(logvero(c(param_mse,10000000))-(logvero(param))))

# P-valor (H0: S[i]^theta=0)
1-pchisq(TRV, 1)

# Estatística TRV
TRV

# Valor Crítico (VC)
qchisq(0.95,1)

# Se TRV < VC, então aceita H0 (não há efeito entre as MP)
```

```
##########################################################################################
# Teste Cramer-von Mises
##########################################################################################
# Considera o último tempo de cada ciclo como a MP
# É considerado nos cálculos se o último tempo do sistema for falha ou MP
##########################################################################################
# Entrada de Dados
dados_bruto<-read.table(file="dados.txt", sep = " ")

# Número de sistemas analisados
m<-max(dados_bruto[,4])

# tam: quantidade de dados para cada sistema
# ciclo: quantidade de ciclos para cada sistema
tam<-vector(mode="integer",length=m)
ciclo<-vector(mode="integer",length=m)

for (k in 1:nrow(dados_bruto)) {
	tam[dados_bruto[k,4]]<-tam[dados_bruto[k,4]]+1
	if ( k==nrow(dados_bruto) ) { ciclo[dados_bruto[k,4]]<-dados_bruto[k,2] }
	else { if (dados_bruto[k,4]<dados_bruto[k+1,4]) { ciclo[dados_bruto[k,4]]<-dados_bruto[k,2] }}
}

#Número Total de falhas considerando todos os sistemas
M<-sum(tam)-sum(ciclo)

##########################################################################################
CVM_aux<-matrix(0,M,2)

##########################################################################################
# Definição das funções mi para os 5 possíveis ciclos
##########################################################################################
mi1<-function(t){((t^beta.hat)*((1/lambda.hat)^beta.hat))}

mi2<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                 (((t-((S[1]^theta.hat)*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[1]^theta.hat))*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))}

mi3<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                 (((T[2]-((S[1]^theta.hat)*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[1]^theta.hat))*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))+
                 (((t-((S[2]^theta.hat)*T[2]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[2]^theta.hat))*T[2])^beta.hat)*((1/lambda.hat)^beta.hat))}

mi4<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                 (((T[2]-((S[1]^theta.hat)*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[1]^theta.hat))*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))+
                 (((T[3]-((S[2]^theta.hat)*T[2]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[2]^theta.hat))*T[2])^beta.hat)*((1/lambda.hat)^beta.hat))+
                 (((t-((S[3]^theta.hat)*T[3]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[3]^theta.hat))*T[3])^beta.hat)*((1/lambda.hat)^beta.hat))}

mi5<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                 (((T[2]-((S[1]^theta.hat)*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[1]^theta.hat))*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))+
                 (((T[3]-((S[2]^theta.hat)*T[2]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[2]^theta.hat))*T[2])^beta.hat)*((1/lambda.hat)^beta.hat))+
                 (((T[4]-((S[3]^theta.hat)*T[3]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[3]^theta.hat))*T[3])^beta.hat)*((1/lambda.hat)^beta.hat))+
                 (((t-((S[4]^theta.hat)*T[4]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[4]^theta.hat))*T[4])^beta.hat)*((1/lambda.hat)^beta.hat))}

##########################################################################################
```

```
linha<-1
linha2<-1

for (sistema in 1:m) {

        dados<-matrix(0,tam[sistema],3)

        # Entrada de dados do sistema
        for (k in 1:tam[sistema]) {
                dados[k,1]<-dados_bruto[linha,1]
                dados[k,2]<-dados_bruto[linha,2]
                dados[k,3]<-dados_bruto[linha,3]
                linha<-linha+1}

        total<-nrow(dados)
        ciclos<-dados[total,2]

        # Cria vetor com os tempos de falha e severidades da MP
        falhas<-matrix(0,(total-ciclos),2)

        # Cria vetor com os tempos de MP e Severidades
        T<-vector(mode="integer",length=ciclos)
        S<-vector(mode="integer",length=ciclos)

        j<-1
        k<-1

        for (i in 1:(total-1)){
                if (dados[i,2]<dados[i+1,2]) {T[k]<-dados[i,1]
                                                S[k]<-dados[i,3]
                                                k<-k+1 }
                else {  falhas[j,1]<-dados[i,1]
                        falhas[j,2]<-dados[i,2]
                        j<-j+1}
                if (i+1==total)         {T[k]<-dados[i+1,1]
                                         S[k]<-dados[i+1,3]}
        }

        if (ciclos==1) {mit<-mi1(T[ciclos])}
        if (ciclos==2) {mit<-mi2(T[ciclos])}
        if (ciclos==3) {mit<-mi3(T[ciclos])}
        if (ciclos==4) {mit<-mi4(T[ciclos])}
        if (ciclos==5) {mit<-mi5(T[ciclos])}

        tot_falhas<-nrow(falhas)

        for (i in 1:tot_falhas){
                if (falhas[i,2]==1) {CVM_aux[linha2,1]<-mi1(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}
                if (falhas[i,2]==2) {CVM_aux[linha2,1]<-mi2(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}
                if (falhas[i,2]==3) {CVM_aux[linha2,1]<-mi3(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}
                if (falhas[i,2]==4) {CVM_aux[linha2,1]<-mi4(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}
                if (falhas[i,2]==5) {CVM_aux[linha2,1]<-mi5(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}
        # a coluna 1 da matriz CVM_aux tem as estimativas de mi(t)
```

```
        }
}
##########################################################################################
# Formato da matriz do Teste Cramer-von Mises (CVM)
# 1a coluna - Ordem da falha
# 2a Coluna - Ri estimado pela Função mi
# 3a coluna - Ri estimado pela Ordem da falha
# 4a coluna - Diferença ao quadrado 2a e 3a colunas
##########################################################################################
CVM<-matrix(0,M,4)

ord<-order(CVM_aux[,2])

for (i in 1:M) {
        CVM[i,1]<-i
        CVM[i,2]<-CVM_aux[ord[i],2]
        CVM[i,3]<-(((2*CVM[i,1])-1)/(2*M))
        CVM[i,4]<-(CVM[i,2]-CVM[i,3])^2
}

#Estatística do Teste CVM
CVM2<-(1/(12*M))+sum(CVM[,4])

# H0: Modelo segue NHPP com u(t)
# Se CVM2 < Valor Crítico, então aceita H0.

##########################################################################################
# Gráfico para avaliação da Bondade do Ajuste do Modelo
##########################################################################################
# Considera o último tempo de cada ciclo como a MP
# Se o tempo da MP for falha, deve replicar esse valor nos dados_bruto
# Essa duplicação deve ser feita apenas para construção deste gráfico
##########################################################################################
# Entrada de Dados
dados_bruto<-read.table(file="dados.txt", sep = " ")

# Número de sistemas analisados
m<-max(dados_bruto[,4])

# tam: quantidade de dados para cada sistema
# ciclo: quantidade de ciclos para cada sistema
tam<-vector(mode="integer",length=m)
ciclo<-vector(mode="integer",length=m)

# tempo_sist: ultimo tempo do sistema
tempo_sist<-vector(mode="integer",length=m)

for (k in 1:nrow(dados_bruto)) {
        tam[dados_bruto[k,4]]<-tam[dados_bruto[k,4]]+1
        if ( k==nrow(dados_bruto) ) { ciclo[dados_bruto[k,4]]<-dados_bruto[k,2]
                                        tempo_sist[dados_bruto[k,4]]<-dados_bruto[k,1]}
        else { if (dados_bruto[k,4]<dados_bruto[k+1,4]) {
                ciclo[dados_bruto[k,4]]<-dados_bruto[k,2]
                tempo_sist[dados_bruto[k,4]]<-dados_bruto[k,1]}
        }
}

#Número Total de falhas considerando todos os sistemas
M<-sum(tam)-sum(ciclo)

MCF_aux<-matrix(0,M,m+1)

MCF<-matrix(0,M,4)

info_ciclo<-matrix(0,sum(ciclo),3)
k<-1
```

```
linha2<-1
auxMP<-0
# Retirando os tempos de MP dos ciclos ou do sistema e ficando somente as falhas em MCF_aux
# info_ciclo tem a informação dos tempos finais (coluna 1) de cada ciclo (coluna 2) dos sistemas (coluna 3)
for (linha in 1:sum(tam)) {
        if ( linha==nrow(dados_bruto) ) { info_ciclo[linha2,1]<-dados_bruto[linha,1]
                                          info_ciclo[linha2,2]<-dados_bruto[linha,2]
                                          info_ciclo[linha2,3]<-dados_bruto[linha,4]
        }
        else {    if (dados_bruto[linha,2]<dados_bruto[linha+1,2]) {info_ciclo[linha2,1]<-dados_bruto[linha,1]
                                                                     info_ciclo[linha2,2]<-dados_bruto[linha,2]
                                                                     info_ciclo[linha2,3]<-dados_bruto[linha,4]
                                                                     linha2<-linha2+1
                                                                     auxMP<-1
                  }
                  if (dados_bruto[linha,4]<dados_bruto[linha+1,4]) {info_ciclo[linha2,1]<-dados_bruto[linha,1]
                                                                     info_ciclo[linha2,2]<-dados_bruto[linha,2]
                                                                     info_ciclo[linha2,3]<-dados_bruto[linha,4]
                                                                     linha2<-linha2+1
                                                                     auxMP<-1
                  }
                  if (auxMP==0) {MCF_aux[k,1]<-dados_bruto[linha,1]
                                 k<-k+1
                  }
                  auxMP<-0
          }
}

# Cada coluna da matriz MCF_aux (a partir da 2a) representa um sistema
# Determina em que ciclo esta cada tempo de falha da matriz MCF_aux para os diferentes sistemas
for (i in 1:M) {

        linha<-1
        for (sistema in 1:m) {
                  while (info_ciclo[linha,3]==sistema){
                            if(MCF_aux[i,1]>=info_ciclo[linha,1]) {
                                 MCF_aux[i,sistema+1]<-info_ciclo[linha,2]+1
                                 linha<-linha+1
                            }
                            else { MCF_aux[i,sistema+1]<-info_ciclo[linha,2]
                                 while (info_ciclo[linha,3]==sistema){ linha<-linha+1
                                                                       if(linha==sum(ciclo)+1){
                                                                                  sistema<-sistema+1
                                                                                  linha<-1}
                                 }
                            }
                  }
        }
}


##########################################################################################
# Definição das funções mi para os 5 possíveis ciclos
##########################################################################################
mi1<-function(t){((t^beta.hat)*((1/lambda.hat)^beta.hat))}

mi2<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                  (((t-((S[1]^theta.hat)*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[1]^theta.hat))*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))}

mi3<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                   (((T[2]-((S[1]^theta.hat)*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[1]^theta.hat))*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))+
                   (((t-((S[2]^theta.hat)*T[2]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[2]^theta.hat))*T[2])^beta.hat)*((1/lambda.hat)^beta.hat))}
```

```
mi4<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                    (((T[2]-((S[1]^theta.hat)*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[1]^theta.hat))*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))+
                    (((T[3]-((S[2]^theta.hat)*T[2]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[2]^theta.hat))*T[2])^beta.hat)*((1/lambda.hat)^beta.hat))+
                    (((t-((S[3]^theta.hat)*T[3]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[3]^theta.hat))*T[3])^beta.hat)*((1/lambda.hat)^beta.hat))}

mi5<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                    (((T[2]-((S[1]^theta.hat)*T[1]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[1]^theta.hat))*T[1])^beta.hat)*((1/lambda.hat)^beta.hat))+
                    (((T[3]-((S[2]^theta.hat)*T[2]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[2]^theta.hat))*T[2])^beta.hat)*((1/lambda.hat)^beta.hat))+
                    (((T[4]-((S[3]^theta.hat)*T[3]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[3]^theta.hat))*T[3])^beta.hat)*((1/lambda.hat)^beta.hat))+
                    (((t-((S[4]^theta.hat)*T[4]))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(S[4]^theta.hat))*T[4])^beta.hat)*((1/lambda.hat)^beta.hat))}
################################################################################
# Cálculo da estimativa do MCF do processo
for (i in 1:M) {

        MCF[i,1]<-MCF_aux[i,1] #Recebe o tempo de falha
        mit<-0
        linha<-1

        for (sistema in 1:m) {
                dados<-matrix(0,tam[sistema],3)

                # Entrada de dados do sistema
                for (k in 1:tam[sistema]) {
                        dados[k,1]<-dados_bruto[linha,1]
                        dados[k,2]<-dados_bruto[linha,2]
                        dados[k,3]<-dados_bruto[linha,3]
                        linha<-linha+1}
                total<-nrow(dados)
                ciclos<-dados[total,2]

                # Cria vetor com os tempos de falha e severidades da MP
                falhas<-matrix(0,(total-ciclos),2)

                # Cria vetor com os tempos de MP e Severidades
                T<-vector(mode="integer",length=ciclos)
                S<-vector(mode="integer",length=ciclos)

                j<-1
                k<-1
                for (q in 1:(total-1)){
                        if (dados[q,2]<dados[q+1,2]) {T[k]<-dados[q,1]
                                                        S[k]<-dados[q,3]
                                                        k<-k+1 }
                        else {  falhas[j,1]<-dados[q,1]
                                falhas[j,2]<-dados[q,2]
                                j<-j+1}
                        if (q+1==total)         {T[k]<-dados[q+1,1]
                                                 S[k]<-dados[q+1,3]}
                }
                # Verficar em que ciclo esta cada sistema, para utilizar a mi correspondente
                if (MCF_aux[i,sistema+1]==1) {mit<-mit+mi1(MCF[i,1])}
                if (MCF_aux[i,sistema+1]==2) {mit<-mit+mi2(MCF[i,1])}
                if (MCF_aux[i,sistema+1]==3) {mit<-mit+mi3(MCF[i,1])}
                if (MCF_aux[i,sistema+1]==4) {mit<-mit+mi4(MCF[i,1])}
                if (MCF_aux[i,sistema+1]==5) {mit<-mit+mi5(MCF[i,1])}
        }
        MCF[i,2]<-mit/m #Estimativa do MCF do processo
```

```
}
ord<-order(MCF[,1])
MCF<-MCF[ord,]

# Cálculo da MCF empírica do processo
for (i in 1:M) {

        ri<-0
        for (sistema in 1:m) {
                if(MCF[i,1]<=tempo_sist[sistema]){ri<-ri+1}
        }

        if(i==1) {MCF[i,3]<-(1/ri)}
        else {MCF[i,3]<-(1/ri)+MCF[i-1,3]}
        MCF[i,4]<-ri
}

# Gráfico MCF estimada X MCF empírica
MCF_estimado<-MCF[,2]
MCF_empirico<-MCF[,3]
plot(MCF_estimado,MCF_empirico)
abline(0,1)

##########################################################################################
# RELATÓRIO FINAL
##########################################################################################

##########################################################################################
# Resumo Verificações
##########################################################################################

# Deve ser zero para indicar que o algoritmo convergiu
convergencia

if(detver1>=0){"Verificar"}
if(detver2<=0){"Verificar"}
if(detver3>=0){"Verificar"}

if(av$values[1]>=0){"Verificar"}
if(av$values[2]>=0){"Verificar"}
if(av$values[3]>=0){"Verificar"}

##########################################################################################
# Estimativas
##########################################################################################

lambda.hat
IC_lambda_Inf
IC_lambda_Sup

beta.hat
IC_beta_Inf
IC_beta_Sup

theta.hat
IC_theta_Inf
IC_theta_Sup

##########################################################################################
# Teste de Razão de Verossimilhanças
##########################################################################################

# P-valor (H0: S[i]^theta=0)
1-pchisq(TRV, 1)

# Estatística TRV
```

```
TRV

# Valor Crítico (VC)
qchisq(0.95,1)

# Se TRV < VC, então aceita H0

##########################################################################################
# Teste Cramer-von Mises
##########################################################################################

#Estatística do Teste CVM
CVM2

# H0: Modelo segue NHPP com u(t)
# Se CVM2 < Valor Crítico, então aceita H0

##########################################################################################
# Otimização Tempo de MP
##########################################################################################

# Custo Manutenção Corretiva
CMC<-1

# custo Manutenção Preventiva
CMP<-5

# Tempo da realização da MP
MP<-T[4]

# Severidade da MP realizada
SMP<-S[4]

# mi=Número de falhas esperadas ultima MP até t
mi<-function(t){((((t-((SMP^theta.hat)*MP))^beta.hat)*((1/lambda.hat)^beta.hat))-((((1-
(SMP^theta.hat))*MP)^beta.hat)*((1/lambda.hat)^beta.hat))}

# Função Custo Médio
FCM<-function(t) { ((CMC*mi(t+MP))+(CMP))/t }

# Otimização do Tempo para realização de MP
TO<-optimize(FCM, c(0,500))

TO$minimum+MP

TO$objective

##########################################################################################
# Função Confiabilidade
##########################################################################################

R<-function(t){exp(-mi(MP+t))}

##########################################################################################
```

## A.6 - Programa referente ao modelo de Redução de Intensidade Falha (modelo RF)

```
##################################################################################
# Formato do arquivo de entrada dados.txt
# 1a coluna - Tempos de falha acumulados (o ultimo valor do ciclo deve ser a MP)
# 2a Coluna - Número do ciclo de MP ordenado
# 3a coluna - BSC da MP realizada que influênciará o próximo ciclo de MP
# 4a coluna - Número do sistema ordenado
##################################################################################
library(nlme)
##################################################################################
# Função da LogVerossimilhança para o Modelos de Confiabilidade de Sistemas
# Reparáveis considerando MP Imperfeita
##################################################################################
logvero <- function(x){

	# Entrada dos Parâmetros
	lambda<-x[1]
	beta<-x[2]
	b<-x[3]

	lv<-0

	# Entrada de Dados
	dados_bruto<-read.table(file="dados.txt", sep = " ")

	# Número de Sistemas analisados
	m<-max(dados_bruto[,4])

	tam<-vector(mode="integer",length=m)

	# Define a quantidade de dados para cada sistema
	for (k in 1:nrow(dados_bruto)) {
		tam[dados_bruto[k,4]]<-tam[dados_bruto[k,4]]+1
	}
	linha<-1

	for (sistema in 1:m) {

		dados<-matrix(0,tam[sistema],3)

		# Entrada de dados do sistema k
		for (k in 1:tam[sistema]) {
			dados[k,1]<-dados_bruto[linha,1]
			dados[k,2]<-dados_bruto[linha,2]
			dados[k,3]<-dados_bruto[linha,3]
			linha<-linha+1
		}

		total<-nrow(dados)
		ciclos<-dados[total,2]

		n<-vector(mode="integer",length=ciclos)
		T<-vector(mode="integer",length=ciclos)
		S<-vector(mode="integer",length=ciclos)

		# Cria vetor com os tempos e severidade da MP em cada ciclo
		# e o número de falhas em cada cilco
		for (i in 1:(total-1)){

			if (dados[i,2]<dados[i+1,2]) {T[dados[i,2]]<-dados[i,1]
								S[dados[i,2]]<-dados[i,3]}

			else {n[dados[i,2]]<-n[dados[i,2]]+1}
		}
```

```
                T[dados[i+1,2]]<-dados[i+1,1]
                S[dados[i+1,2]]<-dados[i+1,3]

                # Número total de falhas
                N<-sum(n)

                # Cria vetor com os tempos de falha e severidades da MP
                t<-matrix(0,ciclos,max(n))

                j<-1
                for (i in 1:(total-1)){
                        if (dados[i,2]<dados[i+1,2]) { j<-1 }

                        else {t[dados[i,2],j]<-dados[i,1]
                            j<-j+1}
                }
                ######################################################################
                # Cálculo da LogVerossimilhança
                ######################################################################
                lv<-lv+(N*log(beta))+(N*log(((1/lambda)^beta)))

                Va<-0 #Idade inicial

                aux<-0
                j<-1
                for (i in 1:n[j]){
                        if(n[j]>0) {aux<-aux+log(t[j,i])}
                }
                lv<-lv+((beta-1)*aux)
                lv<-lv-(((1/lambda)^beta)*((T[j]^beta)-(Va^beta)))

                Va<-T[j]

                for (j in 2:ciclos){
                        lv<-lv+((n[j])*log(((1-b)^(j-1))))

                        aux<-0
                        for (i in 1:n[j]){
                                if(n[j]>0) {aux<-aux+log(t[j,i])}
                        }
                        lv<-lv+((beta-1)*aux)
                        lv<-lv-((((1-b)^(j-1))*(((1/lambda)^beta)))*((T[j]^beta)-(Va^beta)))

                        Va<-T[j]
                }
                ######################################################################
        }

        restricao<-0
        for (j in 1:ciclos){
                if( ((1-b)^(j-1)) < 0 ) {restricao<-1}
                if( ((1-b)^(j-1)) > 1 ) {restricao<-1}
        }
        if(restricao==1) {return(-Inf)} else{return(lv)}
}
#################################################################################
# O método da função optim de otimização encontra o mínimo, portanto para determinar
# o máximo da função LogVerossimilhança foi utilizado seu valor negativo
#################################################################################

logvero_opt <- function(x){
        lambda<-x[1]
        beta<-x[2]
        b<-x[3]
        return(-logvero(c(lambda,beta,b)))
}
```

```
##########################################################################################
# Otimização da LogVerossimilhança
##########################################################################################
mle<-optim(c(100, 1.8, 0.5), logvero_opt, method = "Nelder-Mead")
##########################################################################################
# PARÂMETROS ESTIMADOS
lambda.hat<-mle$par[1]
beta.hat<-mle$par[2]
b.hat<-mle$par[3]

param<-c(mle$par[1], mle$par[2], mle$par[3])

# Deve ser zero para indicar que o algoritmo convergiu
convergencia<-mle$convergence

mat<-fdHess(mle$par, logvero)
hessiano<-mat$Hessian

# Para ser ponto de máximo devem ser alternados os sinais

# VERIFICAÇÃO 1 (Negativo)
detver1<-hessiano[1,1]

#VERIFICAÇÃO 2 (Positivo)
ver2<-matrix(0,2,2)
ver2[1,1]<-hessiano[1,1]
ver2[1,2]<-hessiano[1,2]
ver2[2,1]<-hessiano[2,1]
ver2[2,2]<-hessiano[2,2]

detver2<-det(ver2)

#VERIFICAÇÃO 3 (Negativo)
ver3<-matrix(0,3,3)
ver3[1,1]<-hessiano[1,1]
ver3[1,2]<-hessiano[1,2]
ver3[1,3]<-hessiano[1,3]
ver3[2,1]<-hessiano[2,1]
ver3[2,2]<-hessiano[2,2]
ver3[2,3]<-hessiano[2,3]
ver3[3,1]<-hessiano[3,1]
ver3[3,2]<-hessiano[3,2]
ver3[3,3]<-hessiano[3,3]

detver3<-det(ver3)

# Outra verificação: se os Autovalores forem todos negativos, é ponto de máximo.
# VERIFICAÇÃO AUTOVALORES
av<-eigen(hessiano)

##########################################################################################
# Resumo Verificações
##########################################################################################
if(detver1<0){"OK"}else{"Verificar"}
if(detver2>0){"OK"}else{"Verificar"}
if(detver3<0){"OK"}else{"Verificar"}

if(av$values[1]<0){"OK"}else{"Verificar"}
if(av$values[2]<0){"OK"}else{"Verificar"}
if(av$values[3]<0){"OK"}else{"Verificar"}

##########################################################################################
# Desvios Padrãoes das estimativas
##########################################################################################
dp.lambda.hat<-sqrt(solve(-hessiano)[1,1])
dp.beta.hat<-sqrt(solve(-hessiano)[2,2])
```

```
dp.b.hat<-sqrt(solve(-hessiano)[3,3])

# Nível de Confiança
NC<-0.95

IC_lambda_Inf<-lambda.hat/(exp(qnorm((1+NC)/2)*dp.lambda.hat/lambda.hat))
IC_lambda_Sup<-lambda.hat*(exp(qnorm((1+NC)/2)*dp.lambda.hat/lambda.hat))

IC_beta_Inf<-beta.hat/(exp(qnorm((1+NC)/2)*dp.beta.hat/beta.hat))
IC_beta_Sup<-beta.hat*(exp(qnorm((1+NC)/2)*dp.beta.hat/beta.hat))

IC_b_Inf<-b.hat/(exp(qnorm((1+NC)/2)*dp.b.hat/b.hat))
IC_b_Sup<-b.hat*(exp(qnorm((1+NC)/2)*dp.b.hat/b.hat))

##################################################################################
# Verossimilhança da Manutenção Sem Efeito (MSE)
##################################################################################
logvero_mse <- function(x){

        lambda<-x[1]
        beta<-x[2]
        b<-0
        return(-logvero(c(lambda,beta,b)))
}

mle_mse<-optim(c(100, 1.5), logvero_mse, method = "Nelder-Mead")
param_mse<-c(mle_mse$par[1], mle_mse$par[2])

##################################################################################
# Teste de Razão de Verossimilhanças
##################################################################################
TRV<-(-2*(logvero(c(param_mse,0))-(logvero(param))))

# P-valor (H0: b=0)
1-pchisq(TRV, 1)

# Estatística TRV
TRV

# Valor Crítico (VC)
qchisq(0.95,1)

# Se TRV < VC, então aceita H0 (não há efeito entre as MP)

##################################################################################
# Teste Cramer-von Mises
##################################################################################
# Considera o último tempo de cada ciclo como a MP
# É considerado nos cálculos se o último tempo do sistema for falha ou MP
##################################################################################
# Entrada de Dados
dados_bruto<-read.table(file="dados.txt", sep = " ")

# Número de sistemas analisados
m<-max(dados_bruto[,4])

# tam: quantidade de dados para cada sistema
# ciclo: quantidade de ciclos para cada sistema
tam<-vector(mode="integer",length=m)
ciclo<-vector(mode="integer",length=m)

for (k in 1:nrow(dados_bruto)) {
        tam[dados_bruto[k,4]]<-tam[dados_bruto[k,4]]+1
        if ( k==nrow(dados_bruto) ) { ciclo[dados_bruto[k,4]]<-dados_bruto[k,2] }
        else { if (dados_bruto[k,4]<dados_bruto[k+1,4]) { ciclo[dados_bruto[k,4]]<-dados_bruto[k,2] }}
}
```

```
#Número Total de falhas considerando todos os sistemas
M<-sum(tam)-sum(ciclo)

##########################################################################################
CVM_aux<-matrix(0,M,2)
##########################################################################################
# Definição das funções mi para os 5 possíveis ciclos
##########################################################################################
mi1<-function(t){((t^beta.hat)*((1/lambda.hat)^beta.hat))}
mi2<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                    ((1-b.hat)*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))))}
mi3<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                    ((1-b.hat)*(((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))-((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                    (((1-b.hat)^2)*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))))}
mi4<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                    ((1-b.hat)*(((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))-((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                    (((1-b.hat)^2)*(((T[3]^beta.hat)*((1/lambda.hat)^beta.hat))-((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                    (((1-b.hat)^3)*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-((T[3]^beta.hat)*((1/lambda.hat)^beta.hat))))}
mi5<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                    ((1-b.hat)*(((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))-((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                    (((1-b.hat)^2)*(((T[3]^beta.hat)*((1/lambda.hat)^beta.hat))-((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                    (((1-b.hat)^3)*(((T[4]^beta.hat)*((1/lambda.hat)^beta.hat))-((T[3]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                    (((1-b.hat)^4)*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-((T[4]^beta.hat)*((1/lambda.hat)^beta.hat))))}
##########################################################################################
linha<-1
linha2<-1

for (sistema in 1:m) {
        dados<-matrix(0,tam[sistema],3)

        # Entrada de dados do sistema
        for (k in 1:tam[sistema]) {
                dados[k,1]<-dados_bruto[linha,1]
                dados[k,2]<-dados_bruto[linha,2]
                dados[k,3]<-dados_bruto[linha,3]
                linha<-linha+1}
        total<-nrow(dados)
        ciclos<-dados[total,2]

        # Cria vetor com os tempos de falha e severidades da MP
        falhas<-matrix(0,(total-ciclos),2)

        # Cria vetor com os tempos de MP e Severidades
        T<-vector(mode="integer",length=ciclos)
        S<-vector(mode="integer",length=ciclos)

        j<-1
        k<-1

        for (i in 1:(total-1)){
                if (dados[i,2]<dados[i+1,2]) {T[k]<-dados[i,1]
                                                S[k]<-dados[i,3]
                                                k<-k+1 }
                else {  falhas[j,1]<-dados[i,1]
                        falhas[j,2]<-dados[i,2]
                        j<-j+1}

                if (i+1==total)         {T[k]<-dados[i+1,1]
                                         S[k]<-dados[i+1,3]}
        }
        if (ciclos==1) {mit<-mi1(T[ciclos])}
        if (ciclos==2) {mit<-mi2(T[ciclos])}
        if (ciclos==3) {mit<-mi3(T[ciclos])}
        if (ciclos==4) {mit<-mi4(T[ciclos])}
        if (ciclos==5) {mit<-mi5(T[ciclos])}
```

```
        tot_falhas<-nrow(falhas)

        for (i in 1:tot_falhas){
                if (falhas[i,2]==1) {CVM_aux[linha2,1]<-mi1(falhas[i,1])
                                CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                linha2<-linha2+1}
                if (falhas[i,2]==2) {CVM_aux[linha2,1]<-mi2(falhas[i,1])
                                CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                linha2<-linha2+1}
                if (falhas[i,2]==3) {CVM_aux[linha2,1]<-mi3(falhas[i,1])
                                CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                linha2<-linha2+1}
                if (falhas[i,2]==4) {CVM_aux[linha2,1]<-mi4(falhas[i,1])
                                CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                linha2<-linha2+1}
                if (falhas[i,2]==5) {CVM_aux[linha2,1]<-mi5(falhas[i,1])
                                CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                linha2<-linha2+1}
        # a coluna 1 da matriz CVM_aux tem as estimativas de mi(t)
        }
}

##########################################################################################
# Formato da matriz do Teste Cramer-von Mises (CVM)
# 1a coluna - Ordem da falha
# 2a Coluna - Ri estimado pela Função mi
# 3a coluna - Ri estimado pela Ordem da falha
# 4a coluna - Diferença ao quadrado 2a e 3a colunas
##########################################################################################
CVM<-matrix(0,M,4)

ord<-order(CVM_aux[,2])
for (i in 1:M) {
        CVM[i,1]<-i
        CVM[i,2]<-CVM_aux[ord[i],2]
        CVM[i,3]<-(((2*CVM[i,1])-1)/(2*M))
        CVM[i,4]<-(CVM[i,2]-CVM[i,3])^2
}

#Estatística do Teste CVM
CVM2<-(1/(12*M))+sum(CVM[,4])

# H0: Modelo segue NHPP com u(t)
# Se CVM2 < Valor Crítico, então aceita H0.
##########################################################################################
# Gráfico para avaliação da Bondade do Ajuste do Modelo
##########################################################################################
# Considera o último tempo de cada ciclo como a MP
# Se o tempo da MP for falha, deve replicar esse valor nos dados_bruto
# Essa duplicação deve ser feita apenas para construção deste gráfico
##########################################################################################
# Entrada de Dados
dados_bruto<-read.table(file="dados.txt", sep = " ")

# Número de sistemas analisados
m<-max(dados_bruto[,4])

# tam: quantidade de dados para cada sistema
# ciclo: quantidade de ciclos para cada sistema
tam<-vector(mode="integer",length=m)
ciclo<-vector(mode="integer",length=m)

# tempo_sist: ultimo tempo do sistema
tempo_sist<-vector(mode="integer",length=m)
```

```
for (k in 1:nrow(dados_bruto)) {
        tam[dados_bruto[k,4]]<-tam[dados_bruto[k,4]]+1
        if ( k==nrow(dados_bruto) ) { ciclo[dados_bruto[k,4]]<-dados_bruto[k,2]
                                        tempo_sist[dados_bruto[k,4]]<-dados_bruto[k,1]}
        else { if (dados_bruto[k,4]<dados_bruto[k+1,4]) {
                ciclo[dados_bruto[k,4]]<-dados_bruto[k,2]
                tempo_sist[dados_bruto[k,4]]<-dados_bruto[k,1]}
        }
}
#Número Total de falhas considerando todos os sistemas
M<-sum(tam)-sum(ciclo)

MCF_aux<-matrix(0,M,m+1)
MCF<-matrix(0,M,4)

info_ciclo<-matrix(0,sum(ciclo),3)
k<-1
linha2<-1
auxMP<-0

# Retirando os tempos de MP dos ciclos ou do sistema e ficando somente as falhas em MCF_aux
# info_ciclo tem a informação dos tempos finais (coluna 1) de cada ciclo (coluna 2) dos sistemas (coluna 3)
for (linha in 1:sum(tam)) {
        if ( linha==nrow(dados_bruto) ) { info_ciclo[linha2,1]<-dados_bruto[linha,1]
                                          info_ciclo[linha2,2]<-dados_bruto[linha,2]
                                          info_ciclo[linha2,3]<-dados_bruto[linha,4]
        }
        else {      if (dados_bruto[linha,2]<dados_bruto[linha+1,2]) {info_ciclo[linha2,1]<-dados_bruto[linha,1]
                                                                       info_ciclo[linha2,2]<-dados_bruto[linha,2]
                                                                       info_ciclo[linha2,3]<-dados_bruto[linha,4]
                                                                       linha2<-linha2+1
                                                                       auxMP<-1
                    }
                    if (dados_bruto[linha,4]<dados_bruto[linha+1,4]) {info_ciclo[linha2,1]<-dados_bruto[linha,1]
                                                                       info_ciclo[linha2,2]<-dados_bruto[linha,2]
                                                                       info_ciclo[linha2,3]<-dados_bruto[linha,4]
                                                                       linha2<-linha2+1
                                                                       auxMP<-1
                    }
                    if (auxMP==0) {MCF_aux[k,1]<-dados_bruto[linha,1]
                                   k<-k+1
                    }
                    auxMP<-0
          }
}
# Cada coluna da matriz MCF_aux (a partir da 2a) representa um sistema
# Determina em que ciclo esta cada tempo de falha da matriz MCF_aux para os diferentes sistemas
for (i in 1:M) {
        linha<-1
        for (sistema in 1:m) {
                while (info_ciclo[linha,3]==sistema){
                        if(MCF_aux[i,1]>=info_ciclo[linha,1]) {
                            MCF_aux[i,sistema+1]<-info_ciclo[linha,2]+1
                            linha<-linha+1
                        }
                        else { MCF_aux[i,sistema+1]<-info_ciclo[linha,2]
                            while (info_ciclo[linha,3]==sistema){ linha<-linha+1
                                                                  if(linha==sum(ciclo)+1){
                                                                                  sistema<-sistema+1
                                                                                  linha<-1}
                            }
                        }
                }
        }
}
```

```
##################################################################################
# Definição das funções mi para os 5 possíveis ciclos
##################################################################################
mi1<-function(t){((t^beta.hat)*((1/lambda.hat)^beta.hat))}
mi2<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                    ((1-b.hat)*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))))}
mi3<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                    ((1-b.hat)*(((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))-((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                    (((1-b.hat)^2)*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))))}
mi4<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                    ((1-b.hat)*(((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))-((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                    (((1-b.hat)^2)*(((T[3]^beta.hat)*((1/lambda.hat)^beta.hat))-((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                    (((1-b.hat)^3)*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-((T[3]^beta.hat)*((1/lambda.hat)^beta.hat))))}
mi5<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                    ((1-b.hat)*(((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))-((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                    (((1-b.hat)^2)*(((T[3]^beta.hat)*((1/lambda.hat)^beta.hat))-((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                    (((1-b.hat)^3)*(((T[4]^beta.hat)*((1/lambda.hat)^beta.hat))-((T[3]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                    (((1-b.hat)^4)*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-((T[4]^beta.hat)*((1/lambda.hat)^beta.hat))))}

##################################################################################
# Cálculo da estimativa do MCF do processo
for (i in 1:M) {

        MCF[i,1]<-MCF_aux[i,1] #Recebe o tempo de falha
        mit<-0

        linha<-1

        for (sistema in 1:m) {
                dados<-matrix(0,tam[sistema],3)

                # Entrada de dados do sistema
                for (k in 1:tam[sistema]) {
                        dados[k,1]<-dados_bruto[linha,1]
                        dados[k,2]<-dados_bruto[linha,2]
                        dados[k,3]<-dados_bruto[linha,3]
                        linha<-linha+1}

                total<-nrow(dados)
                ciclos<-dados[total,2]

                # Cria vetor com os tempos de falha e severidades da MP
                falhas<-matrix(0,(total-ciclos),2)

                # Cria vetor com os tempos de MP e Severidades
                T<-vector(mode="integer",length=ciclos)
                S<-vector(mode="integer",length=ciclos)

                j<-1
                k<-1

                for (q in 1:(total-1)){

                        if (dados[q,2]<dados[q+1,2]) {T[k]<-dados[q,1]
                                                        S[k]<-dados[q,3]
                                                        k<-k+1 }
                        else {  falhas[j,1]<-dados[q,1]
                                falhas[j,2]<-dados[q,2]
                                j<-j+1}

                        if (q+1==total)         {T[k]<-dados[q+1,1]
                                                 S[k]<-dados[q+1,3]}
                }
```

```
                # Verficar em que ciclo esta cada sistema, para utilizar a mi correspondente
                if (MCF_aux[i,sistema+1]==1) {mit<-mit+mi1(MCF[i,1])}
                if (MCF_aux[i,sistema+1]==2) {mit<-mit+mi2(MCF[i,1])}
                if (MCF_aux[i,sistema+1]==3) {mit<-mit+mi3(MCF[i,1])}
                if (MCF_aux[i,sistema+1]==4) {mit<-mit+mi4(MCF[i,1])}
                if (MCF_aux[i,sistema+1]==5) {mit<-mit+mi5(MCF[i,1])}
        }
        MCF[i,2]<-mit/m #Estimativa do MCF do processo
}
ord<-order(MCF[,1])
MCF<-MCF[ord,]

# Cálculo da MCF empírica do processo
for (i in 1:M) {
        ri<-0
        for (sistema in 1:m) {
                if(MCF[i,1]<=tempo_sist[sistema]){ri<-ri+1}
        }

        if(i==1) {MCF[i,3]<-(1/ri)}
        else {MCF[i,3]<-(1/ri)+MCF[i-1,3]}
        MCF[i,4]<-ri
}

# Gráfico MCF estimada X MCF empírica
MCF_estimado<-MCF[,2]
MCF_empirico<-MCF[,3]
plot(MCF_estimado,MCF_empirico)
abline(0,1)

##########################################################################################
# RELATÓRIO FINAL
##########################################################################################

##########################################################################################
# Resumo Verificações
##########################################################################################
# Deve ser zero para indicar que o algoritmo convergiu
convergencia

if(detver1>=0){"Verificar"}
if(detver2<=0){"Verificar"}
if(detver3>=0){"Verificar"}

if(av$values[1]>=0){"Verificar"}
if(av$values[2]>=0){"Verificar"}
if(av$values[3]>=0){"Verificar"}

##########################################################################################
# Estimativas
##########################################################################################
lambda.hat
IC_lambda_Inf
IC_lambda_Sup

beta.hat
IC_beta_Inf
IC_beta_Sup

b.hat
IC_b_Inf
IC_b_Sup
```

```
##########################################################################################
# Teste de Razão de Verossimilhanças
##########################################################################################
# P-valor (H0: b=0)
1-pchisq(TRV, 1)

# Estatística TRV
TRV

# Valor Crítico (VC)
qchisq(0.95,1)

# Se TRV < VC, então aceita H0

##########################################################################################
# Teste Cramer-von Mises
##########################################################################################
#Estatística do Teste CVM
CVM2

# H0: Modelo segue NHPP com u(t)
# Se CVM2 < Valor Crítico, então aceita H0

##########################################################################################
# Otimização Tempo de MP
##########################################################################################
# Custo Manutenção Corretiva
CMC<-1

# custo Manutenção Preventiva
CMP<-5

# Tempo da realização da MP
MP<-T[4]

# Severidade da MP realizada
SMP<-S[4]

# Ciclo da MP
ciclo<-5

# mi=Número de falhas esperadas ultima MP até t
mi<-function(t){((((1-b.hat)^(ciclo-1))*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-((MP^beta.hat)*((1/lambda.hat)^beta.hat))))}

# Função Custo Médio
FCM<-function(t) { ((CMC*mi(t+MP))+(CMP))/t }

# Otimização do Tempo para realização de MP
TO<-optimize(FCM, c(0,500))

TO$minimum+MP

TO$objective

##########################################################################################
# Função Confiabilidade
##########################################################################################
R<-function(t){exp(-mi(MP+t))}
##########################################################################################
```

## A.7 - Programa referente ao modelo de Redução de Intensidade Falha com relação Exponencial (modelo RFE)

```
##################################################################################
# Formato do arquivo de entrada dados.txt
# 1a coluna - Tempos de falha acumulados (o ultimo valor do ciclo deve ser a MP)
# 2a Coluna - Número do ciclo de MP ordenado
# 3a coluna - BSC da MP realizada que influênciará o próximo ciclo de MP
# 4a coluna - Número do sistema ordenado
##################################################################################
library(nlme)
##################################################################################
# Função da LogVerossimilhança para o Modelos de Confiabilidade de Sistemas
# Reparáveis considerando MP Imperfeita
##################################################################################
logvero <- function(x){

	# Entrada dos Parâmetros
	lambda<-x[1]
	beta<-x[2]
	theta<-x[3]

	lv<-0

	# Entrada de Dados
	dados_bruto<-read.table(file="dados.txt", sep = " ")

	# Número de Sistemas analisados
	m<-max(dados_bruto[,4])

	tam<-vector(mode="integer",length=m)

	# Define a quantidade de dados para cada sistema
	for (k in 1:nrow(dados_bruto)) {
		tam[dados_bruto[k,4]]<-tam[dados_bruto[k,4]]+1
	}

	linha<-1

	for (sistema in 1:m) {
		dados<-matrix(0,tam[sistema],3)

		# Entrada de dados do sistema k
		for (k in 1:tam[sistema]) {
			dados[k,1]<-dados_bruto[linha,1]
			dados[k,2]<-dados_bruto[linha,2]
			dados[k,3]<-dados_bruto[linha,3]
			linha<-linha+1		}

		total<-nrow(dados)
		ciclos<-dados[total,2]

		n<-vector(mode="integer",length=ciclos)
		T<-vector(mode="integer",length=ciclos)
		S<-vector(mode="integer",length=ciclos)

		# Cria vetor com os tempos e severidade da MP em cada ciclo
		# e o número de falhas em cada cilco
		for (i in 1:(total-1)){

			if (dados[i,2]<dados[i+1,2]) {T[dados[i,2]]<-dados[i,1]
								S[dados[i,2]]<-dados[i,3]}

			else {n[dados[i,2]]<-n[dados[i,2]]+1}
```

```
}

T[dados[i+1,2]]<-dados[i+1,1]
S[dados[i+1,2]]<-dados[i+1,3]

# Número total de falhas
N<-sum(n)

# Cria vetor com os tempos de falha e severidades da MP
t<-matrix(0,ciclos,max(n))

j<-1
for (i in 1:(total-1)){

        if (dados[i,2]<dados[i+1,2]) { j<-1 }

        else {t[dados[i,2],j]<-dados[i,1]
            j<-j+1}
}
#######################################################################
# Cálculo da LogVerossimilhança
#######################################################################
lv<-lv+(N*log(beta))+(N*log(((1/lambda)^beta)))

Va<-0 #Idade inicial

aux<-0
j<-1
for (i in 1:n[j]){
        if(n[j]>0) {aux<-aux+log(t[j,i])}
}
lv<-lv+((beta-1)*aux)
lv<-lv-(((1/lambda)^beta)*((T[j]^beta)-(Va^beta)))

Va<-T[j]

for (j in 2:ciclos){

        aux<-0
        for (i in 1:n[j]){
                if(n[j]>0) {aux<-aux+log(t[j,i])}
        }
        lv<-lv+((beta-1)*aux)

        if (j==2) {
                lv<-lv+((n[j])*log((exp(-S[j-1]*theta))))
                lv<-lv-(((exp(-S[j-1]*theta))*(((1/lambda)^beta)))*((T[j]^beta)-(Va^beta)))}

        if (j==3) {
                lv<-lv+((n[j])*log((exp(-S[j-2]*theta))*(exp(-S[j-1]*theta))))
                lv<-lv-(((exp(-S[j-2]*theta))*(exp(-S[j-1]*theta))*(((1/lambda)^beta)))*((T[j]^beta)-
(Va^beta)))}

        if (j==4) {
                lv<-lv+((n[j])*log((exp(-S[j-3]*theta))*(exp(-S[j-2]*theta))*(exp(-S[j-1]*theta))))

                lv<-lv-(((exp(-S[j-3]*theta))*(exp(-S[j-2]*theta))*(exp(-S[j-
1]*theta))*(((1/lambda)^beta)))*((T[j]^beta)-(Va^beta)))}

        if (j==5) {
                lv<-lv+((n[j])*log((exp(-S[j-4]*theta))*(exp(-S[j-3]*theta))*(exp(-S[j-2]*theta))*(exp(-S[j-
1]*theta))))
                lv<-lv-(((exp(-S[j-4]*theta))*(exp(-S[j-3]*theta))*(exp(-S[j-2]*theta))*(exp(-S[j-
1]*theta))*(((1/lambda)^beta)))*((T[j]^beta)-(Va^beta)))}
```

```
                        Va<-T[j]
                }
                ##########################################################################
        }
        restricao<-0
        for (j in 1:ciclos){
                if( (exp(-S[j]*theta)) < 0 ) {restricao<-1}
                if( (exp(-S[j]*theta)) > 1 ) {restricao<-1}
        }

        if(restricao==1) {return(-Inf)} else{return(lv)}
}
##########################################################################################
# O método da função optim de otimização encontra o mínimo, portanto para determinar
# o máximo da função LogVerossimilhança foi utilizado seu valor negativo
##########################################################################################
logvero_opt <- function(x){
        lambda<-x[1]
        beta<-x[2]
        theta<-x[3]
        return(-logvero(c(lambda,beta,theta)))
}
##########################################################################################
# Otimização da LogVerossimilhança
##########################################################################################
mle<-optim(c(100, 1.8, 0.8), logvero_opt, method = "Nelder-Mead")
##########################################################################################
# PARÂMETROS ESTIMADOS
lambda.hat<-mle$par[1]
beta.hat<-mle$par[2]
theta.hat<-mle$par[3]

param<-c(mle$par[1], mle$par[2], mle$par[3])

# Deve ser zero para indicar que o algoritmo convergiu
convergencia<-mle$convergence

mat<-fdHess(mle$par, logvero)
hessiano<-mat$Hessian

# Para ser ponto de máximo devem ser alternados os sinais

# VERIFICAÇÃO 1 (Negativo)
detver1<-hessiano[1,1]

#VERIFICAÇÃO 2 (Positivo)
ver2<-matrix(0,2,2)
ver2[1,1]<-hessiano[1,1]
ver2[1,2]<-hessiano[1,2]
ver2[2,1]<-hessiano[2,1]
ver2[2,2]<-hessiano[2,2]

detver2<-det(ver2)

#VERIFICAÇÃO 3 (Negativo)
ver3<-matrix(0,3,3)
ver3[1,1]<-hessiano[1,1]
ver3[1,2]<-hessiano[1,2]
ver3[1,3]<-hessiano[1,3]
ver3[2,1]<-hessiano[2,1]
ver3[2,2]<-hessiano[2,2]
ver3[2,3]<-hessiano[2,3]
ver3[3,1]<-hessiano[3,1]
ver3[3,2]<-hessiano[3,2]
ver3[3,3]<-hessiano[3,3]
```

```
detver3<-det(ver3)

# Outra verificação: se os Autovalores forem todos negativos, é ponto de máximo.
# VERIFICAÇÃO AUTOVALORES
av<-eigen(hessiano)
##################################################################################
# Resumo Verificações
##################################################################################
if(detver1<0){"OK"}else{"Verificar"}
if(detver2>0){"OK"}else{"Verificar"}
if(detver3<0){"OK"}else{"Verificar"}

if(av$values[1]<0){"OK"}else{"Verificar"}
if(av$values[2]<0){"OK"}else{"Verificar"}
if(av$values[3]<0){"OK"}else{"Verificar"}

##################################################################################
# Desvios Padrãoes das estimativas
##################################################################################
dp.lambda.hat<-sqrt(solve(-hessiano)[1,1])
dp.beta.hat<-sqrt(solve(-hessiano)[2,2])
dp.theta.hat<-sqrt(solve(-hessiano)[3,3])

# Nível de Confiança
NC<-0.95

IC_lambda_Inf<-lambda.hat/(exp(qnorm((1+NC)/2)*dp.lambda.hat/lambda.hat))
IC_lambda_Sup<-lambda.hat*(exp(qnorm((1+NC)/2)*dp.lambda.hat/lambda.hat))

IC_beta_Inf<-beta.hat/(exp(qnorm((1+NC)/2)*dp.beta.hat/beta.hat))
IC_beta_Sup<-beta.hat*(exp(qnorm((1+NC)/2)*dp.beta.hat/beta.hat))

IC_theta_Inf<-theta.hat/(exp(qnorm((1+NC)/2)*dp.theta.hat/theta.hat))
IC_theta_Sup<-theta.hat*(exp(qnorm((1+NC)/2)*dp.theta.hat/theta.hat))

##################################################################################
# Verossimilhança da Manutenção Sem Efeito (MSE)
##################################################################################
logvero_mse <- function(x){

        lambda<-x[1]
        beta<-x[2]
        theta<-0
        return(-logvero(c(lambda,beta,theta)))
}

mle_mse<-optim(c(100, 1.5), logvero_mse, method = "Nelder-Mead")

param_mse<-c(mle_mse$par[1], mle_mse$par[2])

##################################################################################
# Teste de Razão de Verossimilhanças
##################################################################################
TRV<-(-2*(logvero(c(param_mse,0))-(logvero(param))))

# P-valor (H0: theta=0)
1-pchisq(TRV, 1)

# Estatística TRV
TRV

# Valor Crítico (VC)
qchisq(0.95,1)

# Se TRV < VC, então aceita H0 (não há efeito entre as MP)
```

```
##########################################################################################
# Teste Cramer-von Mises
##########################################################################################
# Considera o último tempo de cada ciclo como a MP
# É considerado nos cálculos se o último tempo do sistema for falha ou MP
##########################################################################################
# Entrada de Dados
dados_bruto<-read.table(file="dados.txt", sep = " ")

# Número de sistemas analisados
m<-max(dados_bruto[,4])

# tam: quantidade de dados para cada sistema
# ciclo: quantidade de ciclos para cada sistema
tam<-vector(mode="integer",length=m)
ciclo<-vector(mode="integer",length=m)

for (k in 1:nrow(dados_bruto)) {
        tam[dados_bruto[k,4]]<-tam[dados_bruto[k,4]]+1
        if ( k==nrow(dados_bruto) ) { ciclo[dados_bruto[k,4]]<-dados_bruto[k,2] }
        else { if (dados_bruto[k,4]<dados_bruto[k+1,4]) { ciclo[dados_bruto[k,4]]<-dados_bruto[k,2] }}
}

#Número Total de falhas considerando todos os sistemas
M<-sum(tam)-sum(ciclo)

##########################################################################################

CVM_aux<-matrix(0,M,2)

##########################################################################################
# Definição das funções mi para os 5 possíveis ciclos
##########################################################################################
mi1<-function(t){((t^beta.hat)*((1/lambda.hat)^beta.hat))}

mi2<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                ((exp(-S[1]*theta.hat))*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))))}

mi3<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                ((exp(-S[1]*theta.hat))*(((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))-((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                ((exp(-S[2]*theta.hat))*(exp(-S[1]*theta.hat))*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-
((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))))}

mi4<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                ((exp(-S[1]*theta.hat))*(((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))-((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                ((exp(-S[2]*theta.hat))*(exp(-S[1]*theta.hat))*(((T[3]^beta.hat)*((1/lambda.hat)^beta.hat))-
((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                ((exp(-S[3]*theta.hat))*(exp(-S[2]*theta.hat))*(exp(-S[1]*theta.hat))*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-
((T[3]^beta.hat)*((1/lambda.hat)^beta.hat))))}

mi5<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                ((exp(-S[1]*theta.hat))*(((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))-((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                ((exp(-S[2]*theta.hat))*(exp(-S[1]*theta.hat))*(((T[3]^beta.hat)*((1/lambda.hat)^beta.hat))-
((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                ((exp(-S[3]*theta.hat))*(exp(-S[2]*theta.hat))*(exp(-S[1]*theta.hat))*(((T[4]^beta.hat)*((1/lambda.hat)^beta.hat))-
((T[3]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                ((exp(-S[4]*theta.hat))*(exp(-S[3]*theta.hat))*(exp(-S[2]*theta.hat))*(exp(-
S[1]*theta.hat))*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-((T[4]^beta.hat)*((1/lambda.hat)^beta.hat))))}

##########################################################################################
```

```
linha<-1
linha2<-1

for (sistema in 1:m) {

        dados<-matrix(0,tam[sistema],3)

        # Entrada de dados do sistema
        for (k in 1:tam[sistema]) {
                dados[k,1]<-dados_bruto[linha,1]
                dados[k,2]<-dados_bruto[linha,2]
                dados[k,3]<-dados_bruto[linha,3]
                linha<-linha+1}

        total<-nrow(dados)
        ciclos<-dados[total,2]

        # Cria vetor com os tempos de falha e severidades da MP
        falhas<-matrix(0,(total-ciclos),2)

        # Cria vetor com os tempos de MP e Severidades
        T<-vector(mode="integer",length=ciclos)
        S<-vector(mode="integer",length=ciclos)

        j<-1
        k<-1

        for (i in 1:(total-1)){
                if (dados[i,2]<dados[i+1,2]) {T[k]<-dados[i,1]
                                              S[k]<-dados[i,3]
                                              k<-k+1 }
                else {  falhas[j,1]<-dados[i,1]
                        falhas[j,2]<-dados[i,2]
                        j<-j+1}

                if (i+1==total)         {T[k]<-dados[i+1,1]
                                         S[k]<-dados[i+1,3]}
        }
        if (ciclos==1) {mit<-mi1(T[ciclos])}
        if (ciclos==2) {mit<-mi2(T[ciclos])}
        if (ciclos==3) {mit<-mi3(T[ciclos])}
        if (ciclos==4) {mit<-mi4(T[ciclos])}
        if (ciclos==5) {mit<-mi5(T[ciclos])}

        tot_falhas<-nrow(falhas)

        for (i in 1:tot_falhas){
                if (falhas[i,2]==1) {CVM_aux[linha2,1]<-mi1(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}
                if (falhas[i,2]==2) {CVM_aux[linha2,1]<-mi2(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}
                if (falhas[i,2]==3) {CVM_aux[linha2,1]<-mi3(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}
                if (falhas[i,2]==4) {CVM_aux[linha2,1]<-mi4(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}
                if (falhas[i,2]==5) {CVM_aux[linha2,1]<-mi5(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}
        # a coluna 1 da matriz CVM_aux tem as estimativas de mi(t)

        }
}
```

```
##########################################################################################
# Formato da matriz do Teste Cramer-von Mises (CVM)
# 1a coluna - Ordem da falha
# 2a Coluna - Ri estimado pela Função mi
# 3a coluna - Ri estimado pela Ordem da falha
# 4a coluna - Diferença ao quadrado 2a e 3a colunas
##########################################################################################
CVM<-matrix(0,M,4)

ord<-order(CVM_aux[,2])

for (i in 1:M) {
        CVM[i,1]<-i
        CVM[i,2]<-CVM_aux[ord[i],2]
        CVM[i,3]<-(((2*CVM[i,1])-1)/(2*M))
        CVM[i,4]<-(CVM[i,2]-CVM[i,3])^2
}

#Estatística do Teste CVM
CVM2<-(1/(12*M))+sum(CVM[,4])

# H0: Modelo segue NHPP com u(t)
# Se CVM2 < Valor Crítico, então aceita H0.

##########################################################################################
# Gráfico para avaliação da Bondade do Ajuste do Modelo
##########################################################################################
# Considera o último tempo de cada ciclo como a MP
# Se o tempo da MP for falha, deve replicar esse valor nos dados_bruto
# Essa duplicação deve ser feita apenas para construção deste gráfico
##########################################################################################
# Entrada de Dados
dados_bruto<-read.table(file="dados.txt", sep = " ")

# Número de sistemas analisados
m<-max(dados_bruto[,4])

# tam: quantidade de dados para cada sistema
# ciclo: quantidade de ciclos para cada sistema
tam<-vector(mode="integer",length=m)
ciclo<-vector(mode="integer",length=m)

# tempo_sist: ultimo tempo do sistema
tempo_sist<-vector(mode="integer",length=m)

for (k in 1:nrow(dados_bruto)) {
        tam[dados_bruto[k,4]]<-tam[dados_bruto[k,4]]+1
        if ( k==nrow(dados_bruto) ) { ciclo[dados_bruto[k,4]]<-dados_bruto[k,2]
                                      tempo_sist[dados_bruto[k,4]]<-dados_bruto[k,1]}
        else { if (dados_bruto[k,4]<dados_bruto[k+1,4]) {
                ciclo[dados_bruto[k,4]]<-dados_bruto[k,2]
                tempo_sist[dados_bruto[k,4]]<-dados_bruto[k,1]}
        }
}

#Número Total de falhas considerando todos os sistemas
M<-sum(tam)-sum(ciclo)

MCF_aux<-matrix(0,M,m+1)

MCF<-matrix(0,M,4)

info_ciclo<-matrix(0,sum(ciclo),3)
k<-1
linha2<-1
auxMP<-0
```

```
# Retirando os tempos de MP dos ciclos ou do sistema e ficando somente as falhas em MCF_aux
# info_ciclo tem a informação dos tempos finais (coluna 1) de cada ciclo (coluna 2) dos sistemas (coluna 3)
for (linha in 1:sum(tam)) {
        if ( linha==nrow(dados_bruto) ) { info_ciclo[linha2,1]<-dados_bruto[linha,1]
                                          info_ciclo[linha2,2]<-dados_bruto[linha,2]
                                          info_ciclo[linha2,3]<-dados_bruto[linha,4]
        }
        else {      if (dados_bruto[linha,2]<dados_bruto[linha+1,2]) {info_ciclo[linha2,1]<-dados_bruto[linha,1]
                                                                       info_ciclo[linha2,2]<-dados_bruto[linha,2]
                                                                       info_ciclo[linha2,3]<-dados_bruto[linha,4]
                                                                       linha2<-linha2+1
                                                                       auxMP<-1
                    }

                    if (dados_bruto[linha,4]<dados_bruto[linha+1,4]) {info_ciclo[linha2,1]<-dados_bruto[linha,1]
                                                                       info_ciclo[linha2,2]<-dados_bruto[linha,2]
                                                                       info_ciclo[linha2,3]<-dados_bruto[linha,4]
                                                                       linha2<-linha2+1
                                                                       auxMP<-1
                    }

                    if (auxMP==0) {MCF_aux[k,1]<-dados_bruto[linha,1]
                                   k<-k+1
                    }
                    auxMP<-0
          }
}

# Cada coluna da matriz MCF_aux (a partir da 2a) representa um sistema
# Determina em que ciclo esta cada tempo de falha da matriz MCF_aux para os diferentes sistemas
for (i in 1:M) {
        linha<-1

        for (sistema in 1:m) {
                    while (info_ciclo[linha,3]==sistema){
                              if(MCF_aux[i,1]>=info_ciclo[linha,1]) {
                                   MCF_aux[i,sistema+1]<-info_ciclo[linha,2]+1
                                   linha<-linha+1
                              }
                              else { MCF_aux[i,sistema+1]<-info_ciclo[linha,2]
                                   while (info_ciclo[linha,3]==sistema){ linha<-linha+1
                                                                         if(linha==sum(ciclo)+1){
                                                                                   sistema<-sistema+1
                                                                                   linha<-1}
                                   }
                              }
                    }
        }
}

##########################################################################################
# Definição das funções mi para os 5 possíveis ciclos
##########################################################################################
mi1<-function(t){((t^beta.hat)*((1/lambda.hat)^beta.hat))}

mi2<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                    ((exp(-S[1]*theta.hat))*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))))}

mi3<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                    ((exp(-S[1]*theta.hat))*(((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))-((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                    ((exp(-S[2]*theta.hat))*(exp(-S[1]*theta.hat))*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-
((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))))}

mi4<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                    ((exp(-S[1]*theta.hat))*(((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))-((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))))+
```

```
                    ((exp(-S[2]*theta.hat))*(exp(-S[1]*theta.hat))*(((T[3]^beta.hat)*((1/lambda.hat)^beta.hat))-
((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                    ((exp(-S[3]*theta.hat))*(exp(-S[2]*theta.hat))*(exp(-S[1]*theta.hat))*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-
((T[3]^beta.hat)*((1/lambda.hat)^beta.hat))))}

mi5<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                    ((exp(-S[1]*theta.hat))*(((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))-((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                    ((exp(-S[2]*theta.hat))*(exp(-S[1]*theta.hat))*(((T[3]^beta.hat)*((1/lambda.hat)^beta.hat))-
((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                    ((exp(-S[3]*theta.hat))*(exp(-S[2]*theta.hat))*(exp(-S[1]*theta.hat))*(((T[4]^beta.hat)*((1/lambda.hat)^beta.hat))-
((T[3]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                    ((exp(-S[4]*theta.hat))*(exp(-S[3]*theta.hat))*(exp(-S[2]*theta.hat))*(exp(-
S[1]*theta.hat))*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-((T[4]^beta.hat)*((1/lambda.hat)^beta.hat))))}

##################################################################################
# Cálculo da estimativa do MCF do processo
for (i in 1:M) {

        MCF[i,1]<-MCF_aux[i,1] #Recebe o tempo de falha
        mit<-0
        linha<-1

        for (sistema in 1:m) {

                dados<-matrix(0,tam[sistema],3)

                # Entrada de dados do sistema
                for (k in 1:tam[sistema]) {
                        dados[k,1]<-dados_bruto[linha,1]
                        dados[k,2]<-dados_bruto[linha,2]
                        dados[k,3]<-dados_bruto[linha,3]
                        linha<-linha+1}

                total<-nrow(dados)
                ciclos<-dados[total,2]

                # Cria vetor com os tempos de falha e severidades da MP
                falhas<-matrix(0,(total-ciclos),2)

                # Cria vetor com os tempos de MP e Severidades
                T<-vector(mode="integer",length=ciclos)
                S<-vector(mode="integer",length=ciclos)

                j<-1
                k<-1

                for (q in 1:(total-1)){
                        if (dados[q,2]<dados[q+1,2]) {T[k]<-dados[q,1]
                                                        S[k]<-dados[q,3]
                                                        k<-k+1 }
                        else {  falhas[j,1]<-dados[q,1]
                                falhas[j,2]<-dados[q,2]
                                j<-j+1}
                        if (q+1==total)         {T[k]<-dados[q+1,1]
                                                 S[k]<-dados[q+1,3]}
                }
                # Verficar em que ciclo esta cada sistema, para utilizar a mi correspondente
                if (MCF_aux[i,sistema+1]==1) {mit<-mit+mi1(MCF[i,1])}
                if (MCF_aux[i,sistema+1]==2) {mit<-mit+mi2(MCF[i,1])}
                if (MCF_aux[i,sistema+1]==3) {mit<-mit+mi3(MCF[i,1])}
                if (MCF_aux[i,sistema+1]==4) {mit<-mit+mi4(MCF[i,1])}
                if (MCF_aux[i,sistema+1]==5) {mit<-mit+mi5(MCF[i,1])}


        }
```

```
        MCF[i,2]<-mit/m #Estimativa do MCF do processo


}

ord<-order(MCF[,1])
MCF<-MCF[ord,]

# Cálculo da MCF empírica do processo
for (i in 1:M) {

        ri<-0

        for (sistema in 1:m) {
                if(MCF[i,1]<=tempo_sist[sistema]){ri<-ri+1}
        }

        if(i==1) {MCF[i,3]<-(1/ri)}
        else {MCF[i,3]<-(1/ri)+MCF[i-1,3]}
        MCF[i,4]<-ri
}

# Gráfico MCF estimada X MCF empírica
MCF_estimado<-MCF[,2]
MCF_empirico<-MCF[,3]
plot(MCF_estimado,MCF_empirico)
abline(0,1)

##########################################################################################
# RELATÓRIO FINAL
##########################################################################################

##########################################################################################
# Resumo Verificações
##########################################################################################
# Deve ser zero para indicar que o algoritmo convergiu
convergencia

if(detver1>=0){"Verificar"}
if(detver2<=0){"Verificar"}
if(detver3>=0){"Verificar"}

if(av$values[1]>=0){"Verificar"}
if(av$values[2]>=0){"Verificar"}
if(av$values[3]>=0){"Verificar"}

##########################################################################################
# Estimativas
##########################################################################################
lambda.hat
IC_lambda_Inf
IC_lambda_Sup

beta.hat
IC_beta_Inf
IC_beta_Sup

theta.hat
IC_theta_Inf
IC_theta_Sup
```

```
##########################################################################################
# Teste de Razão de Verossimilhanças
##########################################################################################
# P-valor (H0: theta=0)
1-pchisq(TRV, 1)

# Estatística TRV
TRV

# Valor Crítico (VC)
qchisq(0.95,1)

# Se TRV < VC, então aceita H0

##########################################################################################
# Teste Cramer-von Mises
##########################################################################################
#Estatística do Teste CVM
CVM2

# H0: Modelo segue NHPP com u(t)
# Se CVM2 < Valor Crítico, então aceita H0

##########################################################################################
# Otimização Tempo de MP
##########################################################################################
# Custo Manutenção Corretiva
CMC<-1

# custo Manutenção Preventiva
CMP<-5

# Tempo da realização da MP
MP<-T[4]

# Severidade da MP realizada
SMP<-S[4]

# mi=Número de falhas esperadas ultima MP até t
mi<-function(t){((exp(-SMP*theta.hat))*(exp(-S[3]*theta.hat))*(exp(-S[2]*theta.hat))*(exp(-
S[1]*theta.hat))*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-((MP^beta.hat)*((1/lambda.hat)^beta.hat))))}

# Função Custo Médio
FCM<-function(t) { ((CMC*mi(t+MP))+(CMP))/t }

# Otimização do Tempo para realização de MP
TO<-optimize(FCM, c(0,500))

TO$minimum+MP

TO$objective

##########################################################################################
# Função Confiabilidade
##########################################################################################
R<-function(t){exp(-mi(MP+t))}
##########################################################################################
```

## A.8 - Programa referente ao modelo de Redução de Intensidade Falha com relação Linear (modelo RFL)

```
##################################################################################
# Formato do arquivo de entrada dados.txt
# 1a coluna - Tempos de falha acumulados (o ultimo valor do ciclo deve ser a MP)
# 2a Coluna - Número do ciclo de MP ordenado
# 3a coluna - BSC da MP realizada que influênciará o próximo ciclo de MP
# 4a coluna - Número do sistema ordenado
##################################################################################
library(nlme)
##################################################################################
# Função da LogVerossimilhança para o Modelos de Confiabilidade de Sistemas
# Reparáveis considerando MP Imperfeita
##################################################################################
logvero <- function(x){
	# Entrada dos Parâmetros
	lambda<-x[1]
	beta<-x[2]
	theta<-x[3]

	lv<-0

	# Entrada de Dados
	dados_bruto<-read.table(file="dados.txt", sep = " ")

	# Número de Sistemas analisados
	m<-max(dados_bruto[,4])

	tam<-vector(mode="integer",length=m)

	# Define a quantidade de dados para cada sistema
	for (k in 1:nrow(dados_bruto)) {
		tam[dados_bruto[k,4]]<-tam[dados_bruto[k,4]]+1
	}
	linha<-1

	for (sistema in 1:m) {

		dados<-matrix(0,tam[sistema],3)

		# Entrada de dados do sistema k
		for (k in 1:tam[sistema]) {
			dados[k,1]<-dados_bruto[linha,1]
			dados[k,2]<-dados_bruto[linha,2]
			dados[k,3]<-dados_bruto[linha,3]
			linha<-linha+1		}

		total<-nrow(dados)

		ciclos<-dados[total,2]

		n<-vector(mode="integer",length=ciclos)
		T<-vector(mode="integer",length=ciclos)
		S<-vector(mode="integer",length=ciclos)

		# Cria vetor com os tempos e severidade da MP em cada ciclo
		# e o número de falhas em cada cilco
		for (i in 1:(total-1)){

			if (dados[i,2]<dados[i+1,2]) {T[dados[i,2]]<-dados[i,1]
								S[dados[i,2]]<-dados[i,3]}

			else {n[dados[i,2]]<-n[dados[i,2]]+1}
```

```
}

T[dados[i+1,2]]<-dados[i+1,1]
S[dados[i+1,2]]<-dados[i+1,3]

# Número total de falhas
N<-sum(n)

# Cria vetor com os tempos de falha e severidades da MP
t<-matrix(0,ciclos,max(n))

j<-1
for (i in 1:(total-1)){

        if (dados[i,2]<dados[i+1,2]) { j<-1 }

        else {t[dados[i,2],j]<-dados[i,1]
            j<-j+1}
}

#######################################################################
# Cálculo da LogVerossimilhança
#######################################################################
        lv<-lv+(N*log(beta))+(N*log(((1/lambda)^beta)))

Va<-0 #Idade inicial

aux<-0
j<-1
for (i in 1:n[j]){
        if(n[j]>0) {aux<-aux+log(t[j,i])}
}
lv<-lv+((beta-1)*aux)
lv<-lv-(((1/lambda)^beta)*((T[j]^beta)-(Va^beta)))

Va<-T[j]

for (j in 2:ciclos){
        aux<-0
        for (i in 1:n[j]){
                if(n[j]>0) {aux<-aux+log(t[j,i])}
        }
        lv<-lv+((beta-1)*aux)

        if (j==2) {
                lv<-lv+((n[j])*log((1-(S[j-1]*theta))))
                lv<-lv-(((1-(S[j-1]*theta))*(((1/lambda)^beta)))*((T[j]^beta)-(Va^beta)))}
        if (j==3) {
                lv<-lv+((n[j])*log((1-(S[j-2]*theta))*(1-(S[j-1]*theta))))
                lv<-lv-(((1-(S[j-2]*theta))*(1-(S[j-1]*theta))*(((1/lambda)^beta)))*((T[j]^beta)-(Va^beta)))}
        if (j==4) {
                lv<-lv+((n[j])*log((1-(S[j-3]*theta))*(1-(S[j-2]*theta))*(1-(S[j-1]*theta))))

                lv<-lv-(((1-(S[j-3]*theta))*(1-(S[j-2]*theta))*(1-(S[j-
1]*theta))*(((1/lambda)^beta)))*((T[j]^beta)-(Va^beta)))}
        if (j==5) {
                lv<-lv+((n[j])*log((1-(S[j-4]*theta))*(1-(S[j-3]*theta))*(1-(S[j-2]*theta))*(1-(S[j-1]*theta))))

                lv<-lv-(((1-(S[j-4]*theta))*(1-(S[j-3]*theta))*(1-(S[j-2]*theta))*(1-(S[j-
1]*theta))*(((1/lambda)^beta)))*((T[j]^beta)-(Va^beta)))}

        Va<-T[j]

}
#######################################################################
}
```

```
        restricao<-0
        for (j in 1:ciclos){
                if( (1-(S[j]*theta)) < 0 ) {restricao<-1}
                if( (1-(S[j]*theta)) > 1 ) {restricao<-1}
        }

        if(restricao==1) {return(-Inf)} else{return(lv)}
}

##########################################################################################
# O método da função optim de otimização encontra o mínimo, portanto para determinar
# o máximo da função LogVerossimilhança foi utilizado seu valor negativo
##########################################################################################
logvero_opt <- function(x){

        lambda<-x[1]
        beta<-x[2]
        theta<-x[3]
        return(-logvero(c(lambda,beta,theta)))
}
##########################################################################################
# Otimização da LogVerossimilhança
##########################################################################################
mle<-optim(c(100, 1.8, 0.8), logvero_opt, method = "Nelder-Mead")
##########################################################################################
# PARÂMETROS ESTIMADOS
lambda.hat<-mle$par[1]
beta.hat<-mle$par[2]
theta.hat<-mle$par[3]

param<-c(mle$par[1], mle$par[2], mle$par[3])

# Deve ser zero para indicar que o algoritmo convergiu
convergencia<-mle$convergence

mat<-fdHess(mle$par, logvero)
hessiano<-mat$Hessian

# Para ser ponto de máximo devem ser alternados os sinais

# VERIFICAÇÃO 1 (Negativo)
detver1<-hessiano[1,1]

#VERIFICAÇÃO 2 (Positivo)
ver2<-matrix(0,2,2)
ver2[1,1]<-hessiano[1,1]
ver2[1,2]<-hessiano[1,2]
ver2[2,1]<-hessiano[2,1]
ver2[2,2]<-hessiano[2,2]

detver2<-det(ver2)

#VERIFICAÇÃO 3 (Negativo)
ver3<-matrix(0,3,3)
ver3[1,1]<-hessiano[1,1]
ver3[1,2]<-hessiano[1,2]
ver3[1,3]<-hessiano[1,3]
ver3[2,1]<-hessiano[2,1]
ver3[2,2]<-hessiano[2,2]
ver3[2,3]<-hessiano[2,3]
ver3[3,1]<-hessiano[3,1]
ver3[3,2]<-hessiano[3,2]
ver3[3,3]<-hessiano[3,3]

detver3<-det(ver3)
```

```
# Outra verificação: se os Autovalores forem todos negativos, é ponto de máximo.
# VERIFICAÇÃO AUTOVALORES
av<-eigen(hessiano)

##################################################################################
# Resumo Verificações
##################################################################################
if(detver1<0){"OK"}else{"Verificar"}
if(detver2>0){"OK"}else{"Verificar"}
if(detver3<0){"OK"}else{"Verificar"}

if(av$values[1]<0){"OK"}else{"Verificar"}
if(av$values[2]<0){"OK"}else{"Verificar"}
if(av$values[3]<0){"OK"}else{"Verificar"}

##################################################################################
# Desvios Padrãoes das estimativas
##################################################################################
dp.lambda.hat<-sqrt(solve(-hessiano)[1,1])
dp.beta.hat<-sqrt(solve(-hessiano)[2,2])
dp.theta.hat<-sqrt(solve(-hessiano)[3,3])

# Nível de Confiança
NC<-0.95

IC_lambda_Inf<-lambda.hat/(exp(qnorm((1+NC)/2)*dp.lambda.hat/lambda.hat))
IC_lambda_Sup<-lambda.hat*(exp(qnorm((1+NC)/2)*dp.lambda.hat/lambda.hat))

IC_beta_Inf<-beta.hat/(exp(qnorm((1+NC)/2)*dp.beta.hat/beta.hat))
IC_beta_Sup<-beta.hat*(exp(qnorm((1+NC)/2)*dp.beta.hat/beta.hat))

IC_theta_Inf<-theta.hat/(exp(qnorm((1+NC)/2)*dp.theta.hat/theta.hat))
IC_theta_Sup<-theta.hat*(exp(qnorm((1+NC)/2)*dp.theta.hat/theta.hat))

##################################################################################
# Verossimilhança da Manutenção Sem Efeito (MSE)
##################################################################################
logvero_mse <- function(x){

        lambda<-x[1]
        beta<-x[2]
        theta<-0
        return(-logvero(c(lambda,beta,theta)))
}

mle_mse<-optim(c(100, 1.5), logvero_mse, method = "Nelder-Mead")

param_mse<-c(mle_mse$par[1], mle_mse$par[2])

##################################################################################
# Teste de Razão de Verossimilhanças
##################################################################################
TRV<-(-2*(logvero(c(param_mse,0))-(logvero(param))))

# P-valor (H0: theta=0)
1-pchisq(TRV, 1)

# Estatística TRV
TRV

# Valor Crítico (VC)
qchisq(0.95,1)

# Se TRV < VC, então aceita H0 (não há efeito entre as MP)
```

```
##########################################################################################
# Teste Cramer-von Mises
##########################################################################################
# Considera o último tempo de cada ciclo como a MP
# É considerado nos cálculos se o último tempo do sistema for falha ou MP
##########################################################################################
# Entrada de Dados
dados_bruto<-read.table(file="dados.txt", sep = " ")

# Número de sistemas analisados
m<-max(dados_bruto[,4])

# tam: quantidade de dados para cada sistema
# ciclo: quantidade de ciclos para cada sistema
tam<-vector(mode="integer",length=m)
ciclo<-vector(mode="integer",length=m)

for (k in 1:nrow(dados_bruto)) {
		tam[dados_bruto[k,4]]<-tam[dados_bruto[k,4]]+1
		if ( k==nrow(dados_bruto) ) { ciclo[dados_bruto[k,4]]<-dados_bruto[k,2] }
		else { if (dados_bruto[k,4]<dados_bruto[k+1,4]) { ciclo[dados_bruto[k,4]]<-dados_bruto[k,2] }}
}

#Número Total de falhas considerando todos os sistemas
M<-sum(tam)-sum(ciclo)

##########################################################################################

CVM_aux<-matrix(0,M,2)

##########################################################################################
# Definição das funções mi para os 5 possíveis ciclos
##########################################################################################
mi1<-function(t){((t^beta.hat)*((1/lambda.hat)^beta.hat))}

mi2<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
			((1-(S[1]*theta.hat))*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))))}

mi3<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
			((1-(S[1]*theta.hat))*(((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))-((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))))+
			((1-(S[2]*theta.hat))*(1-(S[1]*theta.hat))*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-
((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))))}

mi4<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
			((1-(S[1]*theta.hat))*(((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))-((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))))+
			((1-(S[2]*theta.hat))*(1-(S[1]*theta.hat))*(((T[3]^beta.hat)*((1/lambda.hat)^beta.hat))-
((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))))+
			((1-(S[3]*theta.hat))*(1-(S[2]*theta.hat))*(1-(S[1]*theta.hat))*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-
((T[3]^beta.hat)*((1/lambda.hat)^beta.hat))))}

mi5<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
			((1-(S[1]*theta.hat))*(((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))-((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))))+
			((1-(S[2]*theta.hat))*(1-(S[1]*theta.hat))*(((T[3]^beta.hat)*((1/lambda.hat)^beta.hat))-
((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))))+
			((1-(S[3]*theta.hat))*(1-(S[2]*theta.hat))*(1-(S[1]*theta.hat))*(((T[4]^beta.hat)*((1/lambda.hat)^beta.hat))-
((T[3]^beta.hat)*((1/lambda.hat)^beta.hat))))+
			((1-(S[4]*theta.hat))*(1-(S[3]*theta.hat))*(1-(S[2]*theta.hat))*(1-
(S[1]*theta.hat))*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-((T[4]^beta.hat)*((1/lambda.hat)^beta.hat))))}

##########################################################################################
```

```
linha<-1
linha2<-1

for (sistema in 1:m) {
        dados<-matrix(0,tam[sistema],3)

        # Entrada de dados do sistema
        for (k in 1:tam[sistema]) {
                dados[k,1]<-dados_bruto[linha,1]
                dados[k,2]<-dados_bruto[linha,2]
                dados[k,3]<-dados_bruto[linha,3]
                linha<-linha+1}

        total<-nrow(dados)
        ciclos<-dados[total,2]

        # Cria vetor com os tempos de falha e severidades da MP
        falhas<-matrix(0,(total-ciclos),2)

        # Cria vetor com os tempos de MP e Severidades
        T<-vector(mode="integer",length=ciclos)
        S<-vector(mode="integer",length=ciclos)

        j<-1
        k<-1

        for (i in 1:(total-1)){
                if (dados[i,2]<dados[i+1,2]) {T[k]<-dados[i,1]
                                              S[k]<-dados[i,3]
                                              k<-k+1 }
                else {  falhas[j,1]<-dados[i,1]
                        falhas[j,2]<-dados[i,2]
                        j<-j+1}

                if (i+1==total)         {T[k]<-dados[i+1,1]
                                         S[k]<-dados[i+1,3]}
        }
        if (ciclos==1) {mit<-mi1(T[ciclos])}
        if (ciclos==2) {mit<-mi2(T[ciclos])}
        if (ciclos==3) {mit<-mi3(T[ciclos])}
        if (ciclos==4) {mit<-mi4(T[ciclos])}
        if (ciclos==5) {mit<-mi5(T[ciclos])}

        tot_falhas<-nrow(falhas)

        for (i in 1:tot_falhas){
                if (falhas[i,2]==1) {CVM_aux[linha2,1]<-mi1(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}
                if (falhas[i,2]==2) {CVM_aux[linha2,1]<-mi2(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}
                if (falhas[i,2]==3) {CVM_aux[linha2,1]<-mi3(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}
                if (falhas[i,2]==4) {CVM_aux[linha2,1]<-mi4(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}
                if (falhas[i,2]==5) {CVM_aux[linha2,1]<-mi5(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}
        # a coluna 1 da matriz CVM_aux tem as estimativas de mi(t)

        }
}
```

```
##################################################################################
# Formato da matriz do Teste Cramer-von Mises (CVM)
# 1a coluna - Ordem da falha
# 2a Coluna - Ri estimado pela Função mi
# 3a coluna - Ri estimado pela Ordem da falha
# 4a coluna - Diferença ao quadrado 2a e 3a colunas
##################################################################################
CVM<-matrix(0,M,4)

ord<-order(CVM_aux[,2])

for (i in 1:M) {
        CVM[i,1]<-i
        CVM[i,2]<-CVM_aux[ord[i],2]
        CVM[i,3]<-(((2*CVM[i,1])-1)/(2*M))
        CVM[i,4]<-(CVM[i,2]-CVM[i,3])^2
}

#Estatística do Teste CVM
CVM2<-(1/(12*M))+sum(CVM[,4])

# H0: Modelo segue NHPP com u(t)
# Se CVM2 < Valor Crítico, então aceita H0.

##################################################################################
# Gráfico para avaliação da Bondade do Ajuste do Modelo
##################################################################################
# Considera o último tempo de cada ciclo como a MP
# Se o tempo da MP for falha, deve replicar esse valor nos dados_bruto
# Essa duplicação deve ser feita apenas para construção deste gráfico
##################################################################################
# Entrada de Dados
dados_bruto<-read.table(file="dados.txt", sep = " ")

# Número de sistemas analisados
m<-max(dados_bruto[,4])

# tam: quantidade de dados para cada sistema
# ciclo: quantidade de ciclos para cada sistema
tam<-vector(mode="integer",length=m)
ciclo<-vector(mode="integer",length=m)

# tempo_sist: ultimo tempo do sistema
tempo_sist<-vector(mode="integer",length=m)

for (k in 1:nrow(dados_bruto)) {
        tam[dados_bruto[k,4]]<-tam[dados_bruto[k,4]]+1
        if ( k==nrow(dados_bruto) ) { ciclo[dados_bruto[k,4]]<-dados_bruto[k,2]
                                      tempo_sist[dados_bruto[k,4]]<-dados_bruto[k,1]}
        else { if (dados_bruto[k,4]<dados_bruto[k+1,4]) {
                ciclo[dados_bruto[k,4]]<-dados_bruto[k,2]
                tempo_sist[dados_bruto[k,4]]<-dados_bruto[k,1]}
        }
}

#Número Total de falhas considerando todos os sistemas
M<-sum(tam)-sum(ciclo)

MCF_aux<-matrix(0,M,m+1)
MCF<-matrix(0,M,4)

info_ciclo<-matrix(0,sum(ciclo),3)
k<-1
linha2<-1
auxMP<-0
```

```
# Retirando os tempos de MP dos ciclos ou do sistema e ficando somente as falhas em MCF_aux
# info_ciclo tem a informação dos tempos finais (coluna 1) de cada ciclo (coluna 2) dos sistemas (coluna 3)
for (linha in 1:sum(tam)) {
        if ( linha==nrow(dados_bruto) ) { info_ciclo[linha2,1]<-dados_bruto[linha,1]
                                          info_ciclo[linha2,2]<-dados_bruto[linha,2]
                                          info_ciclo[linha2,3]<-dados_bruto[linha,4]
        }
        else {      if (dados_bruto[linha,2]<dados_bruto[linha+1,2]) {info_ciclo[linha2,1]<-dados_bruto[linha,1]
                                                                       info_ciclo[linha2,2]<-dados_bruto[linha,2]
                                                                       info_ciclo[linha2,3]<-dados_bruto[linha,4]
                                                                       linha2<-linha2+1
                                                                       auxMP<-1
                    }


                    if (dados_bruto[linha,4]<dados_bruto[linha+1,4]) {info_ciclo[linha2,1]<-dados_bruto[linha,1]
                                                                       info_ciclo[linha2,2]<-dados_bruto[linha,2]
                                                                       info_ciclo[linha2,3]<-dados_bruto[linha,4]
                                                                       linha2<-linha2+1
                                                                       auxMP<-1
                    }

                    if (auxMP==0) {MCF_aux[k,1]<-dados_bruto[linha,1]
                                   k<-k+1
                    }
                    auxMP<-0
            }
}

# Cada coluna da matriz MCF_aux (a partir da 2a) representa um sistema
# Determina em que ciclo esta cada tempo de falha da matriz MCF_aux para os diferentes sistemas
for (i in 1:M) {

        linha<-1

        for (sistema in 1:m) {
                    while (info_ciclo[linha,3]==sistema){
                            if(MCF_aux[i,1]>=info_ciclo[linha,1]) {
                                MCF_aux[i,sistema+1]<-info_ciclo[linha,2]+1
                                linha<-linha+1
                            }
                            else { MCF_aux[i,sistema+1]<-info_ciclo[linha,2]
                                while (info_ciclo[linha,3]==sistema){ linha<-linha+1
                                                                      if(linha==sum(ciclo)+1){
                                                                                  sistema<-sistema+1
                                                                                  linha<-1}
                                }
                            }
                    }
        }
}


##########################################################################################
# Definição das funções mi para os 5 possíveis ciclos
##########################################################################################
mi1<-function(t){((t^beta.hat)*((1/lambda.hat)^beta.hat))}

mi2<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                 ((1-(S[1]*theta.hat))*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))))}

mi3<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                 ((1-(S[1]*theta.hat))*(((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))-((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                 ((1-(S[2]*theta.hat))*(1-(S[1]*theta.hat))*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-
((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))))}
```

```
mi4<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                    ((1-(S[1]*theta.hat))*(((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))-((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                    ((1-(S[2]*theta.hat))*(1-(S[1]*theta.hat))*(((T[3]^beta.hat)*((1/lambda.hat)^beta.hat))-
((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                    ((1-(S[3]*theta.hat))*(1-(S[2]*theta.hat))*(1-(S[1]*theta.hat))*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-
((T[3]^beta.hat)*((1/lambda.hat)^beta.hat))))}

mi5<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                    ((1-(S[1]*theta.hat))*(((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))-((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                    ((1-(S[2]*theta.hat))*(1-(S[1]*theta.hat))*(((T[3]^beta.hat)*((1/lambda.hat)^beta.hat))-
((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                    ((1-(S[3]*theta.hat))*(1-(S[2]*theta.hat))*(1-(S[1]*theta.hat))*(((T[4]^beta.hat)*((1/lambda.hat)^beta.hat))-
((T[3]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                    ((1-(S[4]*theta.hat))*(1-(S[3]*theta.hat))*(1-(S[2]*theta.hat))*(1-
(S[1]*theta.hat))*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-((T[4]^beta.hat)*((1/lambda.hat)^beta.hat))))}

##################################################################################
# Cálculo da estimativa do MCF do processo
for (i in 1:M) {

        MCF[i,1]<-MCF_aux[i,1] #Recebe o tempo de falha
        mit<-0

        linha<-1

        for (sistema in 1:m) {

                dados<-matrix(0,tam[sistema],3)

                # Entrada de dados do sistema
                for (k in 1:tam[sistema]) {
                        dados[k,1]<-dados_bruto[linha,1]
                        dados[k,2]<-dados_bruto[linha,2]
                        dados[k,3]<-dados_bruto[linha,3]
                        linha<-linha+1}

                total<-nrow(dados)
                ciclos<-dados[total,2]

                # Cria vetor com os tempos de falha e severidades da MP
                falhas<-matrix(0,(total-ciclos),2)

                # Cria vetor com os tempos de MP e Severidades
                T<-vector(mode="integer",length=ciclos)
                S<-vector(mode="integer",length=ciclos)

                j<-1
                k<-1

                for (q in 1:(total-1)){
                        if (dados[q,2]<dados[q+1,2]) {T[k]<-dados[q,1]
                                                        S[k]<-dados[q,3]
                                                        k<-k+1 }
                        else {  falhas[j,1]<-dados[q,1]
                                falhas[j,2]<-dados[q,2]
                                j<-j+1}

                        if (q+1==total)         {T[k]<-dados[q+1,1]
                                                 S[k]<-dados[q+1,3]}
                }

                # Verficar em que ciclo esta cada sistema, para utilizar a mi correspondente
                if (MCF_aux[i,sistema+1]==1) {mit<-mit+mi1(MCF[i,1])}

                if (MCF_aux[i,sistema+1]==2) {mit<-mit+mi2(MCF[i,1])}
                if (MCF_aux[i,sistema+1]==3) {mit<-mit+mi3(MCF[i,1])}
```

```
                    if (MCF_aux[i,sistema+1]==4) {mit<-mit+mi4(MCF[i,1])}
                    if (MCF_aux[i,sistema+1]==5) {mit<-mit+mi5(MCF[i,1])}
          }
          MCF[i,2]<-mit/m #Estimativa do MCF do processo
}
ord<-order(MCF[,1])
MCF<-MCF[ord,]

# Cálculo da MCF empírica do processo
for (i in 1:M) {
          ri<-0

          for (sistema in 1:m) {
                    if(MCF[i,1]<=tempo_sist[sistema]){ri<-ri+1}

          }

          if(i==1) {MCF[i,3]<-(1/ri)}
          else {MCF[i,3]<-(1/ri)+MCF[i-1,3]}
          MCF[i,4]<-ri

}

# Gráfico MCF estimada X MCF empírica
MCF_estimado<-MCF[,2]
MCF_empirico<-MCF[,3]
plot(MCF_estimado,MCF_empirico)
abline(0,1)

##########################################################################################
# RELATÓRIO FINAL
##########################################################################################

##########################################################################################
# Resumo Verificações
##########################################################################################
# Deve ser zero para indicar que o algoritmo convergiu
convergencia

if(detver1>=0){"Verificar"}
if(detver2<=0){"Verificar"}
if(detver3>=0){"Verificar"}

if(av$values[1]>=0){"Verificar"}
if(av$values[2]>=0){"Verificar"}
if(av$values[3]>=0){"Verificar"}

##########################################################################################
# Estimativas
##########################################################################################
lambda.hat
IC_lambda_Inf
IC_lambda_Sup

beta.hat
IC_beta_Inf
IC_beta_Sup

theta.hat
IC_theta_Inf
IC_theta_Sup
```

```
##########################################################################################
# Teste de Razão de Verossimilhanças
##########################################################################################
# P-valor (H0: theta=0)
1-pchisq(TRV, 1)

# Estatística TRV
TRV

# Valor Crítico (VC)
qchisq(0.95,1)

# Se TRV < VC, então aceita H0

##########################################################################################
# Teste Cramer-von Mises
##########################################################################################
#Estatística do Teste CVM
CVM2

# H0: Modelo segue NHPP com u(t)
# Se CVM2 < Valor Crítico, então aceita H0

##########################################################################################
# Otimização Tempo de MP
##########################################################################################
# Custo Manutenção Corretiva
CMC<-1

# custo Manutenção Preventiva
CMP<-5

# Tempo da realização da MP
MP<-T[4]

# Severidade da MP realizada
SMP<-S[4]

# mi=Número de falhas esperadas ultima MP até t
mi<-function(t){((1-(SMP*theta.hat))*(1-(S[3]*theta.hat))*(1-(S[2]*theta.hat))*(1-
(S[1]*theta.hat))*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-((MP^beta.hat)*((1/lambda.hat)^beta.hat))))}

# Função Custo Médio
FCM<-function(t) { ((CMC*mi(t+MP))+(CMP))/t }

# Otimização do Tempo para realização de MP
TO<-optimize(FCM, c(0,500))

TO$minimum+MP

TO$objective

##########################################################################################
# Função Confiabilidade
##########################################################################################
R<-function(t){exp(-mi(MP+t))}
##########################################################################################
```

## A.9 - Programa referente ao modelo de Redução de Intensidade Falha com relação Potência (modelo RFP)

```
##################################################################################
# Formato do arquivo de entrada dados.txt
# 1a coluna - Tempos de falha acumulados (o ultimo valor do ciclo deve ser a MP)
# 2a Coluna - Número do ciclo de MP ordenado
# 3a coluna - BSC da MP realizada que influênciará o próximo ciclo de MP
# 4a coluna - Número do sistema ordenado
##################################################################################
library(nlme)
##################################################################################
# Função da LogVerossimilhança para o Modelos de Confiabilidade de Sistemas
# Reparáveis considerando MP Imperfeita
##################################################################################
logvero <- function(x){

	# Entrada dos Parâmetros
	lambda<-x[1]
	beta<-x[2]
	theta<-x[3]

	lv<-0

	# Entrada de Dados
	dados_bruto<-read.table(file="dados.txt", sep = " ")

	# Número de Sistemas analisados
	m<-max(dados_bruto[,4])

	tam<-vector(mode="integer",length=m)

	# Define a quantidade de dados para cada sistema
	for (k in 1:nrow(dados_bruto)) {
		tam[dados_bruto[k,4]]<-tam[dados_bruto[k,4]]+1
	}

	linha<-1
	for (sistema in 1:m) {

		dados<-matrix(0,tam[sistema],3)

		# Entrada de dados do sistema k
		for (k in 1:tam[sistema]) {
			dados[k,1]<-dados_bruto[linha,1]
			dados[k,2]<-dados_bruto[linha,2]
			dados[k,3]<-dados_bruto[linha,3]
			linha<-linha+1		}

		total<-nrow(dados)
		ciclos<-dados[total,2]

		n<-vector(mode="integer",length=ciclos)
		T<-vector(mode="integer",length=ciclos)
		S<-vector(mode="integer",length=ciclos)

		# Cria vetor com os tempos e severidade da MP em cada ciclo
		# e o número de falhas em cada cilco
		for (i in 1:(total-1)){
			if (dados[i,2]<dados[i+1,2]) {T[dados[i,2]]<-dados[i,1]
								S[dados[i,2]]<-dados[i,3]}

			else {n[dados[i,2]]<-n[dados[i,2]]+1}

		}
```

```
T[dados[i+1,2]]<-dados[i+1,1]
S[dados[i+1,2]]<-dados[i+1,3]

# Número total de falhas
N<-sum(n)

# Cria vetor com os tempos de falha e severidades da MP
t<-matrix(0,ciclos,max(n))

j<-1
for (i in 1:(total-1)){
        if (dados[i,2]<dados[i+1,2]) { j<-1 }

        else {t[dados[i,2],j]<-dados[i,1]
            j<-j+1}
}

#######################################################################
# Cálculo da LogVerossimilhança
#######################################################################
lv<-lv+(N*log(beta))+(N*log(((1/lambda)^beta)))

Va<-0 #Idade inicial

aux<-0
j<-1
for (i in 1:n[j]){
        if(n[j]>0) {aux<-aux+log(t[j,i])}
}
lv<-lv+((beta-1)*aux)
lv<-lv-(((1/lambda)^beta)*((T[j]^beta)-(Va^beta)))

Va<-T[j]

for (j in 2:ciclos){

        aux<-0
        for (i in 1:n[j]){
                if(n[j]>0) {aux<-aux+log(t[j,i])}
        }
        lv<-lv+((beta-1)*aux)

        if (j==2) {
                lv<-lv+((n[j])*log((1-(S[j-1]^theta))))
                lv<-lv-(((1-(S[j-1]^theta))*(((1/lambda)^beta)))*((T[j]^beta)-(Va^beta)))}
        if (j==3) {
                lv<-lv+((n[j])*log((1-(S[j-2]^theta))*(1-(S[j-1]^theta))))
                lv<-lv-(((1-(S[j-2]^theta))*(1-(S[j-1]^theta))*(((1/lambda)^beta)))*((T[j]^beta)-(Va^beta)))}
        if (j==4) {
                lv<-lv+((n[j])*log((1-(S[j-3]^theta))*(1-(S[j-2]^theta))*(1-(S[j-1]^theta))))

                lv<-lv-(((1-(S[j-3]^theta))*(1-(S[j-2]^theta))*(1-(S[j-
1]^theta))*(((1/lambda)^beta)))*((T[j]^beta)-(Va^beta)))}
        if (j==5) {
                lv<-lv+((n[j])*log((1-(S[j-4]^theta))*(1-(S[j-3]^theta))*(1-(S[j-2]^theta))*(1-(S[j-1]^theta))))

                lv<-lv-(((1-(S[j-4]^theta))*(1-(S[j-3]^theta))*(1-(S[j-2]^theta))*(1-(S[j-
1]^theta))*(((1/lambda)^beta)))*((T[j]^beta)-(Va^beta)))}

        Va<-T[j]

}

#######################################################################
}
```

```
        restricao<-0
        for (j in 1:ciclos){
                if( (1-(S[j]^theta)) < 0 ) {restricao<-1}
                if( (1-(S[j]^theta)) > 1 ) {restricao<-1}
        }

        if(restricao==1) {return(-Inf)} else{return(lv)}
}

##########################################################################################
# O método da função optim de otimização encontra o mínimo, portanto para determinar
# o máximo da função LogVerossimilhança foi utilizado seu valor negativo
##########################################################################################
logvero_opt <- function(x){

        lambda<-x[1]
        beta<-x[2]
        theta<-x[3]
        return(-logvero(c(lambda,beta,theta)))
}
##########################################################################################
# Otimização da LogVerossimilhança
##########################################################################################
mle<-optim(c(100, 1.8, 0.8), logvero_opt, method = "Nelder-Mead")
##########################################################################################
# PARÂMETROS ESTIMADOS
lambda.hat<-mle$par[1]
beta.hat<-mle$par[2]
theta.hat<-mle$par[3]

param<-c(mle$par[1], mle$par[2], mle$par[3])

# Deve ser zero para indicar que o algoritmo convergiu
convergencia<-mle$convergence

mat<-fdHess(mle$par, logvero)
hessiano<-mat$Hessian

# Para ser ponto de máximo devem ser alternados os sinais

# VERIFICAÇÃO 1 (Negativo)
detver1<-hessiano[1,1]

#VERIFICAÇÃO 2 (Positivo)
ver2<-matrix(0,2,2)
ver2[1,1]<-hessiano[1,1]
ver2[1,2]<-hessiano[1,2]
ver2[2,1]<-hessiano[2,1]
ver2[2,2]<-hessiano[2,2]

detver2<-det(ver2)

#VERIFICAÇÃO 3 (Negativo)
ver3<-matrix(0,3,3)
ver3[1,1]<-hessiano[1,1]
ver3[1,2]<-hessiano[1,2]
ver3[1,3]<-hessiano[1,3]
ver3[2,1]<-hessiano[2,1]
ver3[2,2]<-hessiano[2,2]
ver3[2,3]<-hessiano[2,3]
ver3[3,1]<-hessiano[3,1]
ver3[3,2]<-hessiano[3,2]
ver3[3,3]<-hessiano[3,3]

detver3<-det(ver3)
```

```
# Outra verificação: se os Autovalores forem todos negativos, é ponto de máximo.
# VERIFICAÇÃO AUTOVALORES
av<-eigen(hessiano)

##################################################################################
# Resumo Verificações
##################################################################################
if(detver1<0){"OK"}else{"Verificar"}
if(detver2>0){"OK"}else{"Verificar"}
if(detver3<0){"OK"}else{"Verificar"}

if(av$values[1]<0){"OK"}else{"Verificar"}
if(av$values[2]<0){"OK"}else{"Verificar"}
if(av$values[3]<0){"OK"}else{"Verificar"}

##################################################################################
# Desvios Padrãoes das estimativas
##################################################################################
dp.lambda.hat<-sqrt(solve(-hessiano)[1,1])
dp.beta.hat<-sqrt(solve(-hessiano)[2,2])
dp.theta.hat<-sqrt(solve(-hessiano)[3,3])

# Nível de Confiança
NC<-0.95

IC_lambda_Inf<-lambda.hat/(exp(qnorm((1+NC)/2)*dp.lambda.hat/lambda.hat))
IC_lambda_Sup<-lambda.hat*(exp(qnorm((1+NC)/2)*dp.lambda.hat/lambda.hat))

IC_beta_Inf<-beta.hat/(exp(qnorm((1+NC)/2)*dp.beta.hat/beta.hat))
IC_beta_Sup<-beta.hat*(exp(qnorm((1+NC)/2)*dp.beta.hat/beta.hat))

IC_theta_Inf<-theta.hat/(exp(qnorm((1+NC)/2)*dp.theta.hat/theta.hat))
IC_theta_Sup<-theta.hat*(exp(qnorm((1+NC)/2)*dp.theta.hat/theta.hat))

##################################################################################
# Verossimilhança da Manutenção Sem Efeito (MSE)
##################################################################################
logvero_mse <- function(x){

        lambda<-x[1]
        beta<-x[2]
        theta<-100000
        return(-logvero(c(lambda,beta,theta)))
}

mle_mse<-optim(c(100, 1.5), logvero_mse, method = "Nelder-Mead")
param_mse<-c(mle_mse$par[1], mle_mse$par[2])

##################################################################################
# Teste de Razão de Verossimilhanças
##################################################################################
TRV<-(-2*(logvero(c(param_mse,100000))-(logvero(param))))

# P-valor (H0: S[i]^theta=0)
1-pchisq(TRV, 1)

# Estatística TRV
TRV

# Valor Crítico (VC)
qchisq(0.95,1)

# Se TRV < VC, então aceita H0 (não há efeito entre as MP)
```

```
##########################################################################################
# Teste Cramer-von Mises
##########################################################################################
# Considera o último tempo de cada ciclo como a MP
# É considerado nos cálculos se o último tempo do sistema for falha ou MP
##########################################################################################
# Entrada de Dados
dados_bruto<-read.table(file="dados.txt", sep = " ")

# Número de sistemas analisados
m<-max(dados_bruto[,4])

# tam: quantidade de dados para cada sistema
# ciclo: quantidade de ciclos para cada sistema
tam<-vector(mode="integer",length=m)
ciclo<-vector(mode="integer",length=m)

for (k in 1:nrow(dados_bruto)) {
		tam[dados_bruto[k,4]]<-tam[dados_bruto[k,4]]+1
		if ( k==nrow(dados_bruto) ) { ciclo[dados_bruto[k,4]]<-dados_bruto[k,2] }
		else { if (dados_bruto[k,4]<dados_bruto[k+1,4]) { ciclo[dados_bruto[k,4]]<-dados_bruto[k,2] }}
}

#Número Total de falhas considerando todos os sistemas
M<-sum(tam)-sum(ciclo)

##########################################################################################

CVM_aux<-matrix(0,M,2)

##########################################################################################
# Definição das funções mi para os 5 possíveis ciclos
##########################################################################################
mi1<-function(t){((t^beta.hat)*((1/lambda.hat)^beta.hat))}

mi2<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
			((1-(S[1]^theta.hat))*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))))}

mi3<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
			((1-(S[1]^theta.hat))*(((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))-((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))))+
			((1-(S[2]^theta.hat))*(1-(S[1]^theta.hat))*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-
((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))))}

mi4<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
			((1-(S[1]^theta.hat))*(((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))-((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))))+
			((1-(S[2]^theta.hat))*(1-(S[1]^theta.hat))*(((T[3]^beta.hat)*((1/lambda.hat)^beta.hat))-
((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))))+
			((1-(S[3]^theta.hat))*(1-(S[2]^theta.hat))*(1-(S[1]^theta.hat))*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-
((T[3]^beta.hat)*((1/lambda.hat)^beta.hat))))}

mi5<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
			((1-(S[1]^theta.hat))*(((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))-((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))))+
			((1-(S[2]^theta.hat))*(1-(S[1]^theta.hat))*(((T[3]^beta.hat)*((1/lambda.hat)^beta.hat))-
((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))))+
			((1-(S[3]^theta.hat))*(1-(S[2]^theta.hat))*(1-(S[1]^theta.hat))*(((T[4]^beta.hat)*((1/lambda.hat)^beta.hat))-
((T[3]^beta.hat)*((1/lambda.hat)^beta.hat))))+
			((1-(S[4]^theta.hat))*(1-(S[3]^theta.hat))*(1-(S[2]^theta.hat))*(1-
(S[1]^theta.hat))*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-((T[4]^beta.hat)*((1/lambda.hat)^beta.hat))))}

##########################################################################################
```

```
linha<-1
linha2<-1

for (sistema in 1:m) {
        dados<-matrix(0,tam[sistema],3)

        # Entrada de dados do sistema
        for (k in 1:tam[sistema]) {
                dados[k,1]<-dados_bruto[linha,1]
                dados[k,2]<-dados_bruto[linha,2]
                dados[k,3]<-dados_bruto[linha,3]
                linha<-linha+1}

        total<-nrow(dados)
        ciclos<-dados[total,2]

        # Cria vetor com os tempos de falha e severidades da MP
        falhas<-matrix(0,(total-ciclos),2)

        # Cria vetor com os tempos de MP e Severidades
        T<-vector(mode="integer",length=ciclos)
        S<-vector(mode="integer",length=ciclos)

        j<-1
        k<-1

        for (i in 1:(total-1)){
                if (dados[i,2]<dados[i+1,2]) {T[k]<-dados[i,1]
                                              S[k]<-dados[i,3]
                                              k<-k+1 }
                else {  falhas[j,1]<-dados[i,1]
                        falhas[j,2]<-dados[i,2]
                        j<-j+1}

                if (i+1==total)         {T[k]<-dados[i+1,1]
                                         S[k]<-dados[i+1,3]}
        }

        if (ciclos==1) {mit<-mi1(T[ciclos])}
        if (ciclos==2) {mit<-mi2(T[ciclos])}
        if (ciclos==3) {mit<-mi3(T[ciclos])}
        if (ciclos==4) {mit<-mi4(T[ciclos])}
        if (ciclos==5) {mit<-mi5(T[ciclos])}

        tot_falhas<-nrow(falhas)

        for (i in 1:tot_falhas){
                if (falhas[i,2]==1) {CVM_aux[linha2,1]<-mi1(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}
                if (falhas[i,2]==2) {CVM_aux[linha2,1]<-mi2(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}
                if (falhas[i,2]==3) {CVM_aux[linha2,1]<-mi3(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}
                if (falhas[i,2]==4) {CVM_aux[linha2,1]<-mi4(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}
                if (falhas[i,2]==5) {CVM_aux[linha2,1]<-mi5(falhas[i,1])
                                     CVM_aux[linha2,2]<-(CVM_aux[linha2,1]/mit)
                                     linha2<-linha2+1}
        # a coluna 1 da matriz CVM_aux tem as estimativas de mi(t)

        }
}
```

```
##################################################################################
# Formato da matriz do Teste Cramer-von Mises (CVM)
# 1a coluna - Ordem da falha
# 2a Coluna - Ri estimado pela Função mi
# 3a coluna - Ri estimado pela Ordem da falha
# 4a coluna - Diferença ao quadrado 2a e 3a colunas
##################################################################################
CVM<-matrix(0,M,4)

ord<-order(CVM_aux[,2])

for (i in 1:M) {
        CVM[i,1]<-i
        CVM[i,2]<-CVM_aux[ord[i],2]
        CVM[i,3]<-(((2*CVM[i,1])-1)/(2*M))
        CVM[i,4]<-(CVM[i,2]-CVM[i,3])^2
}

#Estatística do Teste CVM
CVM2<-(1/(12*M))+sum(CVM[,4])

# H0: Modelo segue NHPP com u(t)
# Se CVM2 < Valor Crítico, então aceita H0.

##################################################################################
# Gráfico para avaliação da Bondade do Ajuste do Modelo
##################################################################################
# Considera o último tempo de cada ciclo como a MP
# Se o tempo da MP for falha, deve replicar esse valor nos dados_bruto
# Essa duplicação deve ser feita apenas para construção deste gráfico
##################################################################################
# Entrada de Dados
dados_bruto<-read.table(file="dados.txt", sep = " ")

# Número de sistemas analisados
m<-max(dados_bruto[,4])

# tam: quantidade de dados para cada sistema
# ciclo: quantidade de ciclos para cada sistema
tam<-vector(mode="integer",length=m)
ciclo<-vector(mode="integer",length=m)

# tempo_sist: ultimo tempo do sistema
tempo_sist<-vector(mode="integer",length=m)

for (k in 1:nrow(dados_bruto)) {
        tam[dados_bruto[k,4]]<-tam[dados_bruto[k,4]]+1
        if ( k==nrow(dados_bruto) ) { ciclo[dados_bruto[k,4]]<-dados_bruto[k,2]
                                      tempo_sist[dados_bruto[k,4]]<-dados_bruto[k,1]}
        else { if (dados_bruto[k,4]<dados_bruto[k+1,4]) {
                ciclo[dados_bruto[k,4]]<-dados_bruto[k,2]
                tempo_sist[dados_bruto[k,4]]<-dados_bruto[k,1]}
        }
}

#Número Total de falhas considerando todos os sistemas
M<-sum(tam)-sum(ciclo)

MCF_aux<-matrix(0,M,m+1)

MCF<-matrix(0,M,4)

info_ciclo<-matrix(0,sum(ciclo),3)
k<-1
linha2<-1
auxMP<-0
```

```
# Retirando os tempos de MP dos ciclos ou do sistema e ficando somente as falhas em MCF_aux
# info_ciclo tem a informação dos tempos finais (coluna 1) de cada ciclo (coluna 2) dos sistemas (coluna 3)
for (linha in 1:sum(tam)) {
        if ( linha==nrow(dados_bruto) ) { info_ciclo[linha2,1]<-dados_bruto[linha,1]
                                          info_ciclo[linha2,2]<-dados_bruto[linha,2]
                                          info_ciclo[linha2,3]<-dados_bruto[linha,4]
        }
        else {      if (dados_bruto[linha,2]<dados_bruto[linha+1,2]) {info_ciclo[linha2,1]<-dados_bruto[linha,1]
                                                                       info_ciclo[linha2,2]<-dados_bruto[linha,2]
                                                                       info_ciclo[linha2,3]<-dados_bruto[linha,4]
                                                                       linha2<-linha2+1
                                                                       auxMP<-1
                    }

                    if (dados_bruto[linha,4]<dados_bruto[linha+1,4]) {info_ciclo[linha2,1]<-dados_bruto[linha,1]
                                                                       info_ciclo[linha2,2]<-dados_bruto[linha,2]
                                                                       info_ciclo[linha2,3]<-dados_bruto[linha,4]
                                                                       linha2<-linha2+1
                                                                       auxMP<-1
                    }

                    if (auxMP==0) {MCF_aux[k,1]<-dados_bruto[linha,1]
                                   k<-k+1
                    }
                    auxMP<-0
          }
}

# Cada coluna da matriz MCF_aux (a partir da 2a) representa um sistema
# Determina em que ciclo esta cada tempo de falha da matriz MCF_aux para os diferentes sistemas
for (i in 1:M) {

        linha<-1

        for (sistema in 1:m) {

                    while (info_ciclo[linha,3]==sistema){
                                if(MCF_aux[i,1]>=info_ciclo[linha,1]) {
                                     MCF_aux[i,sistema+1]<-info_ciclo[linha,2]+1
                                     linha<-linha+1
                                }
                                else { MCF_aux[i,sistema+1]<-info_ciclo[linha,2]
                                     while (info_ciclo[linha,3]==sistema){ linha<-linha+1
                                                                           if(linha==sum(ciclo)+1){
                                                                                                   sistema<-sistema+1
                                                                                                   linha<-1}
                                     }
                                }
                    }
        }
}

##########################################################################################
# Definição das funções mi para os 5 possíveis ciclos
##########################################################################################
mi1<-function(t){((t^beta.hat)*((1/lambda.hat)^beta.hat))}

mi2<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                    ((1-(S[1]^theta.hat))*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))))}

mi3<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                    ((1-(S[1]^theta.hat))*(((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))-((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                    ((1-(S[2]^theta.hat))*(1-(S[1]^theta.hat))*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-
((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))))}
```

```
mi4<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                    ((1-(S[1]^theta.hat))*(((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))-((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                    ((1-(S[2]^theta.hat))*(1-(S[1]^theta.hat))*(((T[3]^beta.hat)*((1/lambda.hat)^beta.hat))-
((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                    ((1-(S[3]^theta.hat))*(1-(S[2]^theta.hat))*(1-(S[1]^theta.hat))*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-
((T[3]^beta.hat)*((1/lambda.hat)^beta.hat)))))}

mi5<-function(t){((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))+
                    ((1-(S[1]^theta.hat))*(((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))-((T[1]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                    ((1-(S[2]^theta.hat))*(1-(S[1]^theta.hat))*(((T[3]^beta.hat)*((1/lambda.hat)^beta.hat))-
((T[2]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                    ((1-(S[3]^theta.hat))*(1-(S[2]^theta.hat))*(1-(S[1]^theta.hat))*(((T[4]^beta.hat)*((1/lambda.hat)^beta.hat))-
((T[3]^beta.hat)*((1/lambda.hat)^beta.hat))))+
                    ((1-(S[4]^theta.hat))*(1-(S[3]^theta.hat))*(1-(S[2]^theta.hat))*(1-
(S[1]^theta.hat))*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-((T[4]^beta.hat)*((1/lambda.hat)^beta.hat))))}

##################################################################################
# Cálculo da estimativa do MCF do processo
for (i in 1:M) {

        MCF[i,1]<-MCF_aux[i,1] #Recebe o tempo de falha
        mit<-0

        linha<-1

        for (sistema in 1:m) {

                dados<-matrix(0,tam[sistema],3)

                # Entrada de dados do sistema
                for (k in 1:tam[sistema]) {
                        dados[k,1]<-dados_bruto[linha,1]
                        dados[k,2]<-dados_bruto[linha,2]
                        dados[k,3]<-dados_bruto[linha,3]
                        linha<-linha+1}

                total<-nrow(dados)
                ciclos<-dados[total,2]

                # Cria vetor com os tempos de falha e severidades da MP
                falhas<-matrix(0,(total-ciclos),2)

                # Cria vetor com os tempos de MP e Severidades
                T<-vector(mode="integer",length=ciclos)
                S<-vector(mode="integer",length=ciclos)

                j<-1
                k<-1

                for (q in 1:(total-1)){
                        if (dados[q,2]<dados[q+1,2]) {T[k]<-dados[q,1]
                                                        S[k]<-dados[q,3]
                                                        k<-k+1 }
                        else {  falhas[j,1]<-dados[q,1]
                                falhas[j,2]<-dados[q,2]
                                j<-j+1}

                        if (q+1==total)         {T[k]<-dados[q+1,1]
                                                 S[k]<-dados[q+1,3]}
                }

                # Verficar em que ciclo esta cada sistema, para utilizar a mi correspondente
                if (MCF_aux[i,sistema+1]==1) {mit<-mit+mi1(MCF[i,1])}
                if (MCF_aux[i,sistema+1]==2) {mit<-mit+mi2(MCF[i,1])}
                if (MCF_aux[i,sistema+1]==3) {mit<-mit+mi3(MCF[i,1])}
                if (MCF_aux[i,sistema+1]==4) {mit<-mit+mi4(MCF[i,1])}
```

```
                    if (MCF_aux[i,sistema+1]==5) {mit<-mit+mi5(MCF[i,1])}
          }
          MCF[i,2]<-mit/m #Estimativa do MCF do processo
}

ord<-order(MCF[,1])
MCF<-MCF[ord,]

# Cálculo da MCF empírica do processo
for (i in 1:M) {

          ri<-0

          for (sistema in 1:m) {
                    if(MCF[i,1]<=tempo_sist[sistema]){ri<-ri+1}

          }

          if(i==1) {MCF[i,3]<-(1/ri)}
          else {MCF[i,3]<-(1/ri)+MCF[i-1,3]}
          MCF[i,4]<-ri

}

# Gráfico MCF estimada X MCF empírica
MCF_estimado<-MCF[,2]
MCF_empirico<-MCF[,3]
plot(MCF_estimado,MCF_empirico)
abline(0,1)

##########################################################################################
# RELATÓRIO FINAL
##########################################################################################

##########################################################################################
# Resumo Verificações
##########################################################################################
# Deve ser zero para indicar que o algoritmo convergiu
convergencia

if(detver1>=0){"Verificar"}
if(detver2<=0){"Verificar"}
if(detver3>=0){"Verificar"}

if(av$values[1]>=0){"Verificar"}
if(av$values[2]>=0){"Verificar"}
if(av$values[3]>=0){"Verificar"}

##########################################################################################
# Estimativas
##########################################################################################
lambda.hat
IC_lambda_Inf
IC_lambda_Sup

beta.hat
IC_beta_Inf
IC_beta_Sup

theta.hat
IC_theta_Inf
IC_theta_Sup
```

```
##########################################################################################
# Teste de Razão de Verossimilhanças
##########################################################################################
# P-valor (H0: S[i]^theta=0)
1-pchisq(TRV, 1)

# Estatística TRV
TRV

# Valor Crítico (VC)
qchisq(0.95,1)

# Se TRV < VC, então aceita H0

##########################################################################################
# Teste Cramer-von Mises
##########################################################################################
#Estatística do Teste CVM
CVM2

# H0: Modelo segue NHPP com u(t)
# Se CVM2 < Valor Crítico, então aceita H0

##########################################################################################
# Otimização Tempo de MP
##########################################################################################
# Custo Manutenção Corretiva
CMC<-1

# custo Manutenção Preventiva
CMP<-5

# Tempo da realização da MP
MP<-T[4]

# Severidade da MP realizada
SMP<-S[4]

# mi=Número de falhas esperadas ultima MP até t
mi<-function(t){((1-(SMP^theta.hat))*(1-(S[3]^theta.hat))*(1-(S[2]^theta.hat))*(1-
(S[1]^theta.hat))*(((t^beta.hat)*((1/lambda.hat)^beta.hat))-((MP^beta.hat)*((1/lambda.hat)^beta.hat))))}

# Função Custo Médio
FCM<-function(t) { ((CMC*mi(t+MP))+(CMP))/t }

# Otimização do Tempo para realização de MP
TO<-optimize(FCM, c(0,500))

TO$minimum+MP

TO$objective

##########################################################################################
# Função Confiabilidade
##########################################################################################

R<-function(t){exp(-mi(MP+t))}

##########################################################################################
```