INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA MECÂNICA




FERRAMENTA AUTOMÁTICA PARA AVALIAÇÃO DA FIABILIDADE E DISPONIBILIDADE DE SISTEMAS REPARÁVEIS

RÚBEN PINTO MARTINS
novembro de 2022

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

FERRAMENTA AUTOMÁTICA PARA AVALIAÇÃO DA FIABILIDADE E DISPONIBILIDADE DE SISTEMAS REPARÁVEIS

Rúben Pinto Martins

2022

Instituto Superior de Engenharia do Porto

Engenharia Mecânica

POLITÉCNICO
DO PORTO

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

FERRAMENTA AUTOMÁTICA PARA AVALIAÇÃO DA FIABILIDADE E DISPONIBILIDADE DE SISTEMAS REPARÁVEIS

Rúben Pinto Martins

1160758

Dissertação apresentada ao Instituto Superior de Engenharia do Porto para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Mecânica, realizada sob a orientação da Doutora Sandra Cristina de Faria Ramos.

2022

Instituto Superior de Engenharia do Porto

Engenharia Mecânica

POLITÉCNICO DO PORTO

POLITÉCNICO
DO PORTO

JÚRI

Presidente
Professora Doutora Rafaela Carla Barros Casais
Professora adjunta do Instituto Superior de Engenharia do Porto - ISEP
Orientador
Professora Doutora Sandra Cristina de Faria Ramos
Professora adjunta do Departamento de Matemática do Instituto Superior de Engenharia do Porto - ISEP
Arguente
Professor Doutor Filipe José Palhares Chaves
Professor adjunto do Instituto Politécnico do Cávado e do Ave - IPCA

AGRADECIMENTOS

Em primeiro lugar gostaria de agradecer à professora Sandra Ramos, que aceitou orientar a minha dissertação de mestrado, revelando desde o início atenção, suporte e incentivo. Os seus concelhos e instruções, assim como a compartilha do seu conhecimento foram vitais para a realização deste trabalho.

Gostaria de agradecer á minha Família por todo o apoio, compreensão e paciência demonstrados ao longo desta etapa. À minha mãe, ao meu pai, à minha irmã, aos meus avôs, aos meus padrinhos, à minha prima, por sonharem e acreditarem nesse projeto junto comigo.

Agradeço a todos os meus amigos e amigas, assim como a todos os professores que contribuíram para a minha formação.

Deixo um agradecimento especial ao meu avô, Manuel Joaquim Pereira Gonçalves, a quem dedico este trabalho.

Por fim, agradeço a Deus pela energia e coragem que me deu para concluir este projeto.

PALAVRAS-CHAVE

Manutenção; Fiabilidade; Disponibilidade; Equipamentos reparáveis; Estatística; Processos Estocásticos; Ferramenta Automática; Python

RESUMO

Nesta dissertação apresentam-se as contribuições de um trabalho de investigação que tem como objetivo o desenvolvimento e implementação de uma ferramenta automática que disponibiliza, quase em tempo real, indicadores de fiabilidade e de disponibilidade para um qualquer sistema reparável.

A ferramenta recebe dados recolhidos no equipamento em estudo, modela o processo gerador das avarias por recurso a vários modelos matemáticos/estatísticos, seleciona o modelo mais adequado aos dados recolhidos e, como base nesse modelo, apresenta um conjunto de indicadores úteis. Estes indicadores podem ser usados, tanto para prever o comportamento do sistema, como para avaliar como é que este responde a ações externas, como, por exemplo, à respetiva manutenção. Os trabalhos são iniciados com uma revisão dos conceitos e instrumentos fundamentais que suportam, grosso modo, o trabalho desenvolvido, seguindo-se o desenho, implementação e validação da ferramenta.

A ferramenta é desenvolvida em ambiente Python e de forma automática. Este automatismo facilita a obtenção de resultados em tempo quase real e permite que o perito do processo tenha acesso a um conjunto de indicadores do estado do sistema sem que tenha conhecimentos específicos sobre modelação de processo geradores de avarias. O resumo dos dados recolhidos, assim como a informação produzida pela ferramenta desenvolvida são apresentados de forma organizada através de um *dashboard*, também desenvolvido em ambiente Python, de forma que a interpretação dos resultados seja mais rápida e eficiente.

KEYWORDS

Maintenance; Reliability; Availability; Repairable Equipments; Statistics; Stochastic Processes; Automatic Tool, Python

ABSTRACT

This dissertation presents the contributions of a research work that aims to develop and implement an automatic tool that provides, almost in real time, reliability, and availability indicators for any repairable system.

The tool receives data collected from the equipment, models the process that generates the faults using various mathematical/statistical models, selects the most appropriate model for the data collected and based on that model, presents a set of useful indicators. These indicators can be used both to predict the behavior of the system and to assess how it responds to external actions, such as, for example, its maintenance. The work begins with a review of the fundamental concepts and instruments that roughly support the work developed, followed by the design, implementation, and validation of the tool.

The tool is developed in a Python environment and automatically. This automatism facilitates obtaining results in near real time and allows the process expert to have access to a set of system status indicators without having specific knowledge about modeling fault-generating processes. The summary of the collected data, as well as the information produced by the developed tool, are presented in an organized way through a dashboard, also developed in a Python environment, so that the interpretation of the results is faster and more efficient.

LISTA DE SÍMBOLOS E ABREVIATURAS

Lista de abreviaturas

CBM	*Condition Based Maintenance*
HPP	*Homoeneous Poisson Process*
ISEP	Instituto Superior de Engenharia do Porto
MTBF	*Mean Time Between Failures*
MTTF	*Mean Time To Failure*
MTTR	Mean Time To Repair
NHPP	*Non Homoeneous Poisson Process*
OEE	*Overall Equipement Effectiveness*
PR	Processo de Renovação
RCM	*Reliability Centered Maintenance*

Lista de Unidades

h	Horas

Lista de Símbolos

%	Percentagem
η	Parâmetro de escala da distribuição de Weibull
B	Parâmetro de forma da distribuição de Weibull
N	Tamanho da amostra
u.t	Unidades de tempo
#	Número da ocorrência

GLOSSÁRIO DE TERMOS

Condition Based Maintenance	Manutenção baseada na condição
Homoeneous Poisson Process	Processo de Poisson homogéneo
Mean Time Between Failures	Tempo médio entre falhas
Mean Time To Failure	Tempo medio até á falha
Mean Time To Repair	Tempo médio para Reparação
Non Homoeneous Poisson Process	Processo de Poisson não homogéneo
Overall Equipement Effectiveness	Eficiência global do equipamento
Reliability Centered Maintenance	Manutenção centrada na fiabilidade

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

ÍNDICE

INTRODUÇÃO

1 INTRODUÇÃO

Este capítulo faz uma introdução ao trabalho desenvolvido através da apresentação de uma contextualização do tema, dos objetivos do trabalho e da metodologia de investigação seguida. É ainda apresentada uma descrição da estrutura do documento e uma breve referência à linguagem de programação usada para implementar a ferramenta proposta.

1.1 Contextualização

É senso comum considerar que uma empresa é mais competitiva do que uma sua concorrente se, de forma sustentável, conseguir obter a preferência parte dos clientes de determinado produto/serviço por estas produzido/oferecido. No ramo industrial, essa competitividade está fortemente correlacionada com a forma como os processos de produção são mantidos. Ora, sabe-se que processos bem controlados promovem maiores níveis de produção com menores taxas de produtos não conformes, contribuindo para uma redução de custos de produção e da pegada ambiental.

A manutenção dos equipamentos/ativos terá uma quota-parte importante no bom desempenho dos processos de produção, pois a sua execução de forma adequada assegura as condições operacionais dos ativos.

A manutenção preditiva e a manutenção com base na condição permitem antever a ocorrência de avarias, poupar na manutenção corretiva e prolongar a vida útil do ativo. Ambos os tipos de manutenção têm como *input* históricos de dados que fornecem a base para encontrar tendências traduzidas por indicadores úteis. Na obtenção dessas tendências, a Indústria 4.0, que engloba um amplo conjunto de métodos avançados como a inteligência artificial, a robótica, a *internet* das coisas e a computação em nuvem atua como importante aliada. A articulação destas quatro tecnologias permite a recolha de dados a partir de instrumentos/equipamentos interligados assim como o seu processamento em tempo real, sendo o resultado deste processamento fulcral na tomada rápida e assertiva de decisão. Decisões assertivas têm um impacto direto na redução de custos e na prevenção de danos ambientais.

1.2 Objetivos

O objetivo principal desta dissertação é o desenvolvimento de uma ferramenta automática, orientada por dados (*data driven*), que disponibilize indicadores de fiabilidade e de disponibilidade para um sistema reparável geral de forma a, não só prever o comportamento do sistema com vista à aplicação de políticas de manutenção preditiva e de manutenção com base na condição, mas também para avaliar como é que este responde a ações externas, como, por exemplo, à respetiva manutenção. Este objetivo geral pode ser traduzido pelos seguintes objetivos específicos:

- ✓ Familiarização com a linguagem Python;
- ✓ Estudo de modelos matemáticos para o processo das avarias;
- ✓ Desenvolvimento conceptual do modelo de falhas/avarias do sistema;
- ✓ Implementação e teste do modelo com dados simulados e reais;
- ✓ Desenvolvimento de um *dashboard.*

1.3 Metodologia

O trabalho de investigação foi desenvolvido no Laboratório de Engenharia Matemática do ISEP (LEMA). O LEMA é um grupo de investigação nas áreas da Matemática Industrial e da Engenharia Matemática. Enquanto grupo multidisciplinar, o LEMA desenvolve atividades nos domínios da modelação, análise numérica, sistemas dinâmicos, investigação operacional e análise de dados, muitas vezes aplicadas à resolução de problemas reais do mundo empresarial. Em particular, todo o processo de classificação dos problemas e a consequente modelação matemática tendo em vista a resolução efetiva dos mesmos é uma mais-valia do LEMA, o que lhe permite ser um dos parceiros das edições Portuguesas dos European Study Groups with Industry.

No que respeita a procedimentos metodológicos, a primeira etapa do trabalho consistiu numa revisão da literatura com vista a uma descrição resumida de conceitos e ferramentas que suportam teoricamente o trabalho desenvolvido. Esta revisão pretendeu ainda perceber o estado da arte do tema estudado. A etapa dois consistiu na identificação de diferentes modelos matemáticos/estatísticos adequados à modelação dos diferentes processos geradores de avarias em sistemas reparáveis. Nesta etapa foram ainda estudados métodos de avaliação da tendência temporal das avarias de equipamentos.
A terceira etapa consistiu na implementação computacional dos vários modelos estudados e a sua articulação de forma automática.
Já a quarta etapa consistiu no teste e validação da ferramenta proposta tanto em dados reais como em dados simulados.

A última etapa passou pelo desenvolvimento de um painel visual (*dashboard)* que apresente de forma organizada e intuitiva os resultados produzidos pela ferramenta, como métricas e indicadores do equipamento.

1.4 Estrutura do relatório

Além desta introdução (Capítulo 1) que enquadra e introduz o tema estudados, esta dissertação é composta por mais quatro capítulos, estruturados da forma seguinte:

Capítulo 2: neste capítulo são apresentados os conceitos e instrumentos fundamentais que suportaram teoricamente todo o trabalho desenvolvido. Não se pretende, de modo algum, uma descrição completa ou exaustiva dos conceitos e instrumentos, mas apenas considerar o detalhe necessário a uma compreensão adequada do trabalho realizado. Apresenta-se ainda um conjunto de referências atualizadas que abrangem a teoria necessária ao desenvolvimento da ferramenta proposta.

Capítulo 3: Este capítulo incluirá a descrição de todos os passos seguidos até se obter um protótipo da ferramenta a desenvolver. São também descritos os aspetos computacionais associados à implementação da ferramenta.

Capítulo 4: É realizada a validação da ferramenta proposta. Esta validação considerará, principalmente, dados sintéticos de forma a aferir a adequabilidade da ferramenta.

Capítulo 5: Este capítulo encerra o documento com um resumo das contribuições mais relevantes do trabalho desenvolvido. É ainda realizada uma análise crítica da ferramenta desenvolvida e discutidas algumas possibilidades de trabalho futuro.

1.5 Ferramenta computacional

O trabalho computacional necessário neste trabalho foi desenvolvido com a linguagem de programação de alto nível Python. Esta linguagem foi concebida na Holanda, no final da década de 80, por Guido van Rossum, a partir de uma linguagem existente na época designada de ABC.
Segundo Menezes (2010) o Python é uma linguagem completa, que conta com bibliotecas para aceder a conjuntos de dados armazenados em diferentes formatos, para processar ficheiros XML, para construir interfaces gráficas e mesmo jogos, entre outras funcionalidades. Nesta linguagem é possível recorrer a um conjunto alargado de funções já existentes, evitando-se assim a escrita de muitas linhas de código. Esta facilidade aumenta a produtividade e a eficiência do programador, pois as funções

disponibilizadas nas bibliotecas do Python são testadas por diferentes utilizadores e, portanto, são, quase sempre, isentas de erros.

O Python é uma linguagem livre, ou seja, pode ser utilizado gratuitamente, e está disponível para vários sistemas operacionais tais como Linux, FreeBSD, Windows ou MacOSX. O acesso a esta linguagem e às suas bibliotecas opcionais podem ser consultadas em https://www.python.org/.

REVISÃO BIBLIOGRÁFICA

2.1 Fiabilidade de sistemas não reparáveis

2.2 Fiabilidade de sistemas reparáveis

2.3 Simulação

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo são apresentados os conceitos fundamentais que suportam, grosso modo, o trabalho desenvolvido. Na secção seguinte faz-se uma caracterização de sistemas não reparáveis, descrevendo-se os conceitos gerais associados a estes sistemas, a função de fiabilidade e a taxa instantânea de falha, e as distribuições mais utilizadas em fiabilidade. Segue-se, uma revisão de conceitos fundamentais associados com sistemas reparáveis. É, ainda, feita uma revisão referente à simulação de distribuições e processos e a técnicas de seleção de modelos. Termina-se o capítulo com uma breve revisão sobre os conceitos relacionados com manutenção.

2.1 Sistemas não reparáveis

Um sistema define-se como não reparável se for constituído por um ou vários itens que por razões económicas ou tecnológicas não carecem de reparação, a avaria significa o fim da sua vida (Santos & Brito, 2003).

2.1.1 Alguns conceitos fundamentais

Antes de se iniciar a descrição de distribuições de probabilidade úteis na modelação do tempo até à falha de componentes, introduzem-se alguns conceitos fundamentais relativamente a sistemas não reparáveis (também designados de componentes ou itens).

Fiabilidade [NP EN 13306 (2007)]

"Aptidão de um bem para cumprir uma função requerida sob determinadas condições, durante um dado intervalo de tempo."

A fiabilidade pode definir-se ainda como o grau de confiança atribuído ao funcionamento, sem falhas, por parte de um sistema, em certas condições (ambientais, carga, etc.) e durante um período de tempo superior a t unidades (Ramos, 2021 e referências incluídas).

Segundo Assis (1997), a fiabilidade resulta quer da conceção e da qualidade de fabrico do sistema (características intrínsecas), quer das condições de carga e ambientais em que decorrerá a sua operação (características extrínsecas).

Avaria [NP EN 13306 (2007)]

"Cessação da aptidão de um bem para cumprir uma função requerida."

Os termos 'avaria' e 'falha' são praticamente sinónimos e podem usar-se indistintamente. É, contudo, comum usar-se o termo "avaria" com mais amplitude para um equipamento como um todo e o termo "falha" num sentido mais restrito, dirigido ao elemento/componente (Ramos, 2021 e referências incluídas).

Tempo medio até à falha[1]. É um indicador importante para a definição da fiabilidade de um bem. Este é utilizado em casos de componentes não reparáveis que são substituídos por componentes novos à medida que vão falhando:

$$MTTF = E(T) = \int_0^{\infty} t\, f(t)dt, \qquad \textbf{(1)}$$

onde T representa a v.a. "Tempo até a falha de um determinado item (em unidades de tempo)" e $f(t)$ é a função densidade de probabilidade dessa variável.

Tempo mediano até à falha. Este indicador de fiabilidade divide a distribuição dos tempos até à falha em duas metades, sendo que probabilidade do item falhar antes do tempo mediano ($t_{0,5}$) inferior ou igual a 0,5. Matematicamente traduz-se por:

$$\mathrm{t_{0,5}}\colon \mathrm{P(T} \leq 0{,}5) = 0{,}5. \qquad \textbf{(2)}$$

Moda. A moda da distribuição é o tempo de falha mais provável, ou seja, é onde a função densidade de probabilidade, f(t), atinge o seu máximo:

$$f(t_{moda}) = \max\, f(t)\,, 0 \leq t \leq \infty. \qquad \textbf{(3)}$$

Variância do tempo até à falha. Este indicador fornece informação sobre a dispersão dos tempos relativamente ao tempo médio e é obtido pela fórmula:

$$V(T) = E(T^2) - [E(T)]^2 = \int_0^{+\infty} t^2\, f(t)dt - \left[\int_0^{+\infty} tf(t)dt\right]^2. \qquad \textbf{(4)}$$

2.1.2 Função de fiabilidade

A fiabilidade pode definir-se matematicamente como a probabilidade do elemento funcionar, nas condições fixas estabelecidas, por um período de tempo definido. A sua expressão matemática é traduzida por:

[1] Mean Time to Failure (MTTF), na literatura em Inglês.

$$R(t) = P(T > t) = \int_{t}^{\infty} f(u)du \quad \textbf{(5)}$$

A probabilidade da falha acontecer nesse mesmo período é expressa pela função $F(.)$, função de infiabilidade, pelo que, para um dado t:

$$F(t) = P(T \leq t) = 1 - R(t) \quad \textbf{(6)}$$

A Figura 1 representa como as funções referidas acima estão relacionadas para o caso em que as falhas ocorrem de forma aleatória no tempo (taxa de falhas constante). A linha a cor vermelha representa a função de infiabilidade que tende para um à medida que o tempo aumenta e a linha a cor negra representa a função de fiabilidade que tende para zero com o avançar do tempo.

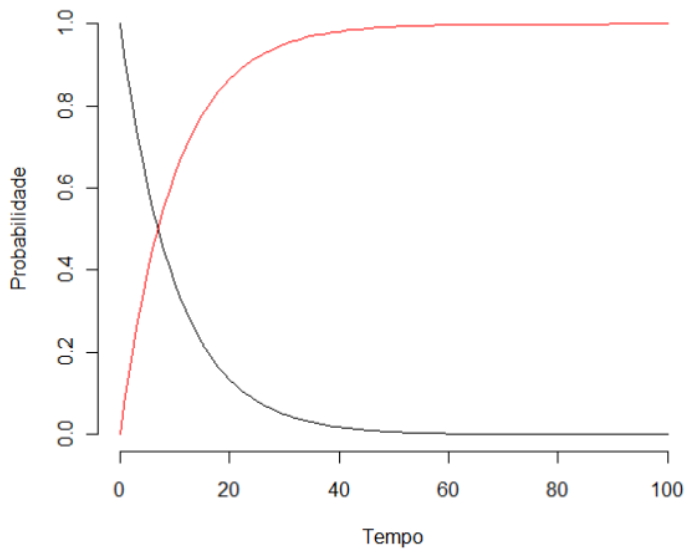

Figura 1 - Função Fiabilidade e Função Infiabilidade [(Ramos, 2021)]

Função de risco. A função de risco é a probabilidade de um componente/item vir a falhar no intervalo de tempo $[t, t + \Delta t]$ sabendo que no instante t está operacional. Matematicamente é representada por

$$\lambda(t) = \frac{f(t)}{R(T)}, t > 0. \quad \textbf{(7)}$$

$\lambda(t)$ descreve a forma como a taxa instantânea de falha muda com o tempo, sendo que pode, no caso de monotonia, ser de 3 tipos:

- crescente: o risco de falha do componente aumenta com o tempo.
- decrescente: o risco de falha do componente diminui com o tempo.
- constante: as falhas ocorrem de forma aleatória no tempo.

Alguns componentes apresentam uma função de risco com comportamento temporal traduzido por uma curva designada de curva de banheira (Figura 2).

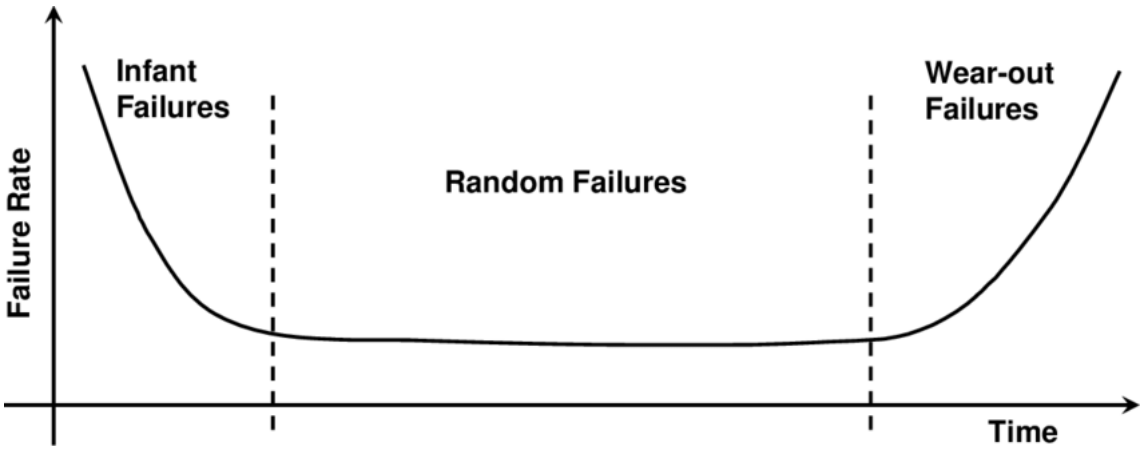

Figura 2 - Curva da banheira [(van der Poel & van Waveren, 2012)]

As falhas exibidas na primeira parte da curva, onde a taxa de falha diminui, são chamadas de falhas precoces ou falhas de mortalidade infantil. O meio da curva é referido como o período de vida útil, e presume-se que as falhas exibem uma taxa constante, ou seja, ocorrem de forma aleatória no tempo. A última parte da curva descreve o período de desgaste, e assume-se que a taxa de falha aumenta à medida que os mecanismos de desgaste aceleram (Smith, 2017).

Compreender bem a curva da banheira permite, em primeiro lugar, determinar a vida útil do componente, a sua fiabilidade e mantê-lo controlado. Mas a principal vantagem é sem dúvida conseguir planear melhor como e quando fazer manutenção.

2.1.3 Distribuições de uso frequente em fiabilidade

Segundo (Barabadi, 2013) para se fazer uma análise de fiabilidade eficaz do sistema é necessário ter uma estimativa precisa da distribuição de falhas para os diferentes componentes que constituem o sistema. Quando os dados de falha dos componentes estão disponíveis e exibem um alto grau de variabilidade ou aleatoriedade, funções de distribuição de probabilidade (por exemplo, exponencial, lognormal, distribuição Weibull, entre outras) podem ser usadas para análise de fiabilidade.

De seguida, apresenta-se uma breve descrição das distribuições de probabilidade de uso frequente na modelação do tempo até à ocorrência de falha de componentes.

Distribuição Exponencial

A análise de fiabilidade de componente com recurso à distribuição Exponencial é muito popular devido ao facto de ser o modelo adequado para situações em que as avarias ocorrem de forma aleatória com o tempo. O modelo exponencial é simples e

fácil de implementar e tem sido demonstrado na literatura que fornece boas aproximações para distribuições do tempo de falha (Das, 2008). Uma variável aleatória T contínua segue uma distribuição exponencial com parâmetro $\lambda\ (\lambda > 0)$ se a sua função densidade de probabilidade for dada por:

$$f(t) = \lambda e^{-\lambda t}\ , t > 0 \quad \textbf{(8)}$$

A função fiabilidade é então obtida por:

$$R(t) = P(T > t) = \int_{t}^{\infty} f(u)du = e^{-\lambda t}, t > 0, \quad \textbf{(9)}$$

e a função de infiabilidade, $F(t)$, é expressa por:

$$F(t) = P(T < t) = 1 - R(t) = 1 - e^{-\lambda t}, t > 0 \quad \textbf{(10)}$$

O tempo médio até á falha[2] (MTTF) e a variância de uma variável exponencial são expressas, respetivamente, por:

$$MTTF = E(T) = \int_{0}^{\infty} R(t)dt = \int_{0}^{\infty} e^{-\lambda t}\, dt = \frac{1}{\lambda} \quad e \quad V(T) = \frac{1}{\lambda^2}. \quad \textbf{(11)}$$

Note-se que tanto o valor médio como a variância de uma variável aleatória com distribuição exponencial são funções do parâmetro dessa distribuição.

Por fim, a função de risco expressa-se por:

$$\lambda(t) = \frac{f(t)}{R(t)} = \frac{\lambda e^{-\lambda t}}{e^{-\lambda t}} = \lambda. \quad \textbf{(12)}$$

O resultado anterior permite concluir que a distribuição exponencial é adequada para modelar situações em que as avarias ocorrem de forma aleatória no tempo já que a sua função de risco é constante, ou seja, não depende do tempo.

Distribuição de Weibull

A distribuição Weibull tem o nome do físico sueco Waloddi Weibull, que a usou em 1939 para representar a distribuição da resistência à rutura de materiais e em 1951 para uma variedade de outras aplicações. É, talvez, o modelo de vida útil mais frequentemente usado na literatura de fiabilidade (Nafidi et al., 2019). Segundo Kang et al (2021) a distribuição Weibull encontra aplicações nas áreas da meteorologia, economia, administração de empresas, saúde, física, ciências sociais, hidrologia,

[2] Mean time to failure, na literatura em Inglês.

biologia e engenharia. Esta é ainda uma distribuição versátil já que a sua função densidade de probabilidade pode apresentar diversas formas. Dependendo do valor do parâmetro de forma, β, a sua função de taxa de falha pode ser decrescente, constante ou crescente. Como tal, pode ser usada para modelar o comportamento de falha de vários sistemas da vida real (Ben-Daya et al., 2009).

A função de densidade de probabilidade da distribuição Weibull com três parâmetros é dada por:

$$f(t) = \frac{\beta(t-\gamma)^{\beta-1}}{\eta^{\beta}} exp\left[-\left(\frac{t-\gamma}{\eta}\right)^{\beta}\right], t \geq \gamma, \quad \textbf{(13)}$$

onde $\gamma \in \mathbb{R}$ é o parâmetro de posição ou vida garantida, $\eta > 0$ é o parâmetro de escala ou vida característica, $\beta > 0$ é o parâmetro de forma.

Quanto o parâmetro γ assume o valor zero, ou seja, as falhas podem ocorrer desde a origem dos tempos, obtém-se a distribuição de Weibull com dois parâmetros. Neste caso, a função densidade de probabilidade é dada por:

$$f(t) = \frac{\beta}{\eta}\left(\frac{t}{\eta}\right)^{\beta-1} exp\left[-\left(\frac{t}{\eta}\right)^{\beta}\right], t > 0. \quad \textbf{(14)}$$

A Figura 3 mostra o gráfico da função densidade de probabilidade de uma variável de Weibull biparamétrica para diferentes valores de parâmetro de forma (β) e parâmetro de escala (η) constante e de valor igual a 10.

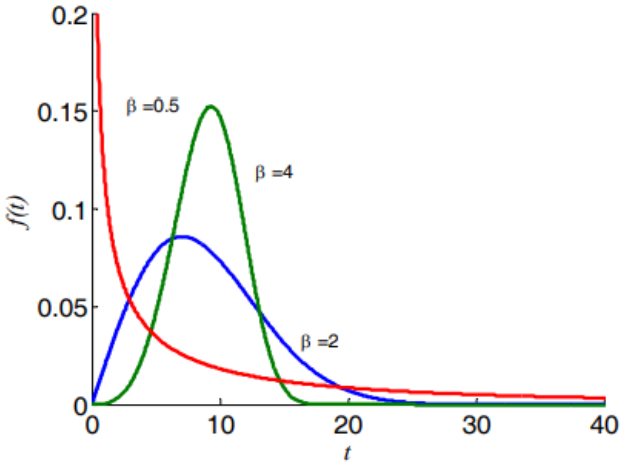

Figura 3 - Gráfico pdf da distribuição de Weibull [(Ben-Daya et al., 2009)]

No caso biparamétrico, as funções de fiabilidade e de risco são dadas, respetivamente, por:

$$R(t) = P(T > t) = exp\left[-\left(\frac{t}{\eta}\right)^{\beta}\right] e \quad \textbf{(15)}$$

$$\lambda(t) = \frac{\beta}{\eta}\left(\frac{t}{\eta}\right)^{\beta-1}, t > 0. \quad \textbf{(16)}$$

Na Figura 4 estão representadas várias funções de risco para diferentes parâmetros de forma sendo que o parâmetro de escala é constante e assume o valor de 10.

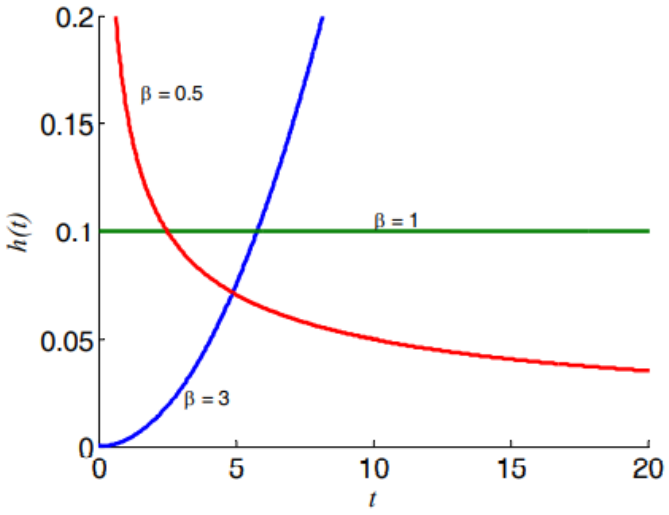

Figura 4 - Gráfico da função de risco de Weibull [(Ben-Daya et al., 2009)]

Segundo Ramos (2021 e referências aí incluídas), quando $\beta = 1$, as taxas de falha são constantes ao longo do tempo indicando assim falhas aleatórias. Enquanto se $\beta < 1$ indica mortalidade infantil, ou seja, que as taxas de falha são altas e decrescentes no início da vida do produto ou sistema. Por sua vez, se $1 < \beta < 4$: indica um desgaste antecipado, tornando-se um problema caso as falhas ocorram dentro da vida projetada para o sistema ou produto. Por fim, se $\beta > 4$ indica desgaste de fim de vida.

O tempo médio e mediano até á falha e obtém-se pelas expressões seguintes:

$$MTTF = E(T) = \eta \times \Gamma\left(1 + \frac{1}{\beta}\right) \quad \textbf{(17)}$$

$$t_{0,5} = \eta[-ln(0,5)]^{\frac{1}{\beta}}, \quad \textbf{(18)}$$

onde Γ(.) representa a função gama que é data por por $\Gamma(t) = \int_0^{+\infty} x^{t-1}e^{-x}\,dx.$ **(19)**

Distribuição lognormal

A distribuição lognormal é um modelo muito flexível que se pode ajustar a muitos tipos de dados de falha. Esta distribuição tem aplicações em engenharia de

manutenção, sendo capaz de modelar a incerteza dos tempos até à falha (Pham, 2005). A função densidade probabilidade de uma variável aleatória com distribuição lognormal pode ser expressa por,

$$f(t) = \frac{1}{\sigma t\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{ln\, t-\mu}{\sigma}\right)^2}, t > 0, \quad \textbf{(20)}$$

onde μ e σ são os parâmetros da distribuição com $\sigma > 0$. Uma representação gráfica da função densidade probabilidade de uma variável aleatória com distribuição lognormal está representado na Figura 5.

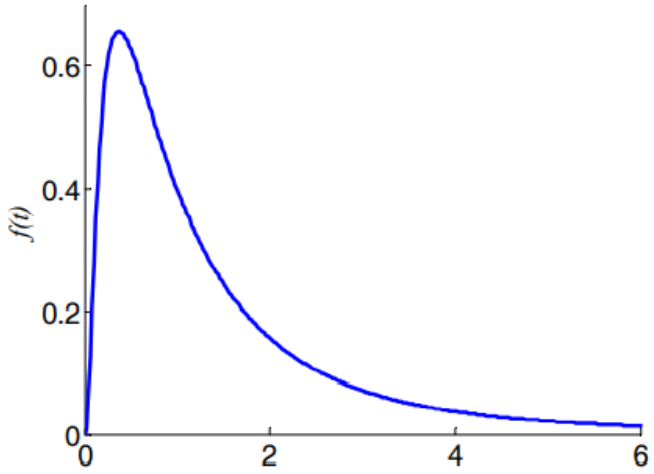

Figura 5 - Gráfico função densidade de probabilidade da distribuição lognormal para μ=0 e σ=1 [(Ben-Daya et al., 2009)]

A média e a variância desta distribuição expressam-se, respetivamente, por:

$$E(T) = MTTF = e^{\mu+\frac{\sigma^2}{2}} \quad \text{e} \quad Var(T) = e^{2\mu+\sigma^2}(e^{\sigma^2} - 1). \quad \textbf{(21) e (22)}$$

Por fim, as expressões de fiabilidade e de risco são:

$$R(t) = P(T > t) = 1 - \Phi\left(\frac{ln\, t - \mu}{\sigma}\right) e \quad \textbf{(23)}$$

$$\lambda(t) = \frac{\varphi\left(\frac{ln\, t - \mu}{\sigma}\right)}{t\sigma\left[1 - \Phi\left(\frac{ln\, t - \mu}{\sigma}\right)\right]} \quad \textbf{(24)}$$

onde φ e Φ são, respetivamente, a função densidade de probabilidade e a função de distribuição de uma variável aleatória com distribuição normal.

Distribuição Gamma

Segundo Pham (2005) a distribuição gamma pode ser usada como uma função de probabilidade de falha para componentes cuja distribuição é assimétrica. Este é um

modelo flexível que pode oferecer um bom ajuste para alguns conjuntos de dados de falha, não sendo, no entanto, amplamente utilizado como uma distribuição modelo para mecanismos de falha comuns. Um uso comum deste modelo ocorre em aplicações de fiabilidade bayesiana. A distribuição Gamma, na sua forma mais simples, é caracterizada por dois parâmetros: o parâmetro de forma (β) e o parâmetro de escala (η). A sua função densidade probabilidade é dada por:

$$f(t) = \frac{\eta^{\beta}}{\Gamma(\beta)}(t)^{\beta-1}e^{-\eta t}, t > 0, \quad \textbf{(25)}$$

onde $\eta > 0$ e β>0. $\Gamma(.)$ representa a função gama já introduzida anteriormente.

Note-se que esta função tem um comportamento muito similar à distribuição de Weibull no que respeita à variação da taxa de falha em função da alteração do parâmetro de forma, já que:

- Quando $0 < \beta < 1$, a taxa de falha diminui;
- Quando $\beta > 1$, a taxa de falha aumenta;
- Quando $\beta = 1$ a taxa de falha é constante.

Note-se ainda que quando $\beta = 1$ tem-se a distribuição exponencial.

O tempo medio até à falha e a variância são expressos, respetivamente, por:

$$E(T) = \frac{\beta}{\eta} \quad \text{e} \quad V(T) = \frac{\beta}{\eta^2}. \quad \textbf{(26) e (27)}$$

Por fim, as funções de fiabilidade e de risco são obtidas por:

$$R(t) = \sum_{n=0}^{\beta-1} \frac{(\eta t)^n}{n!} e^{-\eta t}, t > 0 \quad \textbf{(28)}$$

$$\lambda(t) = \frac{\frac{\eta}{\Gamma(\beta)}(\eta t)^{\beta-1}e^{-\eta t}}{\sum_{n=0}^{\beta-1}\frac{(\eta t)^n}{n!}e^{-\eta t}}, t > 0. \quad \textbf{(29)}$$

Distribuição Normal

A distribuição normal é uma distribuição contínua que tem amplas aplicações. A função densidade de probabilidade tem a conhecida forma de sino e é simétrica em relação ao eixo que passa pelo valor médio da variável normal (Ben-Daya et al., 2009) e expressa-se, matematicamente, por:

$$f(t) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{t-\mu}{\sigma}\right)^2}, t \in \mathbb{R}, \quad \textbf{(30)}$$

onde $\mu \in \mathbb{R}$ (valor médio) e $\sigma \in \mathbb{R}^+$ (desvio padrão) representam os parâmetros da distribuição.

A título de ilustração, o gráfico da função densidade probabilidade da distribuição normal com valor médio 10 e desvio padrão 2 encontra-se representado abaixo (Figura 6).

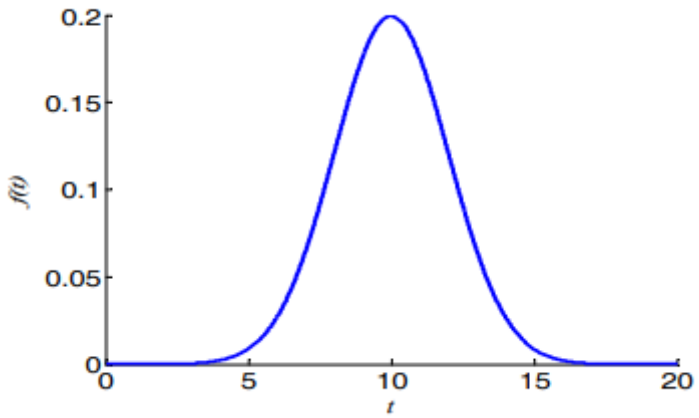

Figura 6 - Distribuição normal com μ=10 e σ=2 [(Ben-Daya et al., 2009)]

A distribuição normal com parâmetros μ=0 e $\sigma^2 = 1$ é designada de distribuição normal padrão e a sua densidade probabilidade expressa-se por:

$$f(t) = \frac{1}{\sqrt{2\pi}} e^{-t^2/2}, t \in \mathbb{R}. \quad \textbf{(31)}$$

As funções de infiabilidade e fiabilidade[3], por sua vez, são dadas por, respetivamente:

$$F(t) = P(T \leq t) = \Phi\left(\frac{t-\mu}{\sigma}\right) \text{ e } R(t) = 1 - \Phi\left(\frac{t-\mu}{\sigma}\right), \quad \textbf{(32) e (33)}$$

onde $\Phi(.)$ representa a função distribuição de uma variável aleatória com distribuição normal padrão.

Distribuição Beta

Segundo Pham (2005), a distribuição beta tem muitas aplicações na engenharia de fiabilidade, sendo, portanto, utilizada de forma frequente, apesar do seu suporte ser o intervalo $[0,1]$.

A distribuição beta é parametrizada por dois parâmetros positivos, α e β, que controlam a forma da distribuição e sua função densidade probabilidade pode exprime-se por,

[3] Neste contexto, $t > 0$.

$$f(t,\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)+\Gamma(\beta)} t^{\alpha-1}(1-t)^{\beta-1}, 0 \leq t \leq 1, \quad \textbf{(34)}$$

onde $\Gamma(.)$ é a função gama e $\alpha, \beta > 0$ os parâmetros de forma.

O valor médio e a variância desta função são dados, respetivamente por:

$$E(T) = \frac{\alpha}{\alpha+\beta} \; e \quad \textbf{(35)}$$

$$Var(T) = \frac{\alpha\beta}{(\alpha+\beta)^2\,(\alpha+\beta+1)}. \quad \textbf{(36)}$$

As duas figuras abaixo exemplificam várias formas que os gráficos da função densidade probabilidade de distribuições beta podem tomar em função dos valores dos seus parâmetros. A Figura 7 ilustra distribuições Beta assimétricas, enquanto que a Figura 8 é relativa a distribuições beta simétricas que são obtidas quando os parâmetros de forma assumem os mesmos valores.

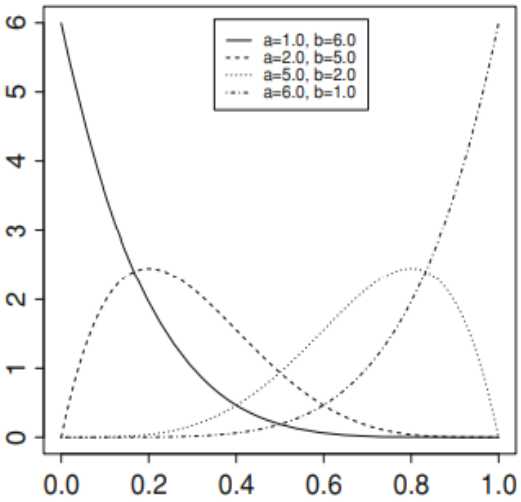

Figura 7 - Distribuições Beta assimétricas [(Gámiz & Lindqvist, 2016)]

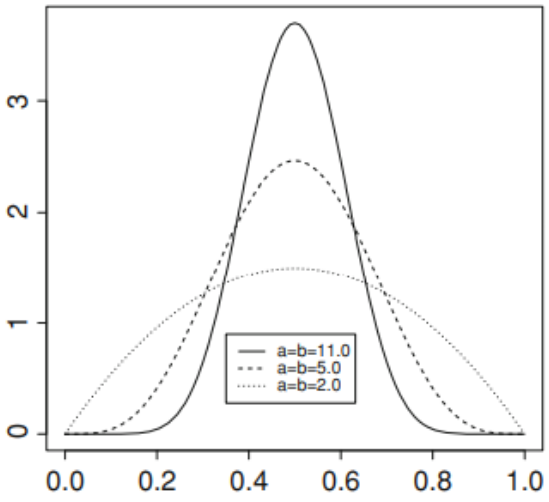

Figura 8 - Distribuições Beta simétricas [(Gámiz & Lindqvist, 2016)]

2.2 Fiabilidade de sistemas reparáveis

Um sistema diz-se reparável se após deixar de realizar uma ou mais de suas funções de forma satisfatória, poder ser restaurado para um desempenho totalmente satisfatório por qualquer método que não seja a substituição de todo o sistema (Lindqvist, 1999).

2.2.1 Alguns conceitos fundamentais

Antes de se iniciar a descrição dos modelos probabilísticos com aplicação na modelação de processos de geração de avarias em sistemas reparáveis, introduzem-se alguns conceitos fundamentais.

Manutenção [NP EN 13306 (2007)]

"Combinação de todas as ações técnicas, administrativas e de gestão, durante um ciclo de vida de um bem, destinadas a mantê-lo ou a repô-lo num estado em que possa desempenhar a função requerida."

Gestão de manutenção [EN 13306(2017)]

"Todas as atividades de gestão que determinam os objetivos, a estratégia e as responsabilidades respeitantes à manutenção e que os implementam por diversos meios tais como o planeamento, o controlo e supervisão da manutenção e a melhoria de métodos na organização, incluindo os aspetos económicos."

Downtime

O tempo de inatividade ou interrupção. O *Downtime* é o período durante o qual o equipamento está em estado de avaria (Smith, 2017).

Up time

O tempo de atividade (*Up time*) descreve por quanto tempo um sistema está operacional.

Tempo médio de reparação

O tempo médio de reparação[4] representa o valor que se espera observar para o tempo de reparação (T) e é dado por,

$$MTTR = E(T) = \int_0^\infty t\, g(t)dt, \qquad \textbf{(37)}$$

onde $g(t)$ representa a função densidade de probabilidade da variável aleatória que representa o tempo de reparação aqui representada por T.

[4] Mean time to repair, na literatura em Inglês.

Tempo médio entre avarias

Este é um importante indicador de fiabilidade utilizado em sistemas reparáveis, que representa o valor que se espera observar para o tempo decorrido entre duas avarias[5] de um determinado equipamento (Pham, 2005).

Disponibilidade [NP EN 13306 (2007)]

"*Aptidão de um bem para cumprir uma função requerida sob determinadas condições num dado instante, ou durante um intervalo de tempo, assumindo que é assegurado o fornecimento dos recursos externos necessários*."

Segundo Pham (2005) a disponibilidade de um sistema é definida, matematicamente, como a probabilidade de o sistema ser bem-sucedido no intervalo de tempo.

Disponibilidade instantânea

Na prática, a disponibilidade instantânea (D) é igual à razão entre o tempo útil (t_u) (tempo de bom funcionamento) e o tempo útil somado do tempo de paragem (t_p) (Ben-Daya et al., 2009):

$$D = \frac{t_u}{t_u + t_p} \quad \textbf{(38)}$$

Disponibilidade média

A disponibilidade média é a proporção de tempo para o qual o sistema está disponível para uso durante uma missão ou período de tempo. Representa o valor médio da disponibilidade instantânea função ao longo do período $(0, T]$ (Katukoori, 1995).

Sem perda de generalidade, para ilustrar o cálculo da disponibilidade média considerou-se o diagrama da Figura 9 onde os X_i representam os tempos de bom funcionamento e os D_i os tempos de paragem.

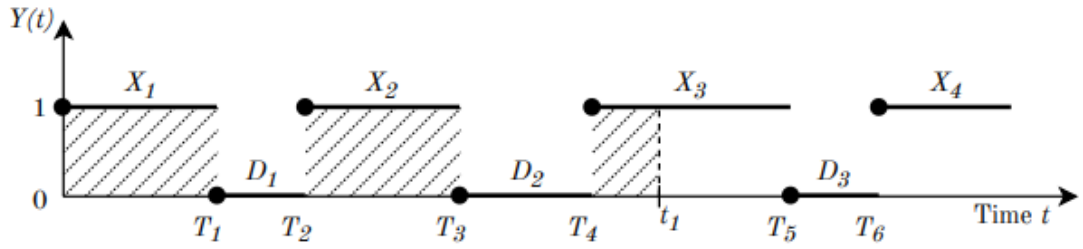

Figura 9 - Diagrama exemplificativo da disponibilidade média

[5] Mean time between failures, na literatura em Inglês.

A expressão de cálculo da disponibilidade média para o exemplo da figura anterior é:

$$\text{Disp. méd}(0,t_1) = \frac{1}{t_1-0} Area_1(0,t_1) = \frac{1}{t_1}\sum_{i=1}^{4}(T_i - T_{i-1})Y(T_{i-1}) + (t_1 - T_4)Y(T_4). \quad \textbf{(39)}$$

A generalização desta fórmula é um processo trivial.

2.2.2 Processo de Poisson Homogéneo

De acordo com Lindqvist (1999), os modelos mais usados para modelar os processos de avaria de um sistema reparável são os processos de renovação (RP), os processos de Poisson homogêneos (HPP) e os processos de Poisson não homogêneos (NHPP).

A teoria da renovação teve a sua origem no estudo de estratégias de substituição de componentes técnicos, mas mais tarde foi desenvolvida como uma teoria geral dentro de processos estocásticos. Como o nome do processo indica, é usado para modelar renovações ou substituições de equipamentos (Rausand & Hoyland, 2004).

Segundo BahooToroody et al. (2020), o modelo de reparação perfeita (também conhecido como processo de renovação) representa um modelo em que os tempos entre as avarias sucessivas de um determinado sistema são variáveis aleatórias distribuídas de forma idêntica.

Neste tipo de sistema de modelação de avaria, considera-se que o sistema fica como novo após a falha, tal como se pode verificar na Figura 10.

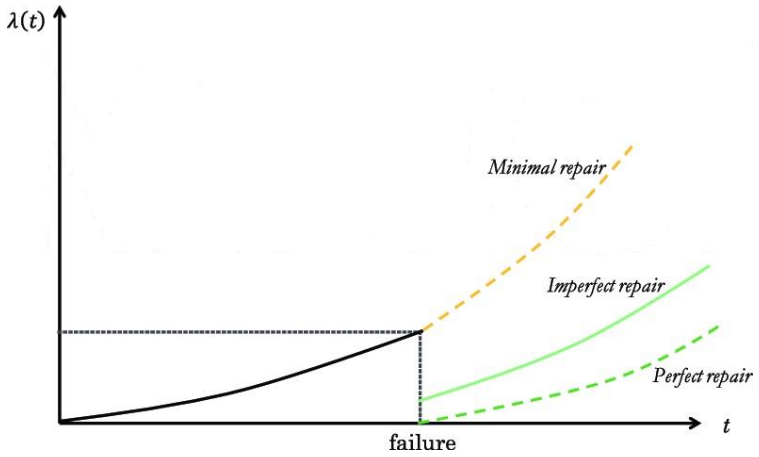

Figura 10 - Gráfico da taxa de risco para reparação perfeita, imperfeita e mínima [(Carlo & Arleo, 2017)]

O nome do processo de Poisson surgiu em homenagem ao matemático francês Siméon Denis Poisson. No processo homogéneo de Poisson (HPP), os tempos entre ocorrências (avarias) sucessivas são exponencialmente distribuídos com o mesmo parâmetro com o mesmo parâmetro λ (taxa de falha).

Um processo de contagem $\{N(t), t \geq 0\}$ é dito ser um HPP com taxa λ $(\lambda > 0)$ se (Rausand & Hoyland, 2004),

1. $N(0) = 0$ (no instante inicial não ocorrem avarias)
2. $N(t)$ tem incrementos independentes
3. O número de ocorrências em qualquer intervalo de tempo de comprimentos t tem distribuição de Poisson com média λt. Ou seja, para todos os $s, t \geq 0$,

$$P(N(t+s) - N(s) = n) = \frac{(\lambda t)^n}{n!} e^{-\lambda t}, n = 0,1,2, \ldots. \quad \textbf{(40)}$$

Como consequência do ponto 3 da definição anterior,

$$E\big(N(t)\big) = Var\big(N(t)\big) = \lambda t. \quad \textbf{(41)}$$

Em suma o processo homogéneo de Poisson pode ser usado para modelar processos geradores de avaria com taxa de ocorrência constante ao longo do tempo.

2.2.3 Processo de Poisson não homogéneo

Um processo de contagem $\{N(t), t \geq 0\}$ é dito ser um Poisson não homogéneo (NHPP) com função intensidade $\lambda(t)$, para $t \geq 0$ se respeitar quatro condições (Rausand & Hoyland, 2004),

1. $N(0) = 0$
2. $\{N(t), t \geq 0\}$ tem incrementos independentes
3. $\mathrm{P}(N(t + \Delta t) - N(t) \geq 2 = o(\Delta t)$, o que significa que o sistema não tem mais de uma falha ao mesmo tempo
4. $\mathrm{P}(N(t + \Delta t) - N(t) = 1 = \lambda(t)\Delta t + o(\Delta t)$

O parâmetro fulcral do NHPP é a sua função intensidade λ(t). A Figura 11 representa uma função intensidade onde os tempos de falha ocorrem em $S1, S2 \ldots$:

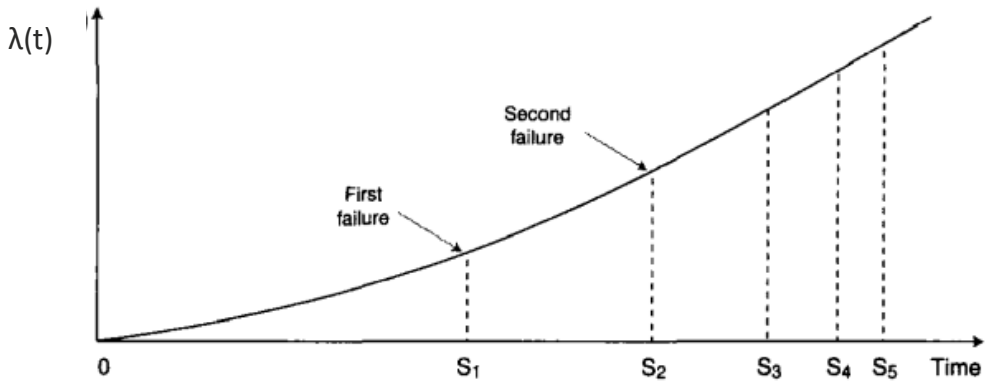

Figura 11 - Função intensidade NHPP [(Rausand & Hoyland, 2004)]

Um modelo NHPP modela situações que se enquadram no grupo de reparação mínima, o que significa que após a reparação o sistema encontra-se em situação quase igual à que estava imediatamente antes da falha (Lindqvist, 1999).

Apesar dos modelos mais comuns para o processo de avaria de um sistema reparável serem os processos de Poisson não homogêneos (NHPP), usados em situações de reparações mínimas, e os processos de renovação (RP) onde se enquadram os Processos de Poisson Homogéneos (HPP), relacionados com situações de reparações ou substituições perfeitas, para algumas aplicações é necessário recorrer a modelos mais gerais e que se enquadram no grupo dos sistemas de reparação imperfeita (Gámiz & Lindqvist, 2016) e sobre os quais se faz uma muito breve referência de seguida pois não serão usados no âmbito deste trabalho.

2.2.4 Processos de renovação imperfeita

Nas últimas décadas foi dada especial aos modelos para situações de reparação imperfeita (Finkelstein, 2015). A reparação imperfeita enquadra-se na reparação que transforma um sistema avariado numa condição intermediária entre "tão bom quanto novo" e "tão ruim quanto velho" (Yang et al., 2016). Ou seja, além de um estado "tão bom quanto novo" e "tão ruim quanto velho", os sistemas reparáveis também podem chegar a estados "melhores do que os antigos, mas piores do que os novos", "melhores do que os novos" ou "piores do que os antigos". Os modelos para situações de reparação imperfeita tentam incorporar esses estados. Os modelos matemáticos envolvidos para acomodar esses estados são muito mais complicados, mas adequados para condições operacionais reais e podem produzir resultados melhores do que as modelações de tipo mínimo ou perfeito (Tanwar et al., 2014).

Entre os modelos para situações de reparação imperfeita, encontram-se os processos mistos, os modelos virtuais de redução de idade, os processos geométricos, os modelos de redução de risco, os modelos de componentes virtuais e generalizações destes (Nafisah et al., 2019).

Um modelo de manutenção imperfeita amplamente utilizado é o modelo Brown-Proschan (BP). Este assume que no momento de cada avaria ocorre uma reparação perfeita com probabilidade p e uma reparação mínima ocorre com probabilidade $1 - p$, independentemente do histórico de avarias anterior (Gjmiz et al., 2011). O modelo é relevante, por exemplo, em situações onde algumas das avarias menores de um sistema complexo são minimamente reparadas, enquanto outras avarias mais sérias resultam em substituição do sistema com a avaria (X. Liu et al., 2020).

Os modelos de idade virtual são os modelos de reparação imperfeita usados com mais frequência. O princípio é que o desgaste não depende da idade cronológica do sistema, mas sim de uma idade virtual, normalmente entre o instante inicial e o tempo decorrido desde que o sistema era novo. Um modelo de idade virtual é inteiramente caracterizado pela taxa de avaria de um novo sistema e pelas suposições de idade virtual. Kijima(1989) propôs duas classes de modelos de idade virtual. Um modelo Tipo

I que pressupõe que uma reparação rejuvenesce a idade virtual numa quantidade proporcional ao último tempo entre avarias, enquanto o modelo do Tipo II supõe que a quantidade rejuvenescida é proporcional à idade virtual logo antes da reparação. Um caso particular é considerar que a eficiência da reparação é uma constante $\rho \in [0,1]$, denominada de fator de restauração (Nguyen et al., 2017). Em suma, Kijima no seu primeiro modelo, presume que as reparações servem apenas para remover os danos criados na última avaria. No seu segundo modelo, presume que a ação de reparação poderia remover todos os danos acumulados até aquele instante de tempo (Kahle, 2007).

No modelo de Kijima Tipo I, a idade virtual do sistema é dada por:

$$v(i) = \sum_{k=1}^{i} D_k X_k, \tag{42}$$

onde D_1, D_2 … é uma sequência de variáveis aleatórias com suporte no intervalo $[0,1]$.

No modelo de Kijima Tipo II, a idade virtual expressa-se por:

$$v(i) = \sum_{k=1}^{i} \left(\prod_{j=k}^{i} D_j \right) X_k. \tag{43}$$

Neste modelo, as condições de D_k são as mesmas do modelo de Kijima Tipo I.

É de referir que os modelos de reparação perfeita e reparação mínima são casos particulares em que $v(i) = 0$ e $v(i) = T_i$, $i = 1,2, \ldots$, respetivamente.

O processo de renovação de tendência (TRP) é um tipo diferente de modelo de reparação imperfeita. O TRP é definido como um processo de renovação transformado no tempo onde a transformação é realizada via a função intensidade $\lambda(\cdot)$ (Gámiz & Lindqvist, 2016). Sendo $\lambda(t)$ uma função não negativa definida para $t \geq 0$, e seja $\Lambda(t) = \int_0^t \lambda(u)du$. Os processos $S_1, S_2, \ldots$ são chamados TRP $(F, \lambda(.))$ se os processos transformados $\Lambda(S_1)$, $\Lambda(S_2)$… são $RP(F)$, ou seja, se $\Lambda(S_i) - \Lambda(S_{i-1}), i = 1,2, \ldots$, são variáveis aleatórias independentes com a mesma distribuição de função distribuição F. A Figura 12 ilustra a definição do processo TRP.

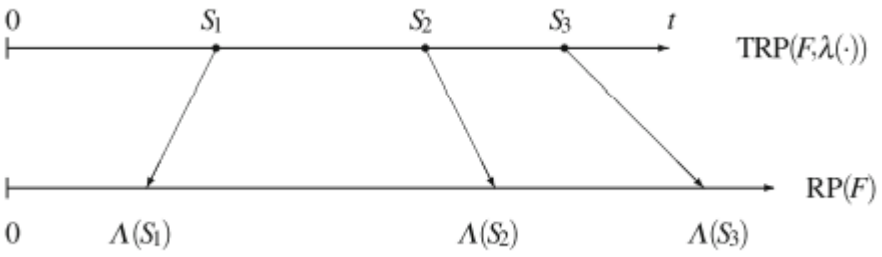

Figura 12 - Ilustração da definição do processo TRP [(Gjmiz et al., 2011)].

De acordo com Lindqvist et al. (2003), o problema de distinguir entre os dois tipos "extremos" de reparação, corresponde à primeira "dimensão" de uma análise de um sistema reparável sob a forma de um cubo modelo (Figura 13). A segunda “dimensão” é o aparecimento ou não de tendência nos tempos entre avarias. Finalmente, a terceira “dimensão” corresponde à existência de heterogeneidade não observada entre os sistemas.

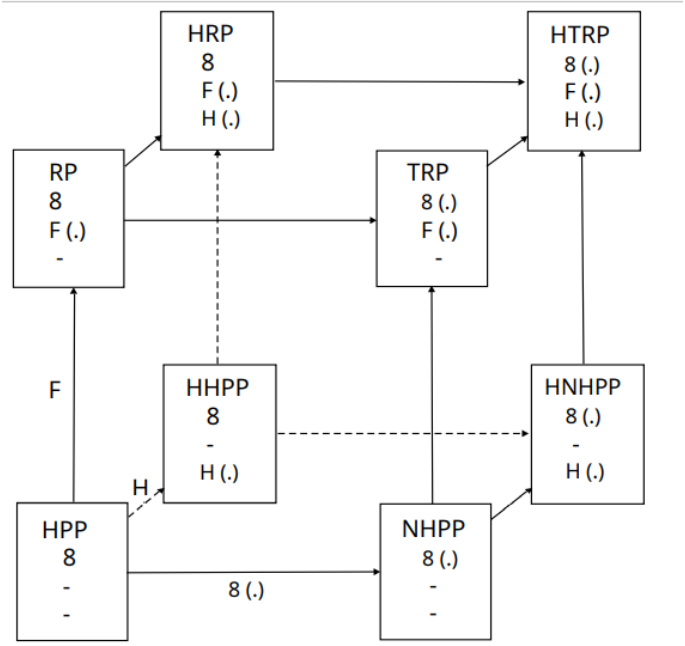

Figura 13 - cubo modelo [(Lindqvist, 1999)]

É de referir que na construção do cubo modelo (Figura 13) os processos HPP, NHPP, RP e TRP com heterogeneidade estão identificados com “H” antes das respetivas abreviaturas. Ao analisar o cubo conclui-se que cada vértice do cubo representa um modelo, e as linhas que os conectam correspondem à mudança de uma das três "coordenadas" $(F, \lambda\,(\cdot), H)$. O movimento para a direita corresponde a introduzir uma tendência temporal. Um movimento para cima corresponde a um caso onde os modelos de Poisson não se adequam. Um movimento em profundidade (para dentro) corresponde à contemplação de heterogeneidade (Lindqvist, 1999).

2.3 Simulação

No contexto empresarial, muitos problemas são de tal maneira complexos que os gestores têm que tomar as suas decisões com base em sistemas de suporte de ajuda. Esses sistemas devem ser adequados para fornecer respostas satisfatórias às suas estratégias, questões táticas ou operacionais. A maioria desses problemas podem ser modelados como sistemas discretos complexos. As tomadas de decisão relativamente aos mesmos podem ser abordadas usando técnicas de simulação (Bem-Daya et al., 2009).

Com o rápido crescimento do poder de computação nas últimas décadas, a simulação por computador tornou-se muito popular em muitas aplicações, principalmente onde a experimentação real é cara (em termo monetários, tempo, ou outra razão) ou talvez impossível. Os modelos de simulação são frequentemente usados para avaliar a fiabilidade de sistemas complexos de grande escala. À medida que os modelos de simulação se tornam mais realistas e os seus graus de liberdade aumentam, estimar a fiabilidade permanece um desafio, porque cada replicação de simulação é cara a nível computacional. Além disso, a avaliação da fiabilidade de um sistema altamente fiável geralmente requer um grande número de replicações de simulação para quantificar com precisão uma pequena probabilidade de evento de falha ou outros indicadores (Choe et al., 2015).

Os algoritmos para estimar a fiabilidade de um sistema geralmente podem ser agrupados em duas categorias: métodos analíticos e métodos de simulação. Os métodos analíticos apresentam abordagens elegantes para enumerar sequências de avaria significativas, mas envolvem suposições simplificadoras restritivas sobre o comportamento estrutural e limites aproximados durante o cálculo da probabilidade. Os métodos de simulação são mais simples de implementar, robustos em desempenho e podem incorporar comportamento estrutural realista, mas tendem a ser computacionalmente caros para sistemas práticos de alta fiabilidade (Mahadevan & Raghothamachar, 1999)

2.3.1 Modelo uniforme

É fácil perceber que um algoritmo pode gerar uma sequência de números que possui precisamente as mesmas estatísticas de uma sequência de números gerada por um processo aleatório natural, porque, em princípio, um algoritmo pode ser feito para produzir qualquer sequência numérica dada, de comprimento finito. Uma sequência de números gerada por um algoritmo, mas permutável com uma sequência verdadeiramente aleatória, é chamada de pseudo-aleatória. A diferença entre uma sequência verdadeiramente aleatória e uma sequência pseudo-aleatória é que a sequência gerada por um algoritmo pode se repetir.

Existem diversos algoritmos para gerar números aleatórios, diferindo uns dos outros na medida em que exibem as seguintes propriedades desejáveis (Ecker & Kupferschmid, 1988):

- Longo período de repetição - é desejável que o algoritmo seja capaz de completar uma simulação sem que os números aleatórios se comecem a repetir.
- Independência estatística aparente de números sucessivos - é necessária para garantir que a sequência pseudo-aleatória seja permutável com uma sequência verdadeiramente aleatória
- Distribuição uniforme – Por vezes, uma simulação requer intervalos de tempo aleatórios que são extraídos de uma distribuição de probabilidade uniforme, ou

seja, uma distribuição cujos valores são igualmente prováveis dentro de um determinado intervalo. Mesmo que tal não aconteça, é conveniente ter um gerador de números aleatórios que produza valores uniformemente distribuídos porque os valores de uma distribuição uniforme são fáceis de transformar em valores de qualquer distribuição de probabilidade arbitrária que possa ser desejada.

- Velocidade- As simulações reais normalmente requerem muitos números aleatórios, portanto, a velocidade com que um algoritmo pode ser executado pode ter uma grande influência no tempo consumido pelo computador. Normalmente, a simplicidade é a chave para a alta velocidade.
- Repetibilidade- Embora seja importante que a sequência seja pseudo-aleatória no sentido já definido, deve ser possível usar a mesma sequência pseudo-aleatória de uma simulação para a próxima.

O gerador de números aleatórios mais simples é o algoritmo congruencial multiplicativo, que se encontra representado, numa forma resumida, na Figura 14.

```
0. Escolha dos paramteros p e m
1. u0 = numero de inicio da sequencia, diferente de 0 e impar
       k inicia em 0
2. uk+1 = (muk)mod p
3. k é obtido por k +1
    ir para o passo 2
```

Figura 14 - Algoritmo congruencial multiplicativo [adaptado de (Ecker & Kupferschmid, 1988)]

É de salientar que todos os números usados no algoritmo são inteiros não negativos. O parâmetro m é chamado de multiplicador, o parâmetro p é chamado de módulo, e o número inicial u_0 é chamado de semente. O princípio de funcionamento deste algoritmo é a fórmula de recorrência:

$$u_{k+1} = (m\, u_k) mod\ p, \tag{44}$$

onde $x\ mod\ p$ é o resto da divisão inteira de x por p.

Note-se que cada número na sequência é gerado a partir do anterior.

2.4 Seleção de modelos

A seleção do modelo estatístico adequado é uma tarefa crucial neste trabalho. Um modelo adequado é um modelo que inclui um número reduzido de parâmetros a serem estimados (modelo parcimonioso) e que se adeque bem dados observados. A

literatura descreve vários critérios para seleção de modelos [veja-se, por exemplo (Bozdogan, 1987),(Wolfinger, 1993),(Littell et al., 1996)] destacando-se, por serem os mais utilizados, aqueles que são baseados na função de verosimilhança. A função de verossimilhança (L) representa o quão provável é uma determinada população produzir uma amostra observada.

O Teste da Razão de Verossimilhança (TRV), o Critério de Informação de Akaike (AIC) e o Critério de Informação Bayesiano ou de Schwarz (BIC) são três métodos de seleção de modelos baseados na L.

O TRV considera a estatística $TS = 2 \times (ML2 - ML1)$, onde $ML2$ representa o máximo do logaritmo natural da função de verossimilhança $(L2)$ para o modelo com maior número de parâmetros $M2$ e $ML1$ representa o máximo do logaritmo natural da função de verossimilhança $(L1)$ para o modelo mais simples $M1$. No caso de $M1$ ser o melhor modelo para o conjunto de dados disponíveis, TS segue, assintoticamente, uma distribuição qui-quadrado não centrada e cujo seu número de graus de liberdade é diferença entre o número de parâmetros dos dois modelos em comparação. A hipótese de que o modelo $M1$ apresenta melhor ajuste é rejeitada, ao nível de significância de $\alpha \times 100\%$, sempre que o valor observado de TS é superior ao quantil de ordem $1 - \alpha$ da distribuição qui-quadrado.

O AIC foi introduzido por Hirotugu Akaike em 1973 no artigo ((Akaike, 1998)). O AIC bonifica a qualidade de ajuste (altos valores para a função de verossimilhança) e, por outro lado, penaliza a quantidade de parâmetros do modelo. Esta penalização previne o sobre-ajuste. Dada uma coleção de modelos estatísticos candidatos para os dados, o modelo com menor AIC é o escolhido de acordo com este critério. A estimativa do AIC para um determinado modelo com k parâmetros pode ser expressa por:

$$AIC = 2 \times k - 2 \times ln\,(\hat{ML}), \quad \textbf{(45)}$$

onde $\hat{ML}$ é o máximo da função de verossimilhança do modelo. Note que o AIC não fornece uma medida de qualidade do modelo global, é uma medida relativa no que diz respeito à comparação entre modelos candidatos. Portanto, no caso de todos os modelos propostos se ajustam mal aos dados, o AIC não tem a capacidade para alertar para tal facto.

O BIC (Schwarz, 1978) é, como já referido, um critério para seleção de modelos entre um conjunto finito de modelos. O modelo com o valor de BIC mais baixo é o preferido. Este critério é baseado na função de verossimilhança L e está intimamente relacionado com o critério AIC. O BIC é expresso por:

$$BIC = k \times ln\,(n) - 2 \times ln\,(\hat{ML}), \quad \textbf{(46)}$$

onde $\hat{ML}$ é o máximo da função de verosimilhança do modelo, n é a dimensão da amostra e k é o número de parâmetros estimados.

Os testes de hipóteses de avaliação de presença de tendência na taxa de ocorrência de avarias ou de existência de aleatoriedade são também muito úteis na seleção de modelos no contexto deste trabalho, em particular, na seleção dos modelos para os processos geradores de avarias. Existem vários testes de hipóteses com este propósito, entre os quais se destacam o teste de Laplace (selecionado no âmbito deste trabalho) e o teste de Lewis-Robinson.

Um teste de hipótese é um processo estatístico que auxilia o investigador a tomar uma decisão do tipo sim ou não sobre uma ou mais populações previamente definidas, a partir de uma ou mais amostras oriundas dessas populações.
Para realizar um teste de hipóteses, começa-se por formular duas hipóteses distintas: hipótese nula (H_0) e a hipótese alternativa ou fundamental (H_1). De seguida, é realizado um procedimento, baseado numa função da amostra (ou amostras), designada de estatística de teste (TS), que averigua se se deve rejeitar ou não H_0. Os valores de TS para o qual se rejeita H_0 são denominados de região crítica. A dimensão da região crítica é influenciada pelo nível de significância α a que o teste está a ser submetido. Note-se ainda que a posição da região crítica varia consoante a classificação dos testes de hipóteses (estes classificam-se de acordo com a forma da hipótese fundamental (H_1)) que podem ser bilaterais ou unilaterais (à direita ou à esquerda). Na Figura 15 está representa a vermelho a região crítica de um teste bilateral para uma TS que segue a distribuição normal padronizada:

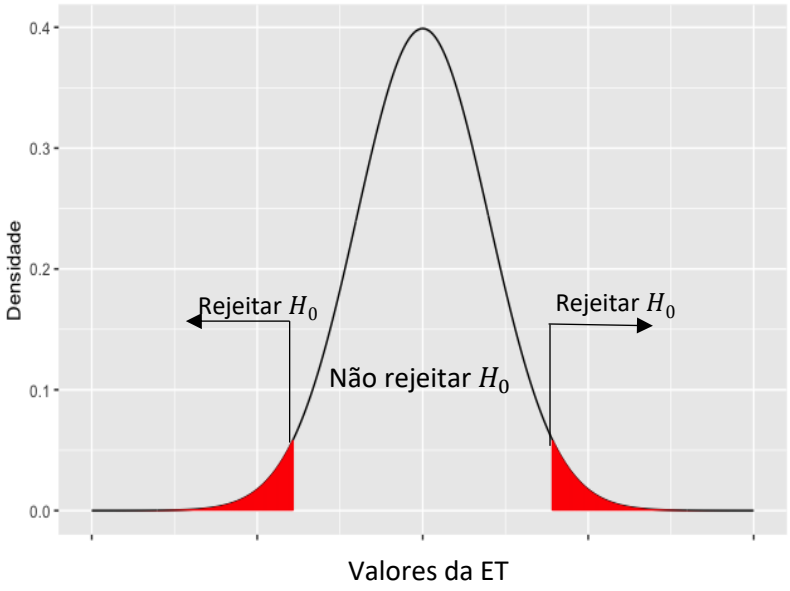

Figura 15 - Teste de hipóteses bilateral [(Sandrini, 2017)]

O teste de Laplace é um teste estatístico que declara que a hipótese nula (H_0) é associada aos processos de renovação, ou seja, não existe tendência entre os tempos das ocorrências, portanto os eventos ocorrem de forma aleatória. Por outro lado, a hipótese alternativa (H_1) é associada ao processo não homogêneo de Poisson (NHPP),

verifica-se tendência entre os tempos de ocorrências, e conclui-se, portanto, que os eventos não ocorrem de forma aleatória:

H_0: o processo de falhas é modelado por um PR
H_1: o processo de falhas é modelado por um NHPP

Após se estabelecerem as hipóteses de teste, o procedimento que permite averiguar se devemos rejeitar ou não H_0 , inicia-se pelo cálculo de μ_{obs}, que tem duas fórmulas de cálculo distintas, uma para quando o histórico de avarias é limitado pelo tempo de observação e outra para quando o histórico é limitado pelo número de avarias, que são dadas respetivamente por:

$$\mu_{obs} = \sqrt{12N} \times \left(\frac{\sum_{i=1}^{N} t_i}{N \times To} - 0.5\right) \quad \textbf{(47)}$$

$$\mu_{obs} = \sqrt{12(N-1)} \times \left(\frac{\sum_{i=1}^{N-1} t_i}{(N-1) \times To} - 0.5\right). \quad \textbf{(48)}$$

Nas expressões anteriores, $t_i, i = 1,2, \dots, N$, representam os tempos as ocorrências e T_0 é, no primeiro caso, o tempo de observação do sistema e o tempo de ocorrência da última ocorrência no segundo caso.

A $TS = U$ segue, quando H_0 é verdadeira uma distribuição aproximadamente normal de média zero e variância 1 (normal padrão).

De seguida é necessário calcular o valor de valor-p[6] ou valor de prova que é dado pela expressão seguinte no caso do teste ser bilateral:

$$\textit{valor-p} = 2 \times P(|U| > u_{obs} | H_0 \text{ é } verdadeira) = 2 \times (1 - \phi(u_{obs})), \quad \textbf{(49)}$$

onde $\phi(.)$ representa a função de distribuição de uma variável aleatória com distribuição normal padrão.

Por fim, se o valor de prova (valor-p) for inferior a α (nível de significância) rejeita-se H_0.

2.5 Manutenção

A manutenção refere-se a um conjunto de atividades ou tarefas executadas num sistema com o propósito de o manter num estado em que este possa executar de forma aceitável as funções que lhe foram atribuídas. A manutenção dos sistemas é

[6] p-value, na literatura em inglês.

necessária para aumentar a fiabilidade geral, evitar falhas dos mesmos e diminuir os custos (Azadeh & Abdolhossein Zadeh, 2016).

A Figura 16 representa uma visão geral dos diferentes tipos de políticas de manutenção.

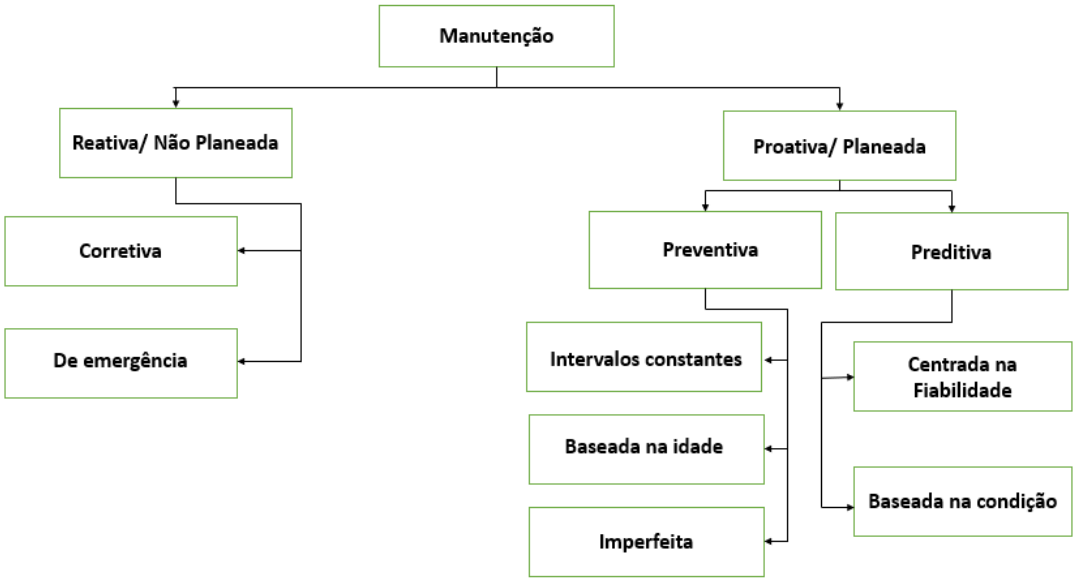

Figura 16 - Visão geral de diferentes tipos de políticas de manutenção [adaptada de (Azadeh & Abdolhossein Zadeh, 2016)]

Neste documento é dado um enfase a manutenção preditiva uma vez que é o tipo de manutenção que está diretamente relacionada com o tema do mesmo.

A manutenção preditiva significa manutenção com foco na previsão de falhas, através do acompanhamento de certos parâmetros e condições dos equipamentos. Esta não surge como substituta da manutenção corretiva e preventiva, mas sim como uma ferramenta adicional, que busca minimizar, através do monitoramento de parâmetros específicos, custos de manutenção e perdas em equipamentos (de Faria et al., 2015).

Este tipo de manutenção pode-se dividir em dois grandes ramos: Manutenção Centrada na Fiabilidade e Manutenção Baseada na Condição.

Manutenção Centrada na Fiabilidade

O desenvolvimento dos métodos de Manutenção Centrada na Fiabilidade (RCM) surgiu na indústria aeronáutica, sendo que foram os grupos de gestão da manutenção formados dentro desta indústria que desenvolveram este processo de manutenção, que foi publicado nas décadas de 70 e 80 (Sajaradj et al., 2019). Posteriormente, observou-se que a RCM também poderia ser aplicada a outros ramos de engenharia

para possíveis melhorias na manutenção dos sistemas. Como resultado disso, muitos investigadores aventuraram-se no estudo do RCM dando-lhe diversas definições baseadas em diversas aplicações (Airoboman et al., 2016).

Segundo Niu et al., (2010), a RCM pode ser definida como uma abordagem de melhoria industrial focada na identificação e no estabelecimento de políticas operacionais de manutenção e de melhoria de capital que irão gerir os riscos de falha do equipamento de uma forma mais eficaz. De acordo com Eriksen et al., (2021) para implementar a RCM deve-se responder às seguintes perguntas:

- Quais são as funções e padrões de desempenho associados ao equipamento no seu atual contexto operacional?
- De que forma pode deixar de cumprir plenamente as suas funções?
- Qual é a causa de cada falha funcional?
- O que acontece quando ocorre cada falha?
- De que forma cada falha importa?
- O que pode ser feito para evitar cada falha?
- O que deve ser feito se não for encontrada uma tarefa preventiva adequada?

Se a RCM for aplicada corretamente, poderá reduzir-se a quantidade de trabalho da manutenção de rotina em 40–70%. Os benefícios da RCM geralmente podem ser classificados em duas grandes categorias: redução de riscos e economia de custos (Niu et al., 2010). De forma a reforçar a importância da manutenção RCM no contexto empresarial, a Tabela 1 apresenta alguns casos de estudo de aplicação desta política de manutenção e os seus respetivos efeitos.

Tabela 1 - Casos de estudo relativos à manutenção RCM

Referencias bibliográficas	Descrição dos casos de estudo
(Yavuz et al., 2019)	Neste artigo foi realizada uma aplicação da abordagem RCM em máquinas de embalagem com propósito de analisar o seu respetivo efeito no OEE. Como resultado desta aplicação, a disponibilidade do equipamento aumentou e problemas relacionados com a qualidade foram eliminados. De salientar que, em Maio de 2018 o OEE era cerca de 76,2%, enquanto que em Outubro de 2018 o OEE já apresentou aproximadamente o valor de 79,2%, portanto verificou-se um aumento positivo no OEE do equipamento estudado.

(Rahimdel et al., 2013)	Neste artigo, foi analisada a fiabilidade do sistema hidráulico de máquinas de perfuração na Mina de Cobre Sarcheshmeh no Irão. A análise mostrou que quatro máquinas estudadas possuem gráficos de fiabilidade e comportamento de falha muito semelhantes. Em todas elas, o sistema hidráulico apresenta um índice de falhas redutor e entraram na sua vida útil após aproximadamente 600 h. De acordo com a análise de fiabilidade, o intervalo de manutenção para um nível de fiabilidade de 80% é de 10 h. Isso significa que o sistema hidráulico das máquinas deve ser verificado e inspecionado a cada 10 horas.
(Nejad et al., 2014)	Este artigo apresenta um plano de manutenção baseado na fiabilidade destinado a componentes da caixa de engrenagens das turbinas eólicas. As engrenagens e rolamentos são classificados com base nos seus danos por fadiga e é desenvolvido um mapa de manutenção para focar nos componentes com maior probabilidade de falha por fadiga e menor nível de fiabilidade. Ao usar este plano, o inspetor de manutenção procura defeitos naqueles componentes com maior probabilidade de falha, em vez de examinar todas as engrenagens e rolamentos. Esta abordagem pode ter como vantagem detetar a fonte de falha num menor tempo.
(Afefy et al., 2019)	Neste artigo, é proposta uma nova estrutura de manutenção centrada na fiabilidade (RCM). A estrutura proposta de RCM foi aplicada e avaliada em um caso estudo real; mais concretamente na fábrica de produção de açucar “Fayoum Sugar Works Company” localizada no Egito. Os resultados revelam que a disponibilidade aumentou de 57,1% para 90,74% e a fiabilidade também aumentou de 99,73% para 99,88%.
(Morad et al., 2014)	Neste artigo foi realizado um estudo de fiabilidade baseada na manutenção, relativamente aos camiões na Mina de Cobre Sungun. A análise realizada indica que as rodas são o componente mais crítico. Posteriormente, a análise baseada na disponibilidade indica que a disponibilidade média do camião basculante é de 91,8% às 12.000 horas de operação. Os resultados da pesquisa são fornecidos à equipa de gestão da manutenção para planear melhores decisões sobre a operação de manutenção, monitoramento da condição dos itens críticos, stock de peças de reposição e seu nível de reordenamento, o que leva à redução do tempo de inatividade do equipamento.

Manutenção Baseada na Condição

A Manutenção Baseada na Condição (CBM) é uma estratégia de manutenção preditiva e preventiva baseada no estado de condição técnica dos ativos. A condição dos ativos é determinada pela monitorização de fatores específicos que revelam o seu estado, como por exemplo a temperatura, corrosão ou fluxo (Taboada et al., 2021). Em geral, o principal objetivo da CBM é fazer uma avaliação em tempo real das condições dos equipamentos para tomar decisões de manutenção, reduzindo assim as manutenções desnecessárias e os custos associados (Goyal & Pabla, 2015). Esta estratégia de manutenção baseia-se na suposição de que a maioria das falhas não ocorrem instantaneamente, e é possível detetar o seu aparecimento numa fase inicial do processo de deterioração. O principal desafio é determinar o momento exato em que a manutenção deve ser realizada e identificar a ação mais adequada (Teixeira et al., 2020). De forma a reforçar a importância da CBM no contexto empresarial, a Tabela 2 apresenta alguns casos de estudo de aplicação desta política de manutenção e os seus respetivos efeitos.

Tabela 2 - Casos de estudo relativos à manutenção CBM

Referencias bibliográficas	Descrição dos casos de estudo
(Duan et al., 2020)	Neste artigo, desenvolveu-se um modelo CBM estocástico para bombas de navios, cujas informações provenientes do sensor podem ser modeladas por um processo estocástico. É proposta uma política de CBM de dois níveis, em que as variáveis de decisão na otimização da CBM são determinadas por um algoritmo computacional. Por fim é ilustrada a superioridade desta nova abordagem por um estudo numérico e pelo caso real da bomba do navio.
(Q. Liu et al., 2022)	Neste artigo, é proposto um modelo de manutenção baseado na condição (CBM) para sistemas reparáveis sujeitos a processos de falha dependentes (falhas suaves devido à degradação do sistema e falhas graves devido a choques aleatórios). Os efeitos das ações de manutenção (manutenção preventiva e manutenção corretiva) são também considerados no modelo. O modelo é utilizado para lidar com o problema de otimização de manutenção de linhas de transmissão de alta tensão. Os resultados mostram que o modelo é viável e pode ser aplicado no campo da otimização da manutenção.
(Wang et al., 2012)	Este artigo apresenta um modelo para auxiliar a tomada de decisão de manutenção com base em dados monitorados relativos a concentrações de metais medidas e observadas em amostras de óleo de uma frota de motores diesel marítimos. É demonstrado

	como o modelo desenvolvido pode ser utilizado num processo de decisão que determina o momento adequado para realizar a substituição preventiva com base num critério de minimização de custos.
(Ma et al., 2020)	Neste artigo, investigou-se as abordagens de análise de fiabilidade e otimização de manutenção de dois equipamentos de refrigeração. Foi desenvolvida uma política RCM com base com base em informações de monitoramento de temperatura. Os resultados mostram que a política de manutenção usada supera uma política clássica de manutenção baseada em idade quando o custo de tempo de inatividade não é baixo ou o custo de substituição preventiva da barra fechada não é muito alto.
(Wan et al., 2018)	O custo de manutenção de uma turbina a gás é significativamente maior do que seu custo de compra original. Este estudo apresenta uma nova abordagem de manutenção baseada na condição (CBM) aplicada numa central termoelétrica em Hangzhou, China. A eficácia desta abordagem é comprovada, pois uma geração extra de energia de 302.640 MW-h pode ser alcançada devido à manutenção reprogramada.

DESENVOLVIMENTO DA FERRAMENTA

3 DESENVOLVIMENTO DA FERRAMENTA

Após a exposição dos conceitos teóricos essenciais para o desenvolvimento desta dissertação, é iniciado o desenvolvimento da parte prática, nomeadamente, a elaboração da ferramenta computacional. O fluxograma da Figura 17 permite ter uma visão geral da estrutura lógica da ferramenta a desenvolver, desde a leitura dos dados até ao cálculo dos indicadores de fiabilidade/disponibilidade.

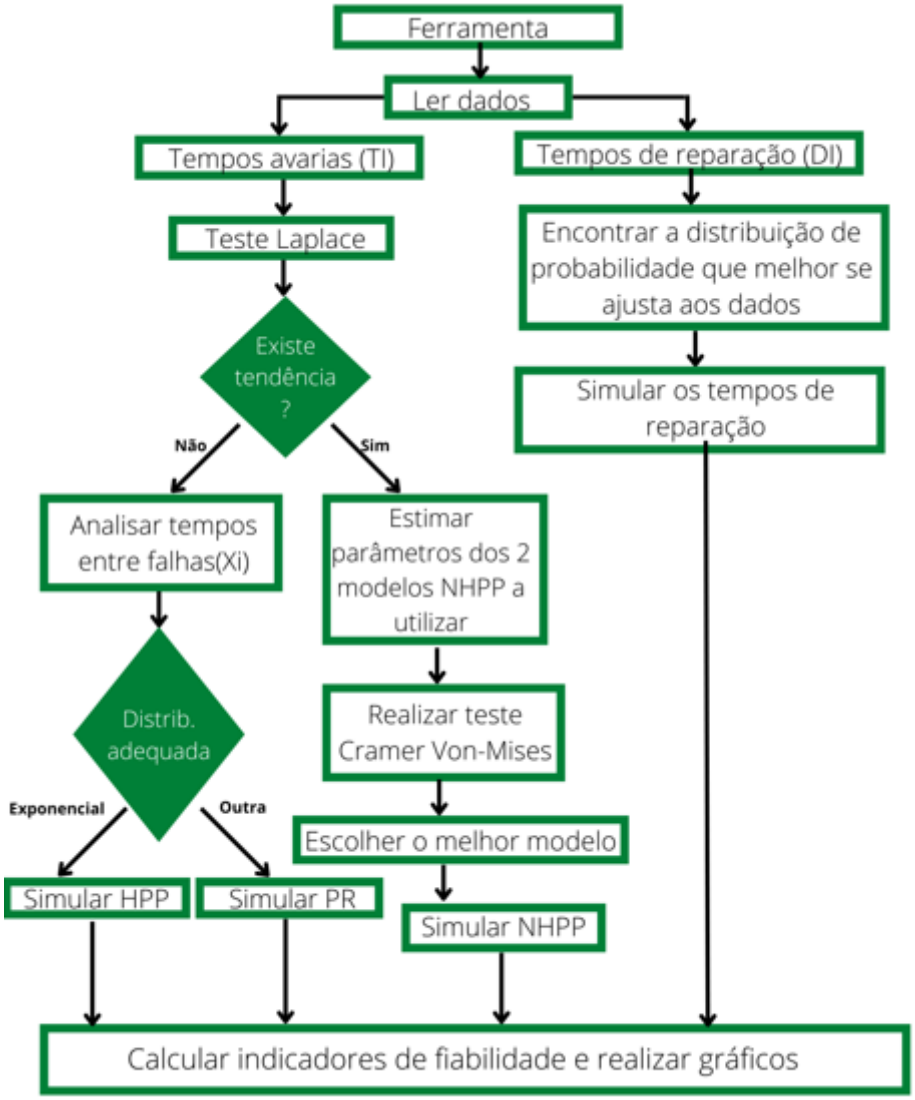

Figura 17 - Fluxograma do modelo lógico da ferramenta [Autor]

3.1 Terminologia utilizada e leitura de dados

Nesta secção os tempos das ocorrências são denodos de $T_1, T_2, T_3 \cdots$, os tempos de reparação por $D_1, D_2, D_{3,} \cdots$ e os tempos entre ocorrências por $X_1, X_2, X_3, \cdots$. A Figura 18 representa o esquema destas notações adotadas:

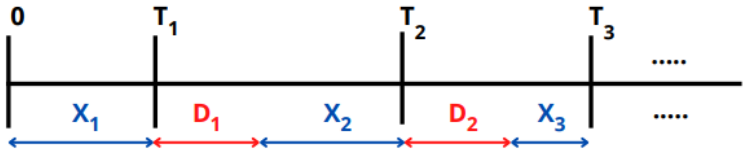

Figura 18 - diagrama dos tempos de ocorrências [Autor]

Pela análise do diagrama facilmente se percebe que se forem conhecidos dois elementos (tempos), facilmente se consegue descobrir o elemento em falta. Veja-se, por exemplo, que $X_2 = T_2 - (T_1 + D_1)$.

Para a construção da ferramenta computacional, e sem perda de generalidade para a leitura de dados armazenados em outros formatos, estabeleceu-se que esta deve ler uma folha Excel onde devem constar na primeira coluna da mesma os dados referentes aos tempos das ocorrências (T_i) e na segunda coluna os dados referentes aos tempos de reparação (D_i).

Após a importação dos dados, foi escrita uma função em Python que recebe estes e calcula os tempos entre ocorrências, ficando assim ficam conhecidos todos os elementos necessários para o funcionamento da ferramenta. A Figura 19 representa a função que permite a obtenção dos X_i:

```
def dadosx (T,D):
    N=len(T)
    x=list()
    # x1=T1
    x.append(T[0])
    for i in range(0,N-1):
        # X(i+1)=T(i+1)-(Ti+Di)
        v=T[i+1]-(T[i]+D[i])
        x.append(v)
    return(x)
```

Figura 19 - Função python para calcular X_i [Autor]

3.2 Estudo da tendência dos dados

Uma vez efetuada a leitura do histórico dos tempos de avarias e a obtenção dos tempos entre avarias, é necessário recorrer a um teste para verificar se existe

tendência temporal nas avarias, ou seja, para verificar se as avarias ocorrem de forma aleatória no tempo ou se revelam alguma tendência. Nesta tarefa foi usado o teste de Laplace descrito no capítulo anterior.

Na Figura 20 apresenta-se a função Python que implementa o teste de Laplace bilateral.

```
from math import sqrt
import statistics
def laplace(d,a,t):
    N=len(d)
    if t==0:
        # Se o historico de avarias for limitado pelo numero de avarias
        u= sqrt(12 * (N - 1)) * ((sum(d) - d[len(d) - 1]) / ((N - 1) * d[len(d) - 1]) - 0.5)
    else:
        # Se o historico de avarias for limitado pelo tempo de observação
        u = sqrt(12 * N) * ((sum(d)) / (N * t) - 0.5)
    print(f'O valor de Uobs é de {u:.4f}.')

    dn=statistics.NormalDist(0,1)
    #calculo do valor de prova
    vp=2*((dn.cdf(99999999)-dn.cdf(abs(u))))
    print(f'O valor de Vp é de {vp:.4f}.')
    if vp>a:
        print(f'{vp:.4f} > {a}. Não se rejeita H0. Logo a taxa de ocorrencia de avarias não '
              f'apresenta tendencia.')
        f=0
        return f
    else:
        print(f'{vp:.4f} < {a} Rejeita-se H0.A Taxa de ocorrencia de avarias '
              f'apresenta tendencia.')
        f=1
        return f
```

Figura 20 -Função python que implementa o teste de Laplace [Autor]

A função `Laplace` recebe um ficheiro com informação sobre as avarias e retorna o resultado 1 ou 0, consoante os dados apresentem ou não tendência.

3.3 Dados sem tendência

Caso não se rejeite a hipótese nula no teste de Laplace, ou seja, não se conclua que existe tendência, dois cenários colocados: (1) os tempos de ocorrências são gerados por um HPP; (2) os tempos de ocorrências são gerados por um PR. Conforme abordado no capítulo anterior, se os tempos entre ocorrências seguirem uma distribuição exponencial o processo a adotar será o HPP, caso sigam outra distribuição de probabilidade será considerado um processo de renovação.

3.3.1 Ajuste de distribuições

A seleção da melhor distribuição de probabilidade para os tempos entre avarias é feita com recurso à função do python `distfit`. Esta função ajusta diferentes distribuições univariadas aos dados e retorna uma seriação de distribuições juntamente com as estimativas dos seus parâmetros. Esta seriação é baseada em diferentes critérios, desde testes de hipóteses, soma de quadrados dos desvios entre a distribuição teórica e a empírica e outras métricas de seleção de modelos.

No âmbito deste trabalho, as distribuições de probabilidade candidatas foram: normal, lognormal, beta, exponencial, Weibull (biparamétrica) e gamma. A distribuição selecionada foi a que surgiu no topo da lista ordenada de distribuições.

O raciocínio descrito acima materializa-se na função escrita em linguagem python `funcaomodelo3` apresentada na figura seguinte.

```
def funçaomodelo3(y):
    e = array(y)
    #escolher as distribuições que queremos ajustar aos dados
    dist = distfit(distr=['norm','lognorm','beta','expon','dweibull','gamma'])
    dist.fit_transform(e)
    # os parametros da distribuição que melhor se ajusta
    a = dist.model['params']
    #o nome da distribuição que melhor se ajusta
    b = dist.model['name']
    prmts=[]
    prmts.append(a)
    prmts.append(b)
```

Figura 21 - Função python para escolher distribuição de probabilidade mais adequada a dados [Autor]

Conforme já referido, o HPP é adequado sempre que as avarias ocorram de forma aleatória e o tempo entre avarias é bem modelado pela distribuição exponencial. Portanto, em termos práticos, será usado o HPP sempre que a função anterior retornar como argumento de saída a distribuição exponencial e será considerado o PR caso seja retornada uma das restantes distribuições candidatas. Após a seleção de uma distribuição adequada para os tempos entre avarias, o que equivalente à seleção do modelo adequado para o processo gerador das avarias, segue-se a etapa da simulação de amostras dos modelos estimados.

3.3.2 Simulação de amostras a partir de Processos Homogéneo de Poisson

Existem, na literatura, vários algoritmos que permitem simular amostras de tempos a partir de HPP (veja-se, por exemplo, os trabalhos de Chen (2016) e de Burnecki & Weron (2010)). Um exemplo de um algoritmo para a geração de uma amostra HPP encontra-se representado na Figura 22 e foi baseado no trabalho de Chen (2016). É de

salientar que este algoritmo considera o caso em que a simulação termina quando o valor simulado para o tempo de ocorrência excede T (limite superior do intervalo de tempo considerado).

```
Input: λ, T
1 Initialize n = 0, t0 = 0;
2 while True do
3 Generate u ~ uniform(0,1);
4 Let w = - ln u/λ; // so that w ~ exponential(λ)
5 Set tn+1 = tn + w;
6 if tn+1 > T then
7 return {tk}k=1,2,...,n
8 else
9 Set n = n + 1;
10 end
11 end
```

Figura 22 - Algoritmo para a geração de uma amostra de HPP [adaptado de (Chen, 2016)]

O código para python seguinte (Figura 23) implementa a sequência de ações apresentadas no algoritmo anterior.

```
def hpp(l,Tsimul):
    t=0
    numeros=list()
    while t< Tsimul:
        #gerar distribuição uniforme
        u=numpy.random.uniform (0,1)
        #gerar os tempos das ocorrencias
        t=t-(ln(u)/(l))
        numeros.append(t)
    return numeros
```

Figura 23 - Função python para gerar amostras de HPP [Autor]

A Figura 24 ilustra 3 realizações de processos HPP, limitadas pelo tempo máximo de simulação de 20 unidades de tempo (u.t.). Os pontos a azul resultam de um PPH com $\lambda = 0{,}3$ ocorrência/u.t., os pontos a laranja resultam de $\lambda = 1{,}0$ ocorrências/u.t. e, por fim, os pontos a verde traduzem a taxa de $\lambda = 3{,}0$ ocorrências/u.t.

Como se pode observar, e como era esperado, à medida que aumenta o valor de λ, o número de ocorrências no intervalo de tempo considerado também aumenta. Por outro lado, é clara a ausência de tendência dado o comportamento linear dos pontos.

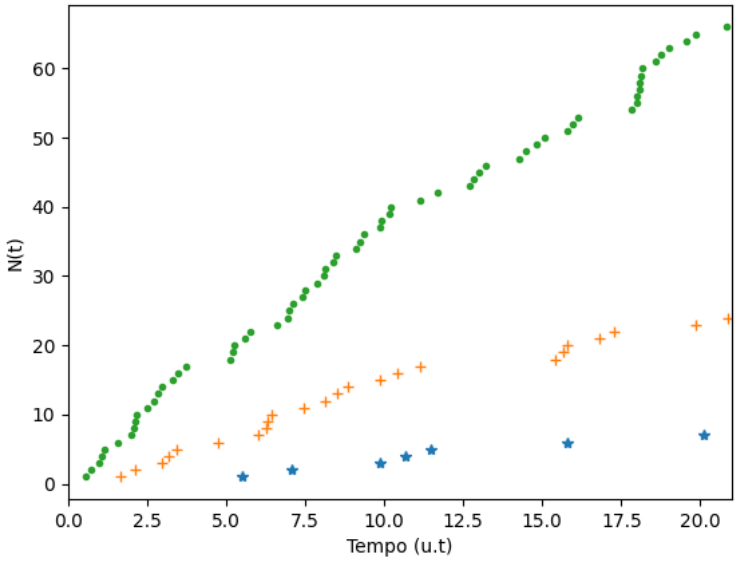

Figura 24 - Simulação de 3 amostras HPP[Autor]

3.3.3 Simulação de amostras a partir de Processo de Renovação

Recorde-se que o PR é uma generalização do HPP e também modela situações físicas com taxas de ocorrência constante. Neste processo, os tempos entre ocorrências não têm de seguir, necessariamente, uma distribuição exponencial como acontece no caso do PPH. O algoritmo implementado para simular amostras de um PR encontra-se ilustrado na Figura 25 (Sandrini, 2017 e referências aí incluídas):

```
Objectivo: Gerar Ti em (0,tF)

1. Iniciar T0 = 0;
2. Atribuir Ti = Ti-1 + ui; se Ti>tF, parar;

Saída: T1 ··· , Tn - tempos das ocorrências.

ui~F (.)
```

Figura 25 – Algoritmo para geração de uma amostra PR [(Sandrini, 2017)]

A função python que implementa o procedimento de simulação de amostras a partir de processos de renovação para as distribuições candidatas (normal, lognormal, exponencial, Weibull (bi-paramétrica) e gama), encontra-se no Anexo 1 por ser extenso.

A Figura 26 ilustra 3 amostras geradas a partir de processos de renovação com tempos entre avarias modelados com a distribuição normal (azul), com a distribuição gama (laranja) e com a distribuição lognormal (verde).

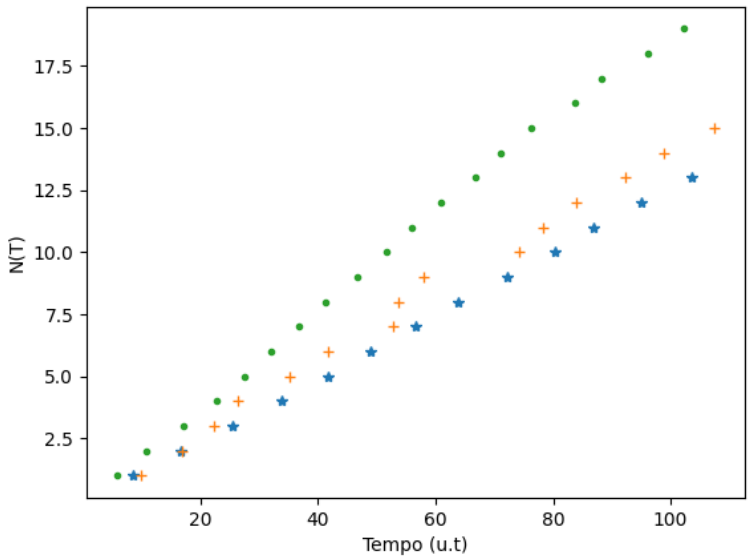

Figura 26 - Simulação de 3 amostras de PR [Autor]

Pela análise do gráfico rapidamente se conclui que, como esperado, não existe qualquer tendência nos tempos entre ocorrências.

3.4 Dados com tendência

Tal como já foi referido anteriormente, quando as avarias ocorrem de forma não aleatória, o NHPP é um possível modelo para o processo gerador das avarias. Nesta classe de modelos, a taxa de ocorrência não é constante podendo tomar várias formas. No âmbito deste trabalho, consideraram-se dois modelos para a taxa de avarias ($\lambda(t)$): o modelo lei de potência e o modelo de Cox-Lewis.

3.4.1 Modelo lei de potência

Este modelo é utilizado para análises de fiabilidade em sistemas reparáveis devido à função de intensidade $\lambda(t)$ admitir diferentes comportamentos, ou seja, modela diferentes tendências das avarias (Fraga & Pinto, 2013.). A função taxa de avarias expressa-se por:

$$\lambda(t) = \lambda\,\beta\,t^{\beta-1}, t, \lambda, \beta > 0. \quad \textbf{(50)}$$

Da análise da expressão anterior é fácil concluir que $\lambda(t)$ é uma função monótona crescente sempre que $\beta > 1$, é uma função monótona decrescente sempre que $\beta < 1$ e é uma função constante quando $\beta = 1$.
As estimativas de máxima verosimilhança dos parâmetros de $\lambda(t)$ são obtidos através das expressões:

$$\hat{\beta} = \frac{N}{\sum_{i=1}^{N} ln\,(\frac{T}{t_i})}\, e \tag{51}$$

$$\hat{\lambda} = \frac{N}{T^{\hat{\beta}}}, \tag{52}$$

onde t_i representa o tempo de ocorrência da avaria i, $i = 1,2, \dots, N$, N é o número de avarias e T é o tempo de observação do sistema.

A Figura 27 ilustra a função `funçaomodelo1` desenvolvida também em python que recebe os tempos das ocorrências e devolve as estimativas do modelo lei de potência.

```python
def funçaomodelo1(d, t):
    s = 0
    N = len(d)
    # historico limitado pelo num de avarias
    if t == 0:
        for c in range(0, N):
            s = s + ln(d[N - 1] / d[c])
        # calculo de beta
        b = N / s
        # calculo de lambda
        l = (N / (d[N - 1] ** b))
    #historico limitado pelo Tobs
    else:
        for c in range(0, N):
            s = s + ln(t / d[c])
        b = N / s
        l = (N / (t ** b))
    print(f'o valor de B é {b:.4f} e o valor de l é {l:.4f}')
    param=[b,l]
    return param
```

Figura 27 - Função python para estimar os parâmetros do modelo lei de potência [Autor]

3.4.2 Modelo de Cox-Lewis

Este modelo desenvolvido por Cox e Lewis é outra forma aceite para a função de ocorrência no modelo NHPP, sendo uma boa alternativa ao modelo de Crow. Trata-se de um modelo parametrizado por dois parâmetros α_0 e α_1 e cuja taxa de avarias dada pela expressão:

$$\lambda(t) = e^{\alpha_0 + \alpha_1 \times t}, t \geq 0, -\infty < \alpha_0, \alpha_1 < \infty. \tag{53}$$

O parâmetro α_0 é semelhante ao parâmetro de escala, enquanto que α_1 é um indicador de crescimento.

À semelhança do modelo anterior, para estimar os parâmetros do modelo de Cox e Lewis pode aplicar-se o método da máxima verosimilhança. A estimativa de verosimilhança de α_1 é obtida, de modo iterativo, pela seguinte expressão:

$$\sum_{i=1}^{N} t_i + N \times {\alpha_1}^{-1} - N \times T \times (1 - e^{-\alpha_1 T})^{-1} = 0. \quad \textbf{(54)}$$

Após a obtenção de uma estimativa para α_1, a estimativa de máxima verosimilhança do parâmetro α_0 é obtida pela fórmula seguinte:

$$\hat{\alpha}_0 = \ln\left(\frac{N\hat{\alpha}_1}{e^{\hat{\alpha}_1 \mathrm{T}} - 1}\right), \quad \textbf{(55)}$$

onde t_i representa o tempo de ocorrência da avaria i, $i = 1,2,\ldots,N$, N é o número de avarias e T é o tempo de observação do sistema.

É importante referir que este modelo é indicado para modelar tempos de reparação de sistemas reparáveis quando $\alpha_1 > 0$.

A Figura 27 ilustra a função `funçaomodelo2` desenvolvida em python que recebe os tempos das ocorrências e devolve as estimativas do modelo de Cox e Lewis.

```
def funçaomodelo2(d):
    N = len(d)
    cl = list()
    tot = sum(d)
    #calculo do parametro alfa1
    x = sympy.Symbol('x', real=True)
    expr2 = sympy.Eq(x * (tot + N * sympy.Pow(x, -1)) - x * N * d[N - 1] *
                     sympy.Pow(1 - sympy.exp(-d[N - 1] * x), -1),
                     0)
    expr_2 = sympy.simplify(expr=expr2)
    solution = sympy.nsolve(expr_2, -0.1)
    s = round(solution, 8)
    #calculo do parametro alfa0
    c = sympy.ln((s * N) / (sympy.exp(s * d[N - 1]) - 1))
    print(f'o valor de a0 é {c} e o valor de a1 é de {s}')
    parametros=[s,c]
    return parametros
```

Figura 28 - Função python para estimar parâmetros do modelo Cox-Lewis [Autor]

3.4.3 Avaliação da adequabilidade dos modelos

A avaliação da adequabilidade do modelo lei de potência e do modelo de Cow-Lewis aos dados foi realizada com recurso ao teste de Cramér-Von Mises. Perante uma amostra aleatória, $X_1, X_2, X_3, \dots, X_N$, o teste considera na hipótese nula (H_0) que o modelo é adequado. A estatística de teste do teste de Cramér-Von Mises (C_N) para o caso do modelo lei de potência e de Cox-Lewis é, respetivamente:

$$C_N = \frac{1}{12N} + \sum_{i=1}^{N}\left[\left(\frac{t_i}{T}\right)^{\hat{\beta}} - \frac{2i-1}{2N}\right]^2 \text{ e} \quad \textbf{(56)}$$

$$C_N = \frac{1}{12N} + \sum_{i=1}^{N}\left[\left(\frac{t_i}{\mathrm{T}}\right)^{\hat{\alpha}_1} - \frac{2i-1}{2N}\right]^2, \quad \textbf{(57)}$$

onde t_i representa o tempo de ocorrência da avaria i, $i = 1,2,\dots,N$, N é o número de avarias e T é o tempo de observação do sistema. $\hat{\beta}$ e $\hat{\alpha}_1$ são as estimativas do parâmetro que controla o tipo de tendência no modelo lei de potência e no modelo de Cox-Lewis, respetivamente.

É importante referir que, caso o fim de observação do histórico de avarias seja limitado pelo número de avarias, é necessário obter estimativas não enviesadas dos parâmetros de forma antes de substituir os mesmos nas expressões acima descritas. Essas estimativas expressam-se por:

$$\hat{\beta} = \frac{N-1}{N}\hat{\beta} \text{ e } \hat{\alpha}_1 = \frac{N-1}{N}\hat{\alpha}_1. \quad \textbf{(58) e (59)}$$

Após a obtenção do valor C_N para os dados observados, é necessário verificar o desvio do mesmo relativamente a um limite tabelado (a tabela de valores limite para o critério de Cramér Von Mises pode ser consultada no Anexo 2). O modelo que apresentar menor desvio é o modelo mais adequado para modelar os dados, sendo, por isso, o modelo escolhido.

3.4.4 Simulação de amostras a partir de Processos Não Homogéneos de Poisson

Tal como para o HPP, existem vários algoritmos para gerar realizações de NHPP (veja-se, por exemplo, os trabalhos de Pasupathy (2011) e de Chen (2016)). A Figura 29 ilustra um algoritmo para a geração de uma amostra a partir NHPP e que é baseado no algoritmo "Thinning" de Burnecki & Weron (2010).

Objectivo: Gerar T_i em $(0, t_f)$

1. Inicializar $t = 0, n = 0, \lambda = max_{t\in[0,t_f]}\lambda(t)$;
2. Atribuir $t = t - \ln(U[0,1])/\lambda$, se t>$t_f$, parar;
3. Se $U[0,1] \leq \lambda(t)/\lambda$, atribuir $n = n + 1, T_n = t$;
4. Voltar ao passo 2.

Figura 29 - Algoritmo para gerar NHPP [(Sandrini, 2017)]

No contexto deste trabalho, foram escritas duas funções em python para simular realizações de NHPP. Uma delas gera realizações NHPP com taxa de avarias governada pelo modelo lei de potência (Figura 30) e na segunda função a taxa de avarias é governada pelo modelo de Cox-Lewis (Figura 31):

```
def nhpp(parametros, Tsimul):
    numeros = list()
    # max λ(t)= λ*t**(β)
    l = parametros[1]*Tsimul**(parametros[0])
    t = 0
    while t<Tsimul:
        u = numpy.random.uniform(0, 1)
        #gerar tempos candidados a NHPP
        t = t - (ln(u) / l)
        # a função taxa de falha λ(t)=λ β t^(β-1)
        fun = parametros[1] * parametros[0] * t ** (parametros[0] - 1)
        #verificar se os tempos são aceites
        if u <= fun / l:
            numeros.append(t)
    return numeros
```

Figura 30 - Função python para obter amostras de NHPP governado pelo modelo lei de potência [Autor]

```
def nhpp2(parametros, Tsimul):
    numeros = list()
    # max λ(t)= (e^(a0)/(a1))*(e^(a1*t)-1)
    l =((math.exp(parametros[1])/parametros[0]))*(math.exp(parametros[0]*Tsimul)-1)
    t = 0
    while t<Tsimul:
        u = numpy.random.uniform(0, 1)
        # gerar tempos candidados a NHPP
        t = t - (ln(u) / l)
        # a função taxa de falha λ(t)= e^(a0+a1*t)
        fun =math.exp(parametros[1]+parametros[0]*t)
        #verificar se os tempos são aceites
        if u <= fun / l:
            numeros.append(t)
    return numeros
```

Figura 31 - Função python para gerar NHPP governado pelo modelo Cox-Lewis [Autor]

As figuras abaixo mostram 3 histogramas de amostras limitadas pelo tempo máximo de simulação de 1000 u.t, geradas pelo NHPP governado pela modelo de lei potência para diferentes valores β:

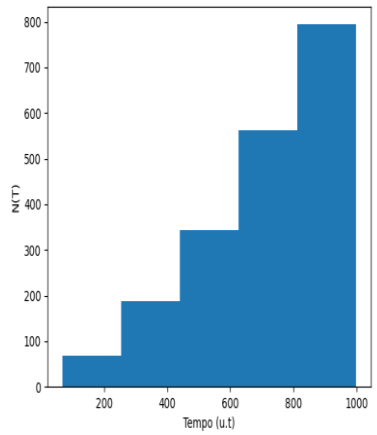

Figura 32 - Histograma de número esperado avarias em função do tempo para β>1 [Autor]

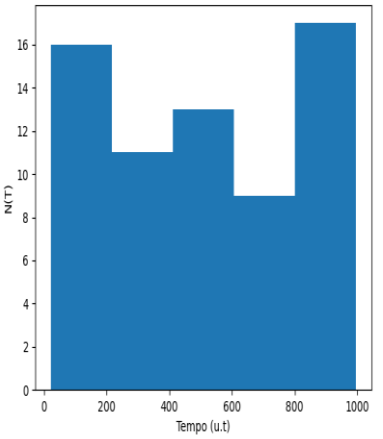

Figura 33 - Histograma de número esperado avarias em função do tempo para β=1 [Autor]

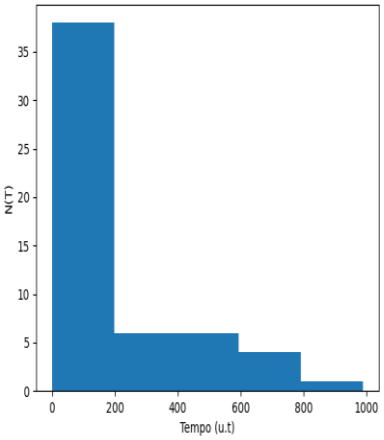

Figura 34 - Histograma de número esperado avarias em função do tempo para β<1 [Autor]

Tal como era expectável, a Figura 32, que apresenta β=2,5, revela uma taxa de ocorrências crescente, já a Figura 33 em que β=1,0 apresenta uma taxa de avarias aproximadamente constante, e, por fim, a Figura 34 tem como β=0,2 apresenta uma taxa de ocorrências decrescente.

Relativamente ao modelo de Cox-Lewis, foi gerada também uma amostra com tempo máximo de simulação de 22000 u.t., para um caso em que $\alpha_1 > 1$ e verificou-se, como era expetável, que a taxa de avarias é crescente ao longo do tempo (Figura 35):

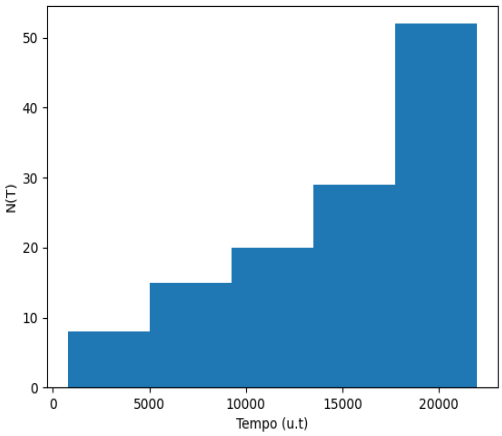

Figura 35 - Histograma de número avarias em função do tempo para α_1>1 [Autor]

3.5 Tempos de reparação

Relativamente à modelação dos tempos de reparação, foi admitido que estes ocorrem de forma aleatória no tempo, pelo que é razoável modelá-los por via de uma distribuição de probabilidade. No ajuste da distribuição de probabilidade que traduz adequadamente os tempos de reparação, D_i, recorreu-se à função `funçãomodelo3` que foi desenvolvida e utilizada anteriormente no âmbito do processo de renovação. Uma vez identificada a distribuição para os tempos de reparação e a estimação dos respetivos parâmetros, facilmente se consegue simular amostras de tempos de reparação. A Figura 36 seguinte ilustra a função densidade de probabilidade estimada a partir de uma amostra gerada a partir da distribuição normal de valor médio 3 e variância 1.

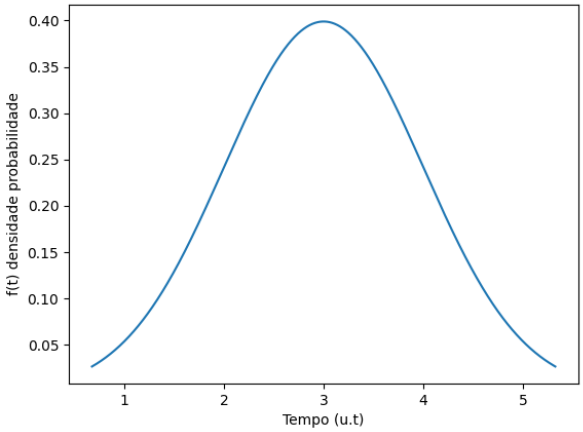

Figura 36 - Simulação da distribuição normal para μ=3 e σ=1 [Autor]

Uma outra suposição importante feita diz respeito à independência entre os tempos entre avarias e os tempos de reparação.

3.6 Ferramenta computacional

Uma vez finalizada toda a estrutura lógica subjacente à ferramenta computacional, é necessário criar uma interface gráfica (*dashboard*) do utilizador. Esta interface permite uma utilização mais amigável da ferramenta pois funciona por meio de elementos visuais em contraste com a interface da linha de comando.

Para a criação do *dashboard* recorreu-se a uma biblioteca python designada de `Tkinter`. O *dashboard* desenvolvido exige que o utilizador preencha alguns campos tais como: o nome do ficheiro Excel com os dados, o nome da folha desse mesmo ficheiro, o tempo de observação, o nível de significância a usar nos testes de hipóteses, o tempo de simulação e o tempo (t) para o qual será calculada a disponibilidade média. Por sua vez a ferramenta disponibiliza os seguintes indicadores: MTTR, MTBF, disponibilidade instantânea, disponibilidade média $[0,T_{simul}]$, disponibilidade média

[0, t]. São ainda mostradas informações sobre o processo utilizado, assim como sobre a distribuição associada aos tempos de reparação. Por fim são ainda apresentados 2 gráficos, o primeiro diz respeito à variação do número esperado de avarias em função do tempo, enquanto o segundo mostra a função densidade de probabilidade associada aos tempos de reparação. A Figura 37 abaixo representa o *dashboard*:

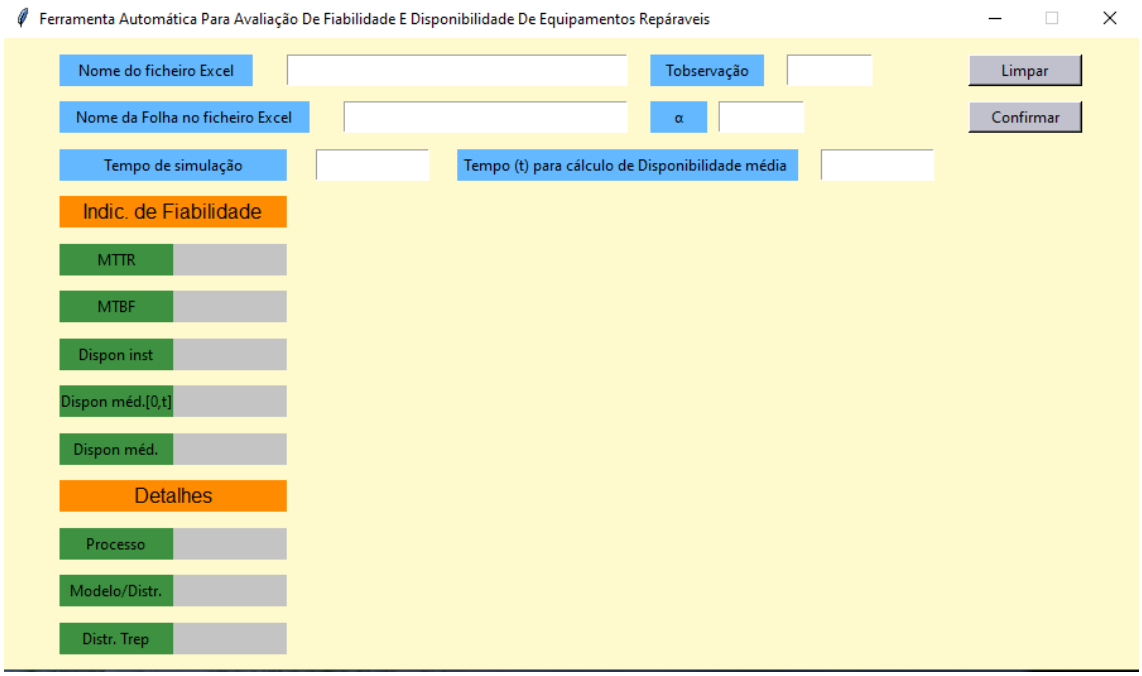

Figura 37 - dashboard da ferramenta computacional [Autor]

É importante notar que todas as funções desenvolvidas em python tiveram de ser articuladas de forma que a ferramenta funcione de forma automática.

VALIDAÇÃO DA FERRAMENTA

4.1 Teste com dados simulados

4.2 Teste com dados reais

4 Validação da Ferramenta

Após a conclusão da ferramenta computacional, é tempo de testá-la de forma a ter uma perceção sobre o desempenho da mesma em diferentes cenários. Esta é, então, a secção de teste da ferramenta, que se circunscreve a aplicações tanto com dados simulados, como com dados reais.

4.1 Teste com dados simulados

Nesta subsecção serão simuladas várias amostras de dados e, posteriormente, será feita uma análise dos *outputs* emitidos pela ferramenta de forma a avaliar o seu desempenho.

4.1.1 Simulação de amostra a partir de HPP

Para simular uma amostra de tempos de ocorrências cujo processo de ocorrência pode ser modelado por um HPP, recorreu-se à função python, apresentada anteriormente, que simula realizações desse mesmo processo. A função tem como parâmetros de entrada a de taxa de falha (λ) e o tempo de simulação ($T_{simulação}$). A função retorna os tempos de N avarias. A Tabela 3 indica os detalhes da amostra que se simulou:

Tabela 3 - Detalhes da amostra para gerar T_i através de um HPP

Parâmetros	Valor
λ	0,03 avaria/u.t. *(input)*
$T_{simulação}$	1610 u.t. *(input)*
N	56 *(output)*

Relativamente aos tempos de reparação, optou-se por simular os mesmos com base numa distribuição lognormal, os parâmetros da mesma encontram-se na Tabela 4:

Tabela 4 - Parâmetros da distribuição lognormal para gerar amostra de D_i

Parâmetros	Valor
Média	3 u.t.
Desvio Padrão	0.006 u.t.
N	56

Por fim, no que diz respeito aos dados a introduzir na ferramenta, foram os seguintes (Tabela 5):

Tabela 5 - *Inputs* amostra HPP

Parâmetros	Valor
$T_{observação}$	1610 u.t.
α	0.1
$T_{simulação}$	5000 u.t.
t	2000 u.t.

Uma vez introduzidos todos os *inputs* necessários na ferramenta, realizaram-se várias simulações. Note-se que de simulação para simulação os resultados variam ligeiramente entre si, mas essa diferença é pequena. Os intervalos de resultados emitidos pela ferramenta apresentam-se abaixo (Tabela 6):

Tabela 6 - Resultados para amostra HPP

Parâmetros	Valores aproximados
MTTR	2,9 a 3,1 u.t
MTBF	24 a 27,1 u.t
Disp instan.	88,3 a 90,2 %
Disp méd [0, t]	88,4 a 90,6 %
Disp méd [0, T]	89,1 a 91,2 %

A Figura 38 mostra o exemplo de uma das simulações que foi realizada onde é possível ver que os resultados vão de encontro aos da tabela acima, assim como, os gráficos do número de avarias e da distribuição dos tempos de reparação.

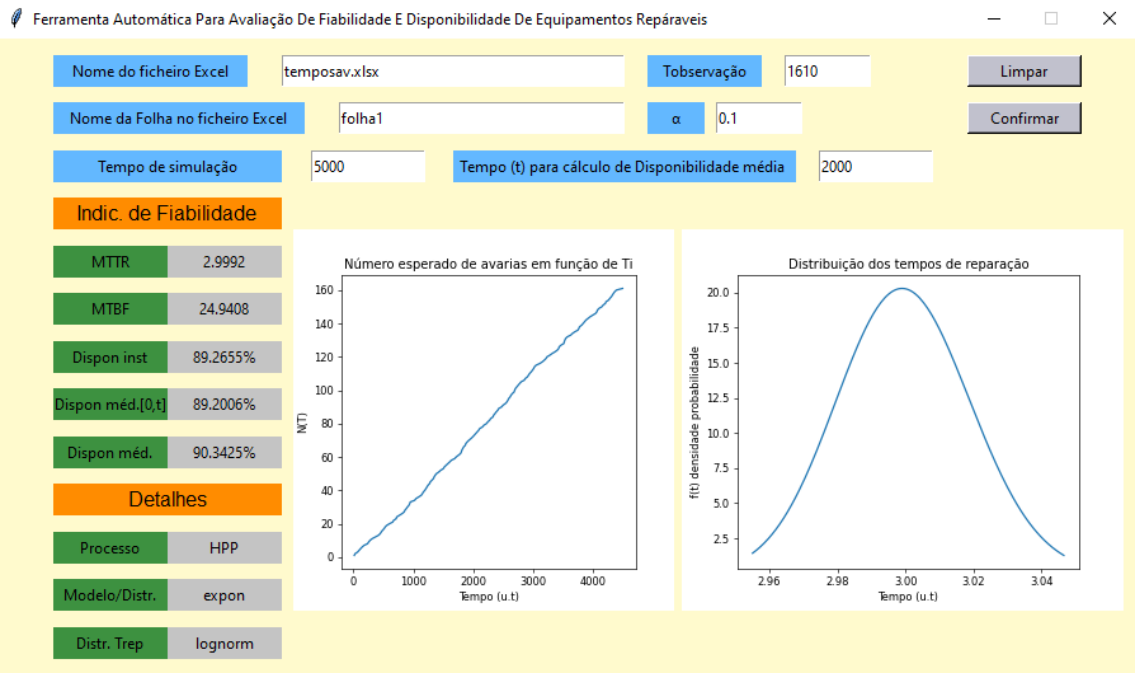

Figura 38 -Resultados no dashboard para a amostra HPP [Autor]

Pela análise do gráfico do número de avarias, verifica-se que o declive da reta se mantém praticamente constante o que significa que não existe variação da tendência, portanto as ocorrências ocorrem de forma aleatória tal como é expectável num Processo homogéneo de Poisson. De acordo com a pesquisa bibliográfica efetuada, no HPP admite-se que o sistema não se degrada com o tempo, ou seja, quando um sistema é reparado fica como novo (modelo de reparação perfeita), sendo, portanto, natural que os valores entre ocorrências apresentem um comportamento sem tendência ao longo do tempo.

Nesse sentido ao comparar a disponibilidade média num intervalo de tempo correspondente ao período de início de vida, com a disponibilidade média correspondente ao período total de vida verifica-se que os valores são muito aproximados. É importante ainda referir que a distribuição dos tempos de reparação tem desvio padrão baixo (0,006), portanto a variância dos mesmos é pequena. Isto, significa que os tempos de reparação no período inicial de vida são semelhantes aos tempos de reparação no fim de vida, sendo então este outro fator que contribui para a homogeneidade da disponibilidade média. Relativamente ao resultado do MTTR já seria de esperar um valor próximo de 3 uma vez que está em concordância com a simulação feita para os tempos de reparação (distribuição lognormal de média 3).

A justificação para a obtenção de disponibilidades na ordem dos 90% deve-se ao facto em média o sistema funcionar durante 26 u.t e ter um tempo de paragem de 3 horas (1 - 3/26 ≈90%).

4.1.2 Simulação de amostra a partir de PR

Nesta subsecção foi obtida uma amostra a partir de processos de renovação, em que os tempos entre ocorrências da mesma seguem uma distribuição de Weibull. A Tabela 7 indica os detalhes da amostra.

Tabela 7 - Detalhes da amostra para gerar T_i através de um PR

Parâmetros	Valor
Forma	1,5
Localização	40
$T_{simulação}$	3000 u.t
N	75

Em relação aos tempos de reparação, utilizou-se uma simulação que segue a distribuição Beta com as seguintes características (Tabela 8):

Tabela 8 - Parâmetros da distribuição Beta para gerar amostra de D_i

Parâmetros	Valor
Forma1 (α)	4
Forma2 (β)	4
N	75

Por sua vez na Tabela 9, estão presentes todos os dados necessários a inserir no *dashboard:*

Tabela 9 - inputs amostra PR

Parâmetros	Valor
$T_{observação}$	3000 u.t.
α	0.01
$T_{simulação}$	5000 u.t.
t	1000 u.t.

Os resultados obtidos encontram-se a seguir (Tabela 10):

Tabela 10 - Resultados para amostra PR

Parâmetros	Valores aproximados
MTTR	0,47 a 0,52 u.t.
MTBF	38,90 a 39,1 u.t.
Disp instan.	98,63 a 98,91%
Disp méd [0, t]	98,64 a 98,93%
Disp méd [0, T]	98,66 a 98,82%

A Figura 39 ilustra exemplos de valores obtidos no *dashboard* para uma simulação realizada, assim como os respetivos gráficos associados:

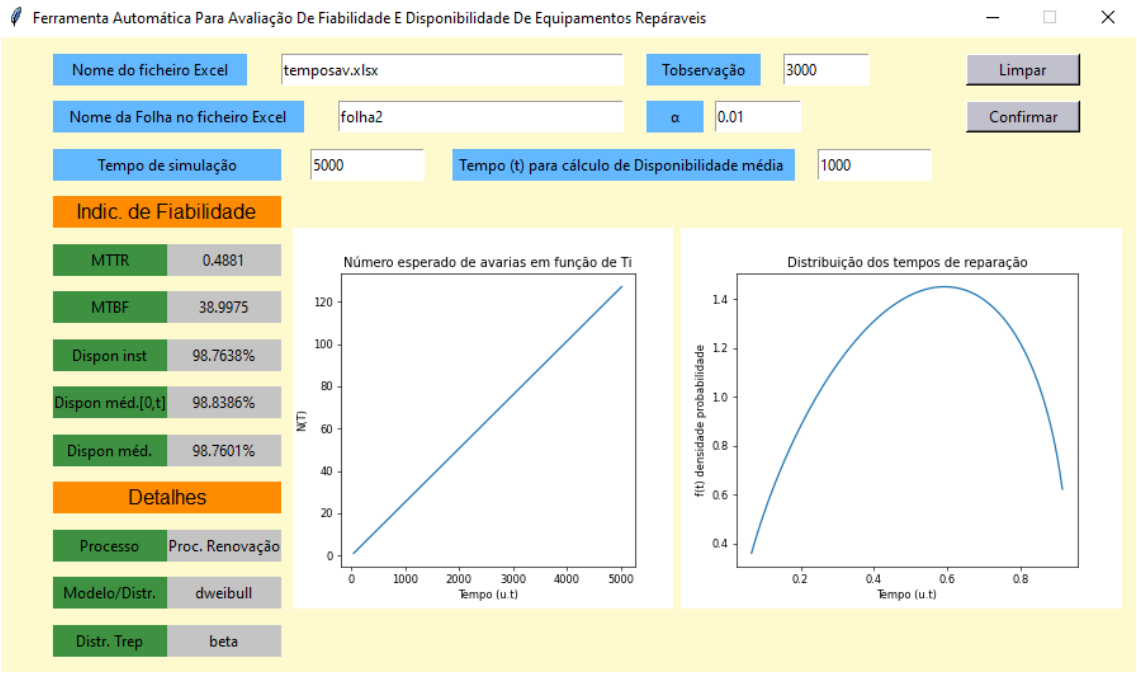

Figura 39 - Resultados no dashboard para a amostra PR [Autor]

De forma semelhante ao HPP, num processo de renovação os tempos de ocorrências são independentes, portanto assume-se novamente que o sistema não se degrada com o tempo. Relativamente ao gráfico do número esperado de avarias, facilmente se percebe que este apresenta um declive constante indicando que não existe tendência na taxa de aparecimento de avarias. Dessa forma, é expectável que os valores de disponibilidades no período inicial de vida do sistema [0, t] sejam semelhantes aos valores da disponibilidade média total [0, T] (considerando que os tempos de reparação não tem grande variação ao longo do tempo). De acordo com os resultados obtidos, verifica-se que ao comparar as disponibilidades estas não variam muito entre si (andam na casa dos 98%) tal como seria de esperar. A explicação para os valores das disponibilidades serem bastante elevados deve-se ao facto de os temos de paragem

(0.5 u.t. em média) serem muito inferiores quando comparados aos tempos de funcionamento (39 u.t em média).
No que diz respeito ao valor medio dos tempos de reparações, espera-se um valor próximo de 0.5 uma vez que a média da distribuição beta é dada por: $\mu = \frac{\alpha}{\alpha+\beta}$.

4.1.3 Simulação de amostra a partir de NHPP para β>1

Nesta fase é simulada uma amostra NHPP em que a função taxa de falha é regida pelo modelo Lei de potência, para tal recorreu-se à função Python desenvolvida neste trabalho que realiza esse mesmo efeito.

A Tabela 11 apresenta todos os dados relevantes constituintes dessa mesma amostra:

Tabela 11 - Detalhes da amostra para gerar T_i através de um NHPP, em que β>1

Parâmetros	Valor
Forma	1,7
Escala	0,000064
$T_{simulação}$	3000 u.t.
N	46

A amostra relativa aos tempos de reparação, foi simulada com base numa distribuição normal com as seguintes características (Tabela 12):

Tabela 12 - Parâmetros da distribuição Normal para gerar amostra de D_i

Parâmetros	Valor
Média	5 u.t.
Desvio padrão	1 u.t.
N	46

Antes da obtenção dos resultados é necessário inserir na ferramenta os *inputs*, que neste caso foram (Tabela 13):

Tabela 13 - Inputs amostra NHPP, para β>1

Parâmetros	Valor
$T_{observação}$	3000 u.t
α	0.1
$T_{simulação}$	5000 u.t
t	500 u.t

Por sua vez, a Tabela 14 apresenta os resultados obtidos:

Tabela 14 - Resultados para amostra NHPP, para β>1

Parâmetros	Valores aproximados
MTTR	4,7 a 5,1 u.t.
MTBF	46,2 a 51,3 u.t.
Disp instan.	90,2 a 91,7%
Disp méd [0, t]	95,1 a 97,9%
Disp méd [0, T]	90,8 a 92,3%

Na Figura 40 encontram-se representados os gráficos e os indicadores emitidos no *dasboard* referentes as amostras em análise.

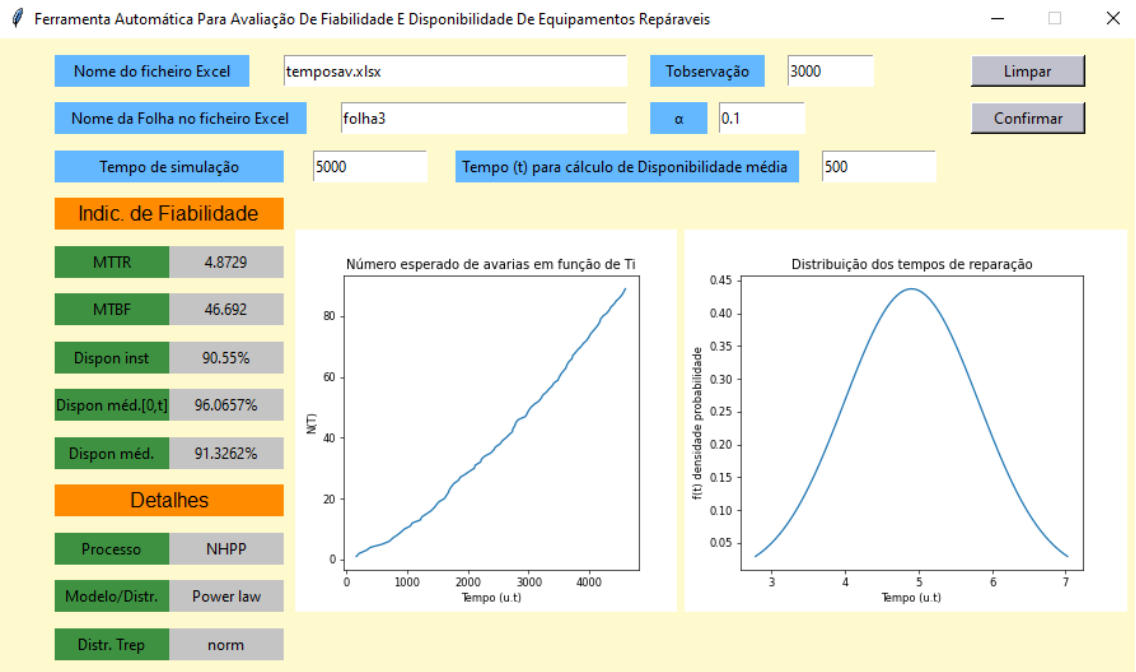

Figura 40 - Resultados no dashboard para a amostra NHPP, para β>1 [Autor]

Pela análise do gráfico do número esperado de avarias facilmente se verifica que existe uma alteração do declive do mesmo. Na fase inicial o declive é mais reduzido o que remete para uma taxa de aparecimento de avarias mais lenta, enquanto na fase final o declive é mais acentuado indicando um surgimento mais rápido de avarias. Isto vai de encontro ao princípio do NHPP regido pelo modelo lei de potência para β>1, que assume que o sistema se degrada ao longo do tempo (dependência temporal). Posto isto, é espectável que na fase inicial de vida do sistema em analise, a disponibilidade seja superior à disponibilidade média total (mais uma vez, considerando que os tempos de reparação não tem grande variância ao longo do período de vida em estudo). Nesse sentido obteve-se resultados na ordem dos 96% para a disponibilidade no período inicial de vida, enquanto a disponibilidade média foi na ordem dos 91%. A ordem de grandeza destes valores de disponibilidade deve-se ao facto de o tempo de

funcionamento (cerca de 48 u.t em média) ser bastante superior ao tempo de paragem (cerca de 5 u.t). Relativamente ao MTTR é natural dar um valor próximo de 5 uma vez que se simulou uma distribuição normal de media = 5, com um desvio padrão reduzido (1).

4.1.4 Simulação de amostra a partir de NHPP para β<1

Por fim, de forma idêntica á simulação anterior, obteve-se de novo uma amostra NHPP, mas desta vez com um parâmetro de forma inferior a 1. A Tabela 15 descreve as informações relevantes constituintes da amostra obtida:

Tabela 15 - Detalhes da amostra para gerar T_i através de um NHPP, em que β<1

Parâmetros	Valor
Forma	0,2
Escala	5,7
$T_{simulação}$	3000 u.t
N	31

Já o que diz respeito aos tempos de reparação desta vez considerou-se uma simulação com base numa distribuição exponencial com as seguintes características (Tabela 16):

Tabela 16 - Parâmetros da distribuição exponencial para gerar amostra de D_i

Parâmetros	Valor
λ	0,22
N	31

Por sua vez, os dados inseridos na ferramenta foram os seguintes (Tabela 17):

Tabela 17 – Inputs amostra NHPP, para β<1

Parâmetros	Valor
$T_{observação}$	3000 u.t
α	0.05
$T_{simulação}$	10000 u.t
t	500 u.t

A Tabela 18 mostra os resultados obtidos:

Tabela 18 - Resultados para amostra NHPP, para β<1

Parâmetros	Valores Aprox.
MTTR	3.8 a 4,9 u.t
MTBF	136 a 181 u.t
Disp instan.	96,5 a 97,9 %
Disp méd [0, t]	84,1 a 92,3 %
Disp méd [0, T]	97,6 a 98,5 %

A ilustração dos resultados no *dasboard,* para um teste efetuado, pode ser consultada na Figura 41:

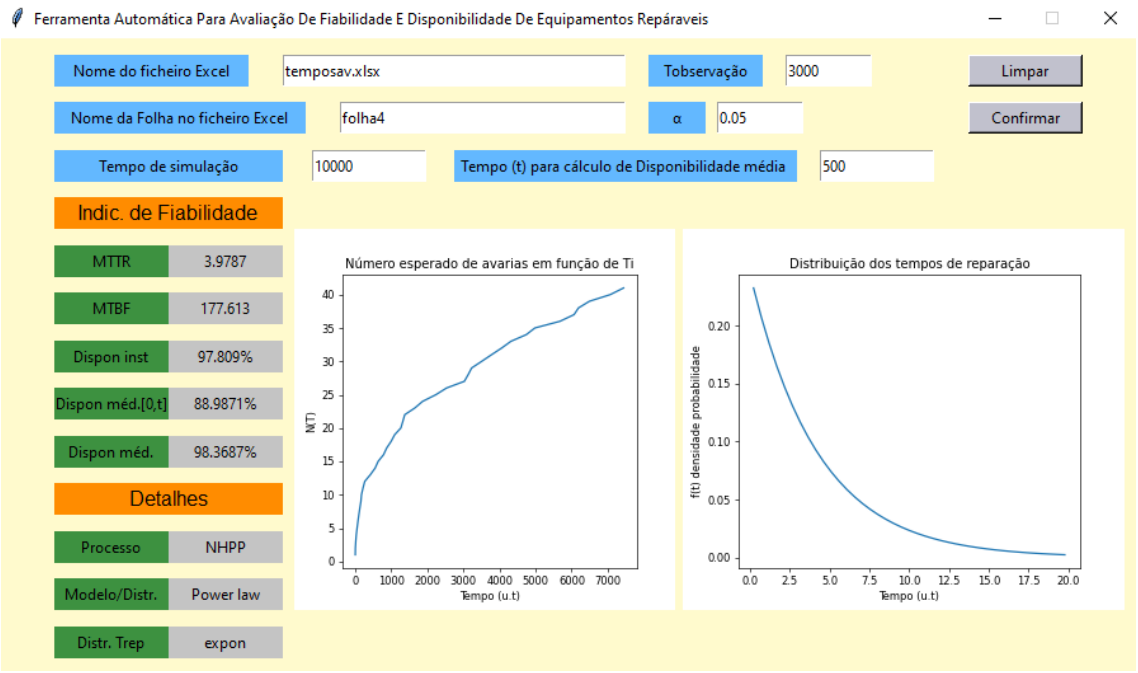

Figura 41 - Resultados no dashboard para a amostra NHPP, para β<1 [Autor]

Em contraste com a simulação anterior (caso em que: β>1), aqui o gráfico do número esperado de avarias apresenta um declive bastante acentuado na fase inicial do período de vida (correspondente a uma taxa de aparecimento de avarias maior) enquanto na parte final este já é menor, indicando, portanto, um aparecimento de avarias mais lento. É de recordar que estas situações são comuns no período inicial da curva da banheira, o chamado período de mortalidade infantil. Perante isto, é espectável que na fase inicial de vida do sistema em analise, a disponibilidade seja inferior à disponibilidade média total (considerando que os tempos de reparação não tem grande variância ao longo do período de vida em estudo). Nesse sentido os resultados obtidos para as disponibilidades estão em concordância com o que é suposto. Note -se que em média por cada 150 u.t de funcionamento do sistema este tem um período de paragem de 4.5 u.t., resultando, portanto, num valor elevado de disponibilidade média.

Em relação ao tempo medio de reparação, é esperado um valor próximo de 4.5 uma vez que a média da distribuição exponencial é obtida por: $\mu = \frac{1}{0.22} \approx 4{,}5$.

4.2 Teste com dados reais

Após se verificar que a ferramenta apresenta resultados fiáveis para dados simulados, procedeu-se ao teste da mesma utilizando dados reais. Dessa forma, recorreu-se ao histórico de avarias de um sistema simples constituído apenas por dois componentes A e B (Disponível em: https://reliawiki.org/index.php/Event_Log_Data). Para obter o mesmo, um engenheiro começou a gravar os eventos em 1 de janeiro de 1997 às 12h e parou de gravar em 18 de março de 1997 às 13h. De referir ainda que, o sistema em estudo, encontra-se em funcionamento das 8h às 17 h (9 horas por dia).
O objetivo desta análise é obter as distribuições de falhas e de tempos de reparação mais adequadas apenas para o componente A (o processo para analisar o componente B seria exatamente o mesmo, não havendo, portanto, necessidade de repetir duas vezes o mesmo raciocínio). A Tabela 19 abaixo apresenta o histórico relativamente ao componente A:

Tabela 19 - Histórico de avarias de um componente real

#	X_i (h)	D_i (h)
1	13	3,8167
2	78,43333	3,333
3	8,93333	0,4167
4	56,25	29,6167
5	33,05	0,4833
6	100,483	4,5167
7	35,7	17,2833
8	112,3167	0,4833
9	23,1	0,4500
10	13,96667	5,5000
11	90,51667	0,4667
12	33,6	-

Uma vez que a ferramenta computacional foi concebida para ler dados de tempos de ocorrências em vez dos tempos entre ocorrências, é necessário transformar os dados recolhidos acima. Nesse sentido, facilmente, se obteve a Tabela 20:

Tabela 20 - Dados reais a introduzir na ferramenta

#	T_i(h)	D_i(h)
1	13	3,8167
2	95,25003	3,333
3	107,5167	0,4167
4	164,1834	29,6167
5	226,8501	0,4833
6	327,8167	4,5167
7	368,0334	17,2833
8	497,6334	0,4833
9	521,2167	0,4500
10	535,6333	5,5000
11	631,65	0,4667
12	665,7167	-

De seguida, é necessário introduzir os parâmetros necessários na ferramenta para que esta consiga emitir os resultados. De referir que o $T_{observação}$ é um parâmetro que tem de se calcular, uma vez que se conhece o intervalo de tempo em que o engenheiro analisou o equipamento (1 de janeiro de 1997 às 12h até 18 de março de 1997 às 13h). O valor deste é obtido então por:

$$T_{observação} = 5 + 30 \times 9 + 28 \times 9 + 17 \times 9 + 5 = 685h \quad \textbf{(60)}$$

Nota: A expressão representa a soma das 5 horas observadas no dia 1 de janeiro com as 9 horas diárias dos restantes 30 dias de janeiro, com as 9 horas diárias dos 28 dias de fevereiro, com as 9 horas dos primeiros 17 dias de março e por fim com as 5 horas observadas no dia 18 de março.

Os restantes parâmetros adotados encontram-se na Tabela 21:

Tabela 21 – Parâmetros para analisar caso real

Parâmetros	Valor
$T_{observação}$	685 h
α	0.05
$T_{simulação}$	3000 h
t	1000 h

A Tabela 22 abaixo mostra os resultados obtidos após introduzir os dados na ferramenta:

Tabela 22 - Resultados obtidos para caso real

Parâmetros	Valores Aprox.
MTTR	2,72 a 3,81 h
MTBF	33,42 a 39,22 h
Disp instan.	90,1 a 92,7 %
Disp méd [0, t]	90,06 a 94,88 %
Disp méd [0, T]	91,68 a 98,5 %

A ilustração dos resultados no dasboard, para um teste efetuado, pode ser consultada na Figura 42:

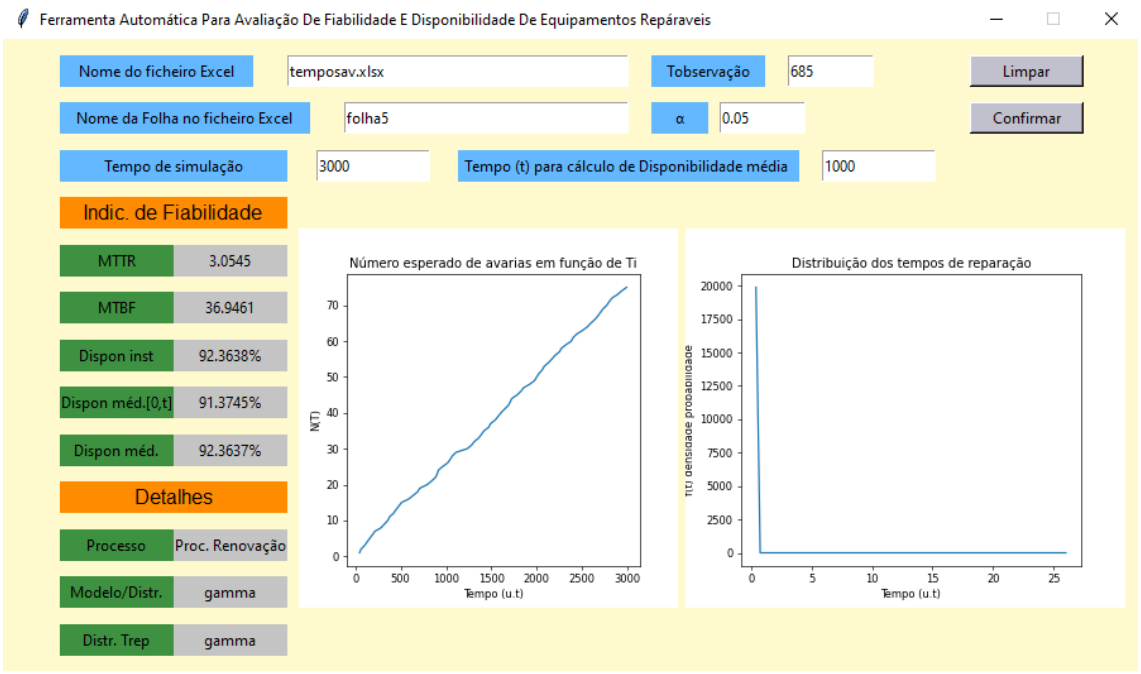

Figura 42 – Resultados no Dashboard para caso real [Autor]

Pela análise facilmente se percebe que os valores obtidos para a disponibilidade estão ordem dos 90%-95%. De acordo com a fonte onde se encontra esse caso de estudo o valor de Uptime do sistema é de 599.35h enquanto o valor de Downtime é de 66,3667h. Isto traduz-se num valor obtido para a disponibilidade esperada na ordem de 90,03% ($Disp = 599{,}35/(599{,}35 + 66{,}3667) = 90{,}03\%$). Ao comparar os valores de disponibilidade obtidos na ferramenta com o que é expetável, verifica-se que existe uma ligeira diferença, mas nada de muito significativo. Essa diferença é explicada devido a dimensão da amostra ser muito reduzida (N=12), uma vez que quanto menor for a dimensão da amostra maior será a margem de erro associada.

Desta forma, pode-se concluir que a ferramenta consegue escolher o modelo mais adequado a um conjunto de dados reais, emitindo com uma boa precisão os valores associados de disponibilidades. Dado que os valores de disponibilidade estão próximos da realidade, será justo assumir também que os valores de MTTR E MTBF também estão realistas (o cálculo da disponibilidade instantânea é dado por $Disp = MTBF/(MTBF + MTTR) \times 100$). Uma vez testada com sucesso em dados reais e simulados, dá-se por encerrada a validação da ferramenta.

CONCLUSÕES

5 CONCLUSÕES E PROPOSTAS DE TRABALHOS FUTUROS

5.1 CONCLUSÕES

A realização deste trabalho permitiu verificar os conceitos gerais associados aos sistemas reparáveis e não reparáveis, assim como identificar importantes indicadores de fiabilidade e de disponibilidade para qualquer sistema reparável. Foram também identificados os diferentes modelos matemáticos/estatísticos adequados à modelação dos diferentes processos geradores de avarias em sistemas reparáveis – processos de renovação, processos de Poisson homogéneos e não homogéneos, distribuições de probabilidade para o caso dos tempos entre avarias que são independentes e igualmente distribuídos. É também realizada uma investigação dos algoritmos relativos a esses processos assim como técnicas de seleção de modelos.

Foi ainda efetuada uma aprendizagem da linguagem Python que permitiu elaborar a ferramenta computacional. Esta tem como função ler os dados do histórico de avarias e de tempos de reparação, fornecendo de seguida vários indicadores de fiabilidade e emitindo gráficos afetos aos dados em analise. Após a construção de todo o modelo lógico da ferramenta assim como o respetivo dasboard, surgiu a necessidade de validação de resultados. Para tal, numa primeira fase foram criadas várias amostras (para tempos de ocorrências e tempos de reparação) através da simulação de dados, de forma a poder controlar o processo. Numa segunda fase já foram recolhidos dados reais de forma a ver o comportamento da ferramenta para dados não controlados. A analise dos resultados foi de encontro ao que era expectável, portanto a ferramenta correspondeu na totalidade, dando-se assim a validação da mesma.

De acordo com a pesquisa efetuada, a manutenção centrada na fiabilidade e manutenção baseada na condição permitem não só antever a ocorrência de avarias, mas também poupar na manutenção corretiva e prolongar a vida útil dos ativos. Nesse contexto foram apresentados vários casos de estudo de implementação destas políticas, visando demonstrar a importância das mesmas e ilustrar as suas vantagens no mundo empresarial.

Conclui-se que a manutenção preditiva tem uma quota-parte importante no bom desempenho dos processos de produção, e que existe uma necessidade cada vez maior dos gestores da manutenção recorrerem a ferramentas automáticas orientadas por dados (data driven) que disponibilizem indicadores de fiabilidade e de disponibilidade para um sistema reparável geral.

5.2 PROPOSTA DE TRABALHOS FUTUROS

Como propostas para trabalhos futuros propõe-se o alargamento das funcionalidades da ferramenta, nomeadamente a emissão de um número maior de indicadores de fiabilidade (por exemplo calcular a probabilidade de o equipamento falhar num determinado intervalo de tempo, etc) assim como a emissão de mais gráficos afetos ao processo (por exemplo gerar um gráfico que mostre a relação entre o tempo de funcionamento e o tempo de paragem do equipamento, etc).

É ainda proposto o alargamento dos modelos que contemplam a estrutura lógica da ferramenta. Por exemplo seria útil adicionar mais distribuições de probabilidade para modelar os tempos de reparação assim como para os tempos entre avarias dos processos de renovação. De igual modo, seria benéfico contemplar mais modelos afetos aos NHPP (neste trabalho utilizou-se apenas o modelo de Crow e de Cox-Lewis).

Numa perspetiva bastante mais ambiciosa, seria interessante alargar a ferramenta para a modelação de dados com base em modelos de reparação imperfeita.

BIBLIOGRAFIA E OUTRAS FONTES DE INFORMAÇÃO

6 BIBLIOGRAFIA E OUTRAS FONTES DE INFORMAÇÃO

Afefy, I. H., Mohib, A., El-Kamash, A. M., & Mahmoud, M. A. (2019). A New Framework of Reliability Centered Maintenance. In *Jordan Journal of Mechanical and Industrial Engineering* (Vol. 13, Issue 3).

Airoboman, A., Ogujor, E. A., Ignatius, O., Airoboman, A. E., & Okakwu, I. K. (2016). On the Centralization of Reliability in Maintenance Practices in the Nigeria Power System Network: A Review Reliability of electric power systems View project Energy Management View project On the Centralization of Reliability in Maintenance Practices in the Nigeria Power System Network: A Review. In *International Journal of Engineering Works Kambohwell Publisher Enterprises* (Vol. 3). www.kwpublisher.com

Akaike, H. (1998). Information theory and an extension of the maximum likelihood principle. In *Selected papers of hirotugu akaike* (pp. 199–213). Springer.

Assis, R. (1997). *Manutenção centrada na fiabilidade: Economia das decisões*.

Azadeh, A., & Abdolhossein Zadeh, S. (2016). An integrated fuzzy analytic hierarchy process and fuzzy multiple-criteria decision-making simulation approach for maintenance policy selection. *SIMULATION*, *92*(1), 3–18. https://doi.org/10.1177/0037549715616686

BahooToroody, A., Abaei, M. M., Arzaghi, E., Song, G., de Carlo, F., Paltrinieri, N., & Abbassi, R. (2020). On reliability challenges of repairable systems using hierarchical bayesian inference and maximum likelihood estimation. *Process Safety and Environmental Protection*, *135*, 157–165. https://doi.org/10.1016/j.psep.2019.11.039

Barabadi, A. (2013). Reliability model selection and validation using Weibull probability plot - A case study. *Electric Power Systems Research*, *101*, 96–101. https://doi.org/10.1016/j.epsr.2013.03.010

Ben-Daya, M., Duffuaa, S. O., Knezevic, J., Ait-Kadi, D., & Raouf, A. (2009). Handbook of maintenance management and engineering. In *Handbook of Maintenance Management and Engineering*. Springer London. https://doi.org/10.1007/978-1-84882-472-0

Bozdogan, H. (1987). Model selection and Akaike's information criterion (AIC): The general theory and its analytical extensions. *Psychometrika*, *52*(3), 345–370.

Burnecki, K., & Weron, R. (2010). *Simulation of risk processes*.

Carlo, F. de, & Arleo, M. A. (2017). Imperfect Maintenance Models, from Theory to Practice. In *System Reliability*. InTech. https://doi.org/10.5772/intechopen.69286

Chen, Y. (2016). *Thinning Algorithms for Simulating Point Processes*.

Choe, Y., Byon, E., & Chen, N. (2015). Importance Sampling for Reliability Evaluation With Stochastic Simulation Models. *Technometrics*, *57*(3), 351–361. https://doi.org/10.1080/00401706.2014.1001523

Das, K. (2008). A comparative study of exponential distribution vs Weibull distribution in machine reliability analysis in a CMS design. *Computers and Industrial Engineering*, *54*(1), 12–33. https://doi.org/10.1016/j.cie.2007.06.030

de Faria, H., Costa, J. G. S., & Olivas, J. L. M. (2015). A review of monitoring methods for predictive maintenance of electric power transformers based on dissolved gas analysis. In *Renewable and Sustainable Energy Reviews* (Vol. 46, pp. 201–209). Elsevier Ltd. https://doi.org/10.1016/j.rser.2015.02.052

Duan, C., Li, Z., & Liu, F. (2020). Condition-based maintenance for ship pumps subject to competing risks under stochastic maintenance quality. *Ocean Engineering*, *218*. https://doi.org/10.1016/j.oceaneng.2020.108180

Ecker, J. G., & Kupferschmid, M. (1988). *Introduction to operations research*. Wiley New York.

Eriksen, S., Utne, I. B., & Lützen, M. (2021). An RCM approach for assessing reliability challenges and maintenance needs of unmanned cargo ships. *Reliability Engineering and System Safety*, *210*. https://doi.org/10.1016/j.ress.2021.107550

Finkelstein, M. (2015). On the optimal degree of imperfect repair. *Reliability Engineering and System Safety*, *138*, 54–58. https://doi.org/10.1016/j.ress.2015.01.010

Fraga, L. G., & Pinto, E. R. (n.d.). O Processo DE WEIBULL POISSON PARA A MODELAGEM DA CONFIABILIDADE EM SISTE-MAS REPARÁVEIS. *Revista Matemática e Estatística Em Foco–ISSN*, *2318*, 0552.

Gámiz, M. L., & Lindqvist, B. H. (2016). Nonparametric estimation in trend-renewal processes. *Reliability Engineering and System Safety*, *145*, 38–46. https://doi.org/10.1016/j.ress.2015.08.015

Gįmiz, M. L., Kulasekera, K. B., Limnios, N., & Lindqvist, B. H. (2011). *Applied nonparametric statistics in reliability*. Springer Science & Business Media.

Goyal, D., & Pabla, B. S. (2015). Condition based maintenance of machine tools-A review. In *CIRP Journal of Manufacturing Science and Technology* (Vol. 10, pp. 24–35). Elsevier Ltd. https://doi.org/10.1016/j.cirpj.2015.05.004

Kahle, W. (2007). Optimal maintenance policies in incomplete repair models. *Reliability Engineering and System Safety*, *92*(5), 563–565. https://doi.org/10.1016/j.ress.2006.05.004

Kang, S., Khanjari, A., You, S., & Lee, J.-H. (2021). Comparison of different statistical methods used to estimate Weibull parameters for wind speed contribution in nearby an offshore site, Republic of Korea. *Energy Reports*, *7*, 7358–7373. https://doi.org/10.1016/j.egyr.2021.10.078

Katukoori, V. K. (1995). Standardizing availability definition. *University of New Orleans, New Orleans, La., USA*.

Lindqvist, B. H. (1999). Statistical Modeling and Analysis of Repairable Systems. In *Statistical and Probabilistic Models in Reliability* (pp. 3–25). Birkhäuser Boston. https://doi.org/10.1007/978-1-4612-1782-4_1

Lindqvist, B. H., Elvebakk, G., & Heggland, K. (2003). The trend-renewal process for statistical analysis of repairable systems. *Technometrics*, *45*(1), 31–44. https://doi.org/10.1198/004017002188618671

Littell, R. C., Milliken, G. A., Stroup, W. W., Wolfinger, R. D., & Schabenberger, O. (1996). *SAS system for mixed models* (Vol. 633). SAS institute Cary, NC.

Liu, Q., Ma, L., Wang, N., Chen, A., & Jiang, Q. (2022). A condition-based maintenance model considering multiple maintenance effects on the dependent failure processes. *Reliability Engineering & System Safety*, *220*, 108267. https://doi.org/10.1016/j.ress.2021.108267

Liu, X., Finkelstein, M., Vatn, J., & Dijoux, Y. (2020). Steady-state imperfect repair models. *European Journal of Operational Research*, *286*(2), 538–546. https://doi.org/10.1016/j.ejor.2020.03.057

Ma, X., Liu, B., Yang, L., Peng, R., & Zhang, X. (2020). Reliability analysis and condition-based maintenance optimization for a warm standby cooling system. *Reliability Engineering and System Safety*, *193*. https://doi.org/10.1016/j.ress.2019.106588

Mahadevan, S., & Raghothamachar, P. (1999). *Adaptive simulation for system reliability analysis of large structures*. www.elsevier.com/locate/compstruc

Menezes, N. N. C. (2010). Introdução a programação com Python. *São Paulo: Novatec*.

Morad, A. M., Pourgol-Mohammad, M., & Sattarvand, J. (2014). Application of reliability-centered maintenance for productivity improvement of open pit mining equipment: Case study of Sungun Copper Mine. *Journal of Central South University*, *21*(6), 2372–2382. https://doi.org/10.1007/s11771-014-2190-2

Nafidi, A., Bahij, M., Achchab, B., & Gutiérrez-Sanchez, R. (2019). The stochastic Weibull diffusion process: Computational aspects and simulation. *Applied Mathematics and Computation*, *348*, 575–587. https://doi.org/10.1016/j.amc.2018.12.017

Nafisah, I., Shrahili, M., Alotaibi, N., & Scarf, P. (2019). Virtual series-system models of imperfect repair. *Reliability Engineering and System Safety*, *188*, 604–613. https://doi.org/10.1016/j.ress.2019.03.046

Nejad, A. R., Gao, Z., & Moan, T. (2014). Fatigue reliability-based inspection and maintenance planning of gearbox components in wind turbine drivetrains. *Energy Procedia*, *53*(C), 248–257. https://doi.org/10.1016/j.egypro.2014.07.234

Nguyen, D. T., Dijoux, Y., & Fouladirad, M. (2017). Analytical properties of an imperfect repair model and application in preventive maintenance scheduling. *European Journal of Operational Research*, *256*(2), 439–453. https://doi.org/10.1016/j.ejor.2016.06.026

Niu, G., Yang, B. S., & Pecht, M. (2010). Development of an optimized condition-based maintenance system by data fusion and reliability-centered maintenance. *Reliability Engineering and System Safety*, *95*(7), 786–796. https://doi.org/10.1016/j.ress.2010.02.016

Pasupathy, R. (2011). *Generating Nonhomogeneous Poisson Processes*. https://filebox.vt.edu/users/pasupath/pasupath.htm

Pham, H. (2005). *System Software Reliability (Springer Series in Reliability Engineering)*. Springer-Verlag.

Rahimdel, M. J., Ataei, M., Khalokakaei, R., & Hoseinie, S. H. (2013). Reliability-based maintenance scheduling of hydraulic system of rotary drilling machines. *International Journal of Mining Science and Technology*, *23*(5), 771–775. https://doi.org/10.1016/j.ijmst.2013.08.023

Ramos, S. (2021). *DISPOSITIVOS PARA AS AULAS TEÓRICAS DA UNIDADE CURRICULAR FIABILIDADE E MANUTENÇÃO*.

Rausand, M., & Hoyland, A. (2004). *System Reliability Theory: Models, Statistical Methods and Applications* (Second Edition). John Wiley & Sons.

Sajaradj, Z., Huda, L. N., & Sinulingga, S. (2019). The Application of Reliability Centered Maintenance (RCM) Methods to Design Maintenance System in Manufacturing (Journal Review). *IOP Conference Series: Materials Science and Engineering*, *505*(1). https://doi.org/10.1088/1757-899X/505/1/012058

Sandrini, M. (2017). *AVALIAÇÃO DE TENDÊNCIA DA FIABILIDADE DE SISTEMAS*.

Santos, A., & Brito, A. (2003). *MODELO DE FIABILIDADE APLICADO AO HISTÓRICO DE AVARIAS – TRATAMENTO ANALÍTICO*.

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, 461–464.

Smith, D. J. (2017). *Reliability, maintainability and risk: practical methods for engineers*. Butterworth-Heinemann.

Taboada, J. v., Diaz-Casas, V., & Yu, X. (2021). CBM challenges and opportunities for O&M of the Johan Sverdrup Oil and Gas Field. *Journal of Petroleum Science and Engineering*, *205*. https://doi.org/10.1016/j.petrol.2021.108890

Tanwar, M., Rai, R. N., & Bolia, N. (2014). Imperfect repair modeling using Kijima type generalized renewal process. In *Reliability Engineering and System Safety* (Vol. 124, pp. 24–31). Elsevier Ltd. https://doi.org/10.1016/j.ress.2013.10.007

Teixeira, H. N., Lopes, I., & Braga, A. C. (2020). Condition-based maintenance implementation: A literature review. *Procedia Manufacturing*, *51*, 228–235. https://doi.org/10.1016/j.promfg.2020.10.033

Wan, A., Gu, F., Chen, J., Zheng, L., Hall, P., Ji, Y., & Gu, X. (2018). Prognostics of gas turbine: A condition-based maintenance approach based on multi-environmental time similarity. *Mechanical Systems and Signal Processing*, *109*, 150–165. https://doi.org/10.1016/j.ymssp.2018.02.027

Wang, W., Hussin, B., & Jefferis, T. (2012). A case study of condition based maintenance modelling based upon the oil analysis data of marine diesel engines using stochastic filtering. *International Journal of Production Economics*, *136*(1), 84–92. https://doi.org/10.1016/j.ijpe.2011.09.016

Wolfinger, R. (1993). Covariance structure selection in general mixed models. *Communications in Statistics-Simulation and Computation*, *22*(4), 1079–1106.

Yang, D., He, Z., & He, S. (2016). Warranty claims forecasting based on a general imperfect repair model considering usage rate. *Reliability Engineering and System Safety*, *145*, 147–154. https://doi.org/10.1016/j.ress.2015.09.012

Yavuz, O., Doğan, E., Carus, E., & Görgülü, A. (2019). Reliability Centered Maintenance Practices in Food Industry. *Procedia Computer Science*, *158*, 227–234. https://doi.org/10.1016/j.procs.2019.09.046

ANEXOS

7 ANEXOS

7.1 ANEX01- Função para gerar processo de renovação

```
def renewalprocess(parametros, Tsimul):
    num = 0
    numeros = list()
    if parametros[1] == 'norm':
        while num <Tsimul:
            a = stats.norm.rvs(loc=parametros[0][0], scale=parametros[0][1], size=1)
            num = num + a
            numeros.append(num)
        return numeros
    elif parametros[1] == 'dweibull':
        while num <Tsimul:
            k = parametros[0][0]
            a = stats.dweibull.rvs(k, loc=parametros[0][1], scale=parametros[0][2], size=1)
            num = num + a
            numeros.append(num)

        return numeros
    elif parametros[1] == 'lognorm':
        while num < Tsimul:
            k = parametros[0][0]
            a = stats.lognorm.rvs(k, loc=parametros[0][1], scale=parametros[0][2], size=1)
            num = num + a
            numeros.append(num)

        return numeros
    elif parametros[1] == 'gamma':
        while num < Tsimul:
            k = parametros[0][0]
            a = stats.gamma.rvs(k, loc=parametros[0][1], scale=parametros[0][2], size=1)
            num = num + a
            numeros.append(num)

        return numeros

    elif parametros[1] == 'beta':
        while num < Tsimul:
            k = parametros[0][0]
            j = parametros[0][1]
            a = stats.beta.rvs(k, j, loc=parametros[0][2], scale=parametros[0][3], size=1)
            num = num + a
            numeros.append(num)

        return numeros
```

7.2 ANEXO2-Valores críticos para o teste de Cramér Von Mises

	α				
M	0.20	0.15	0.10	0.05	0.01
2	0.138	0.149	0.162	0.175	0.186
3	0.121	0.135	0.154	0.184	0.23
4	0.121	0.134	0.155	0.191	0.28
5	0.121	0.137	0.160	0.199	0.30
6	0.123	0.139	0.162	0.204	0.31
7	0.124	0.140	0.165	0.208	0.32
8	0.124	0.141	0.165	0.210	0.32
9	0.125	0.142	0.167	0.212	0.32
10	0.125	0.142	0.167	0.212	0.32
11	0.126	0.143	0.169	0.214	0.32
12	0.126	0.144	0.169	0.214	0.32
13	0.126	0.144	0.169	0.214	0.33
14	0.126	0.144	0.169	0.214	0.33
15	0.126	0.144	0.169	0.215	0.33
16	0.127	0.145	0.171	0.216	0.33

17	0.127	0.145	0.171	0.217	0.33
18	0.127	0.146	0.171	0.217	0.33
19	0.127	0.146	0.171	0.217	0.33
20	0.128	0.146	0.172	0.217	0.33
30	0.128	0.146	0.172	0.218	0.33
60	0.128	0.147	0.173	0.220	0.33
100	0.129	0.147	0.173	0.220	0.34

Fonte:
https://www.webdepot.umontreal.ca/Usagers/angers/MonDepotPublic/STT3500H10/12avril/Cramer-von%20Mises.pdf

7.3 ANEXO3-Função para realizar teste de Cramér Von Mises

```
#função para fazer interpolações de dados de valores da tabela
def interpolation(d, x):
    res = d[0][1] + (x - d[0][0]) * ((d[1][1] - d[0][1]) / (d[1][0] - d[0][0]))

    return res

def testeCvm(d,b,a,T):
    N=len(d)
    tot=0
    #calculo de Cn
    if T==0:
        m = ((N - 1) / N) * b
    else:
        m=b
    for i in range (0,N):
        soma=((d[i]/d[N-1])**m-((2*(i+1)-1)/(2*N)))**2
        tot=tot+soma
    Cn=1/(12*N)+tot

#valores criticos da tabela
cvm = [[2, 0.138, 0.149, 0.162, 0.175, 0.186], [3, 0.121, 0.135, 0.154, 0.184, 0.23],
       [4, 0.121, 0.134, 0.155, 0.191, 0.28], [5, 0.121, 0.137, 0.160, 0.199, 0.30],
       [6, 0.123, 0.139, 0.162, 0.204, 0.31], [7, 0.124, 0.140, 0.165, 0.208, 0.32],
       [8, 0.124, 0.141, 0.165, 0.210, 0.32], [9, 0.125, 0.142, 0.167, 0.212, 0.32],
       [10, 0.125, 0.142, 0.167, 0.212, 0.32], [11, 0.126, 0.143, 0.169, 0.214, 0.32],
       [12, 0.126, 0.144, 0.169, 0.214, 0.32], [13, 0.126, 0.144, 0.169, 0.214, 0.33],
       [14, 0.126, 0.144, 0.169, 0.214, 0.33], [15, 0.126, 0.144, 0.169, 0.215, 0.33],
       [16, 0.127, 0.145, 0.171, 0.216, 0.33], [17, 0.127, 0.145, 0.171, 0.217, 0.33],
       [18, 0.127, 0.146, 0.171, 0.217, 0.33], [19, 0.127, 0.146, 0.171, 0.217, 0.33],
       [20, 0.128, 0.146, 0.172, 0.217, 0.33], [30, 0.128, 0.146, 0.172, 0.218, 0.33],
       [60, 0.128, 0.147, 0.173, 0.220, 0.33], [100, 0.129, 0.147, 0.173, 0.220, 0.34]]
#busca do valor tabelado correspondente para o tamanho da amostra e nivel de confiança
if 2<=N<=20 :
    if a==0.01:
        lim=cvm[N-2][5]
    elif a==0.05:
        lim = cvm[N-2][4]
    elif a == 0.1:
        lim = cvm[N - 2][3]
    elif a==0.15:
        lim = cvm[N-2][2]
    elif a == 0.20:
        lim = cvm[N - 2][1]
```

```
elif 20<N<30:
    if a==0.01:
        data = [[cvm[18][0], cvm[18][5]], [cvm[19][0], cvm[19][5]]]
        lim=interpolation(data,N)
    elif a==0.05:
        data = [[cvm[18][0], cvm[18][4]], [cvm[19][0], cvm[19][4]]]
        lim = interpolation(data, N)
    elif a==0.1:
        data = [[cvm[18][0], cvm[18][3]], [cvm[19][0], cvm[19][3]]]
        lim = interpolation(data, N)
    elif a==0.15:
        data = [[cvm[18][0], cvm[18][2]], [cvm[19][0], cvm[19][2]]]
        lim = interpolation(data, N)
    elif a==0.20:
        data = [[cvm[18][0], cvm[18][1]], [cvm[19][0], cvm[19][1]]]
        lim = interpolation(data, N)
elif N==30:
    if a==0.01:
        lim=cvm[19][5]
    elif a==0.05:
        lim = cvm[19][4]
    elif a == 0.1:
        lim = cvm[19][3]
    elif a==0.15:
        lim = cvm[19][2]
    elif a == 0.20:
        lim = cvm[19][1]
 elif 30<N<60:
     if a==0.01:
         data = [[cvm[19][0], cvm[19][5]], [cvm[20][0], cvm[20][5]]]
         lim=interpolation(data,N)
     elif a==0.05:
         data = [[cvm[19][0], cvm[19][4]], [cvm[20][0], cvm[20][4]]]
         lim = interpolation(data, N)
     elif a==0.1:
         data = [[cvm[19][0], cvm[19][3]], [cvm[20][0], cvm[20][3]]]
         lim = interpolation(data, N)
     elif a==0.15:
         data = [[cvm[19][0], cvm[19][2]], [cvm[20][0], cvm[20][2]]]
         lim = interpolation(data, N)
     elif a==0.20:
         data = [[cvm[19][0], cvm[19][1]], [cvm[20][0], cvm[20][1]]]
         lim = interpolation(data, N)
```

```
elif N==60:
    if a==0.01:
        lim=cvm[20][5]
    elif a==0.05:
        lim = cvm[20][4]
    elif a == 0.1:
        lim = cvm[20][3]
    elif a==0.15:
        lim = cvm[20][2]
    elif a == 0.20:
        lim = cvm[20][1]
elif 60 < N < 100:
    if a==0.01:
        data = [[cvm[20][0], cvm[20][5]], [cvm[21][0], cvm[21][5]]]
        lim=interpolation(data,N)
    elif a==0.05:
        data = [[cvm[20][0], cvm[20][4]], [cvm[21][0], cvm[21][4]]]
        lim = interpolation(data, N)
    elif a==0.1:
        data = [[cvm[20][0], cvm[20][3]], [cvm[21][0], cvm[21][3]]]
        lim = interpolation(data, N)
    elif a==0.15:
        data = [[cvm[20][0], cvm[20][2]], [cvm[21][0], cvm[21][2]]]
        lim = interpolation(data, N)
    elif a==0.20:
        lim = cvm[20][1]
elif 60 < N < 100:
    if a==0.01:
        data = [[cvm[20][0], cvm[20][5]], [cvm[21][0], cvm[21][5]]]
        lim=interpolation(data,N)
    elif a==0.05:
        data = [[cvm[20][0], cvm[20][4]], [cvm[21][0], cvm[21][4]]]
        lim = interpolation(data, N)
    elif a==0.1:
        data = [[cvm[20][0], cvm[20][3]], [cvm[21][0], cvm[21][3]]]
        lim = interpolation(data, N)
    elif a==0.15:
        data = [[cvm[20][0], cvm[20][2]], [cvm[21][0], cvm[21][2]]]
        lim = interpolation(data, N)
    elif a==0.20:
        data = [[cvm[20][0], cvm[20][1]], [cvm[21][0], cvm[21][1]]]
        lim = interpolation(data, N)
```

```
elif N >= 100:
    if a == 0.01:
            lim = cvm[21][5]
    elif a == 0.05:
            lim = cvm[21][4]
    elif a == 0.1:
            lim = cvm[21][3]
    elif a == 0.15:
            lim = cvm[21][2]
    elif a == 0.20:
            lim = cvm[21][1]
#comparação do valor obitido com o valor critico
if Cn>lim:
    print (f'o Valor de Cn ({Cn}) é superior ao valor critico '
           f'tabelado {lim}.O modelo não é adequado.')
else:
    print(f'o Valor de Cn ({Cn}) é inferior ao valor critico'
          f' tabelado {lim}.O modelo é adequado.')
# calculo da diferença entre o valor obtido e o valor critico
dif = abs(Cn-lim)
return dif
```

7.4 ANEXO4 – Funções de calculo no programa principal

```
#função que enumera o numero de avarias
def enumerarav(N):
    valores = list()
    for i in range(1,N+1):
        valores.append(i)
    return valores


#função que calcula a media das 4 simulações
def mediasimulaçoes(simul1,simul2,simul3,simul4):
    contagem=[]
    simulaçãofinal=[]
    N1=len(simul1)
    N2=len(simul2)
    N3=len(simul3)
    N4=len(simul4)
    contagem.append(N1)
    contagem.append(N2)
    contagem.append(N3)
    contagem.append(N4)
    Nf=min(contagem)
    for i in range(0,Nf):
        valor=(simul1[i]+simul2[i]+simul3[i]+simul4[i])/4
        simulaçãofinal.append(valor)
    return simulaçãofinal
```

```
# calcular disponibilidade média [0,t]
def caldispmed(simulaçaoxi, simulaçãod, tdis):
    Ti = []
    Yi = []
    #adicionar o valor do primeiro tempo entre ocorrencias
    Ti.append(simulaçaoxi[0])
    #somar o primeiro tempo de reparação
    Ti.append(simulaçãod[0] + Ti[0])
    j = f = 1
    i = 2
    soma = 0
    while True:
        #qunado o i é par,somar valor de xi ao valor anterior
        if i % 2 == 0:
            valor = Ti[i - 1] + simulaçaoxi[j]
            Ti.append(valor)
            j = j + 1
        # qunado o i é impar,somar valor de di ao valor anterior
        else:
            valor = Ti[i - 1] + simulaçãod[f]
            Ti.append(valor)
            f = f + 1
        i = i + 1
        N = len(Ti)
            if Ti[i - 1] > tdis:
                del (Ti[N - 1])
                Ti.append(tdis)
                break
        print(Ti)
        #gerar valores de Yi alternadamente 1,0
        for k in range(0, N):
            if k % 2 == 0:
                Yi.append(1)
            else:
                Yi.append(0)
        print(Yi)
        #formula de calculo da disponibilidade media
        for h in range(1, N):
            soma = soma + (Ti[h] - Ti[h - 1]) * Yi[h - 1]
        Dispomed = (1 / tdis) * soma
        disponmedf=(1-Dispomed)*100
        print(disponmedf)
        return disponmedf
```

```
# calcular disponibilidade média [0,T]
def caldispmedtot(simulaçaoxi, simulaçãod, tdis):
    #calculo do numero total de dados
    r=len(simulaçaoxi)
    a=2*r
    Ti = []
    Yi = []
    #adicionar o valor do primeiro tempo entre ocorrencias
    Ti.append(simulaçaoxi[0])
    # somar o primeiro tempo de reparação
    Ti.append(simulaçãod[0] + Ti[0])
    j = f = 1
    i = 2
    soma = 0
    for i in range(2,a):
        # qunado o i é par,somar valor de xi ao valor anterior
        if i % 2 == 0:
            valor = Ti[i - 1] + simulaçaoxi[j]
            Ti.append(valor)
            j = j + 1
        # qunado o i é impar,somar valor de di ao valor anterior
        else:
            valor = Ti[i - 1] + simulaçãod[f]
            Ti.append(valor)
            f = f + 1
        i = i + 1

        N = len(Ti)
    print(Ti)
    for k in range(0, N):
        if k % 2 == 0:
            Yi.append(1)
        else:
            Yi.append(0)
    print(Yi)
    #calcular disponibilidade media
    for h in range(1, N):
        soma = soma + (Ti[h] - Ti[h - 1]) * Yi[h - 1]
    Dispomed = (1 / tdis) * soma
    disponmedf = (1 - Dispomed) * 100
    print(disponmedf)
    return disponmedf
```

7.5 ANEXO5-Código do programa final

```
import xlrd
import numpy as np
from scipy import stats
from funçoes import funçaolaplace
from funçoes import funçaomodelos
from funçoes import modeloCramerVonmises
from funçoes import tranformar
from funçoes import funçaosimular
from funçoes import calculo
from tkinter import *
from idlelib.tooltip import Hovertip
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

# criar janela
janela = Tk()
janela.title('Ferramenta Automática Para Avaliação De Fiabilidade E Disponibilidade '
             'De Equipamentos Repáraveis')
janela.configure(background='#FFFACD')
janela.geometry('900x500')
janela.resizable(False,False)

# criar label e entrada nome ficheiro
lb_nomefi = Label(text='Nome do ficheiro Excel', bg='#63B8FF')
lb_nomefi.place(relx=0.05, rely=0.025, relwidth=0.17, relheight=0.05)
nomefi_entry = Entry()
nomefi_entry.place(relx=0.25, rely=0.025, relwidth=0.3, relheight=0.05)
Hovertip(lb_nomefi, f'Insira o nome do ficheiro Excel que pretende analisar \n'
                    f'com extenxão xlsx. Por exemplo se o ficheiro se chamar \n'
                    f'"temposav" deverá inserir no nome o seguinte:"temposav.xlsx" \n'
                    f'Note que na folha excel na primeira coluna tem de introduzir os \n'
                    f'tempos acumulados de avarias (Ti) e na segunda coluna os \n'
                    f'tempos de reparação (Di).')
# criar label e entrada do nome da folha
lb_nomefo = Label(text='Nome da Folha no ficheiro Excel', bg='#63B8FF')
lb_nomefo.place(relx=0.05, rely=0.1, relwidth=0.22, relheight=0.05)
nomefo_entry = Entry()
nomefo_entry.place(relx=0.3, rely=0.1, relwidth=0.25, relheight=0.05)
Hovertip(lb_nomefo, f'Deverá inserir o nome da folha dentro do ficheiro \n'
                    f'Excel que pretende analisar. Por exemplo "folha1".')
```

```
# criar label e entrada do tempo de observação
lb_tobs = Label(text='Tobservação', bg='#63B8FF')
lb_tobs.place(relx=0.57, rely=0.025, relwidth=0.1, relheight=0.05)
tobs_entry = Entry()
tobs_entry.place(relx=0.69, rely=0.025, relwidth=0.075, relheight=0.05)
Hovertip(lb_tobs, f'Deverá inserir o tempo total de observação do \n'
                  f'equipamento para o qual surgiu o historico de \n'
                  f'tempos de avarias que pretende analisar. Caso o \n'
                  f'historico de avarias não seja limitado pelo tempo \n'
                  f'mas sim pelonúmero de avarias deverá considerar que \n'
                  f'Tobservação=0.Nota: As unidades do tempo de observação \n'
                  f'têm de estar em confirmidade com as unidades do historico \n'
                  f'de avarias.Por exemplo  se os tempos acumulados de avarias \n'
                  f'e os tempos de reparação estiverem em horas, o tempo de \n'
                  f'observação tem de estar em horas.')
# criar label e entrada do alfa
lb_alfa = Label(text='α', bg='#63B8FF')
lb_alfa.place(relx=0.57, rely=0.1, relwidth=0.05, relheight=0.05)

alfa_entry = Entry()
alfa_entry.place(relx=0.63, rely=0.1, relwidth=0.075, relheight=0.05)
Hovertip(lb_alfa, f' A região crítica  é o conjunto de valores da \n '
                  f' estatística de teste que nos levam a rejeitar a \n'
                  f' hipótese nula. Por exemplo, se pretender realizar \n'
                  f' uma analise estatistica com um nivel de confiança \n '
                  f' de 90% deverá considerar α=0.1. Nota:α pode assumir\n'
                  f' os seguintes valores:0.01,0.05,0.1,0.15,0.20.')
# criar label e entrada Tempo simulação
lb_Tsimul = Label(text='Tempo de simulação', bg='#63B8FF')
lb_Tsimul.place(relx=0.05, rely=0.175, relwidth=0.2, relheight=0.05)
Tsimul_entry = Entry()
Tsimul_entry.place(relx=0.275, rely=0.175, relwidth=0.1, relheight=0.05)
Hovertip(lb_Tsimul, f'Insira o tempo de simulação que \n'
                    f'pretende. Nota:As unidades de tempo \n'
                    f'devem estar em conformidade com as \n'
                    f'unidades do historico de avarias. ')
# criar label e entrada Tempo calculo disponibilidade media ate t
lb_tcalcdisp = Label(text=f'Tempo (t) para cálculo de Disponibilidade média ', bg='#63B8FF')
lb_tcalcdisp.place(relx=0.4, rely=0.175, relwidth=0.30, relheight=0.05)
Tcalcdisp_entry = Entry()
Tcalcdisp_entry.place(relx=0.72, rely=0.175, relwidth=0.1, relheight=0.05)
Hovertip(lb_tcalcdisp, f'Insira o valor de t para calcular a \n '
                       f'disponibilidade média no intervalo de \n'
                       f'tempo [0,t].Nota:As unidades de tempo \n'
                       f'devem estar em conformidade com as  \n'
                       f'unidades do historico de avarias. ')
```

```
#label indicadores de fiabilidade
lb_indicadores=Label(text=f'Indic. de Fiabilidade',bg='#FF8C00',font=25)
lb_indicadores.place(relx=0.05, rely=0.250, relwidth=0.2, relheight=0.05)
#Label detalhes
lb_detalhes=Label(text=f'Detalhes',bg='#FF8C00',font=25)
lb_detalhes.place(relx=0.05, rely=0.7, relwidth=0.2, relheight=0.05)

#label mttr
lb_inicialmttr=Label(text='MTTR',bg='#3D9140')
lb_inicialmttr.place(relx=0.05, rely=0.325, relwidth=0.1, relheight=0.05)
#label mtbf
lb_inicialmtbf=Label(text='MTBF',bg='#3D9140')
lb_inicialmtbf.place(relx=0.05, rely=0.4, relwidth=0.1, relheight=0.05)
#label disponibilidade istantanea
lb_inicialdisp=Label(text='Dispon inst',bg='#3D9140')
lb_inicialdisp.place(relx=0.05, rely=0.475, relwidth=0.1, relheight=0.05)
#label disponibilidade media t
lb_inicialdispmedia=Label(text='Dispon méd.[0,t]',bg='#3D9140')
lb_inicialdispmedia.place(relx=0.05, rely=0.55, relwidth=0.1, relheight=0.05)
#label disponibilidade media total
lb_inicialdispmediatot=Label(text='Dispon méd.',bg='#3D9140')
lb_inicialdispmediatot.place(relx=0.05, rely=0.625, relwidth=0.1, relheight=0.05)

#label proceso utilizado
lb_processo=Label(text='Processo',bg='#3D9140')
lb_processo.place(relx=0.05, rely=0.775, relwidth=0.1, relheight=0.05)
#label modelo utilizado
lb_modelo=Label(text='Modelo/Distr.',bg='#3D9140')
lb_modelo.place(relx=0.05, rely=0.850, relwidth=0.1, relheight=0.05)
#label distribuição utilizada
lb_distrep=Label(text='Distr. Trep',bg='#3D9140')
lb_distrep.place(relx=0.05, rely=0.925, relwidth=0.1, relheight=0.05)

def funçaolimpar():
    nomefi_entry.delete(0,END)
    nomefo_entry.delete(0,END)
    tobs_entry.delete(0,END)
    alfa_entry.delete(0,END)
    Tsimul_entry.delete(0,END)
    Tcalcdisp_entry.delete(0,END)
```

```
def programafinal():
    T=float(tobs_entry.get())
    alfa=float(alfa_entry.get())
    wbentrada=str(nomefi_entry.get())
    pentrada=str(nomefo_entry.get())
    Tsimul=float(Tsimul_entry.get())
    tdispmed=float(Tcalcdisp_entry.get())
    wb = xlrd.open_workbook(wbentrada)
    p = wb.sheet_by_name(pentrada)
    x = p.col_values(0)
    y = p.col_values(1)
    if y[len(y)-1]==0:
        ultim=len(y)
        y.__delitem__(ultim-1)
    k = tranformar.dadosx(x, y)
    print(x)
    print(y)
    print(k)
    f = funçaolaplace.laplace(x, alfa, T)
    infos=[]
    if f == 0:
        parametros = funçaomodelos.funçaomodelo3(k)
        print(parametros)
        if parametros[1] == 'expon':
            simul1 = funçaosimular.hpp((1/parametros[0][1]), Tsimul)
            simul2 = funçaosimular.hpp((1/parametros[0][1]), Tsimul)
            simul3 = funçaosimular.hpp((1/parametros[0][1]), Tsimul)
            simul4 = funçaosimular.hpp((1/parametros[0][1]), Tsimul)
            infos=['HPP', parametros[1]]
        else:
            simul1 = funçaosimular.renewalprocess(parametros, Tsimul)
            simul2 = funçaosimular.renewalprocess(parametros, Tsimul)
            simul3 = funçaosimular.renewalprocess(parametros, Tsimul)
            simul4 = funçaosimular.renewalprocess(parametros, Tsimul)
            infos=['Proc. Renovação',parametros[1]]

        if f == 1:
            b = funçaomodelos.funçaomodelo1(x, 0)
            r = modeloCramerVonmises.testeCvm(x, b[0], alfa, T)
            s = funçaomodelos.funçaomodelo2(x)
            t = modeloCramerVonmises.testeCvm(x, s[0], alfa, T)
```

```
        if s[0] > 0:
            if t < r:
                parametros = s
                simul1 = funçaosimular.nhpp2(parametros, Tsimul)
                simul2 = funçaosimular.nhpp2(parametros, Tsimul)
                simul3 = funçaosimular.nhpp2(parametros, Tsimul)
                simul4 = funçaosimular.nhpp2(parametros, Tsimul)
                infos = ['NHPP', 'Cox-Lewis']
            else:
                parametros = b
                simul1 = funçaosimular.nhpp(parametros, Tsimul)
                simul2 = funçaosimular.nhpp(parametros, Tsimul)
                simul3 = funçaosimular.nhpp(parametros, Tsimul)
                simul4 = funçaosimular.nhpp(parametros, Tsimul)
                infos = ['NHPP', 'Power law']
        else:
            parametros = b
            simul1 = funçaosimular.nhpp(parametros, Tsimul)
            simul2 = funçaosimular.nhpp(parametros, Tsimul)
            simul3 = funçaosimular.nhpp(parametros, Tsimul)
            simul4 = funçaosimular.nhpp(parametros, Tsimul)
            infos = ['NHPP', 'Power law']


simul=calculo.mediasimulaçoes(simul1,simul2,simul3,simul4)
parametrosD = funçaomodelos.funçaomodelo3(y)
print(parametrosD)
infos.append(parametrosD[1])
N = len(simul)
repsimul1 = funçaosimular.temposreparaçao(parametrosD, N)
repsimul2 = funçaosimular.temposreparaçao(parametrosD, N)
repsimul3 = funçaosimular.temposreparaçao(parametrosD, N)
repsimul4 = funçaosimular.temposreparaçao(parametrosD, N)
repsimul=calculo.mediasimulaçoes(repsimul1,repsimul2,repsimul3,repsimul4)
somarepsimul = sum(repsimul)
mttr = somarepsimul / N
finalmttr.set(str(round(mttr, 4)))
simulaçaodosXi = tranformar.dadosx(simul, repsimul)
somaxi = sum(simulaçaodosXi)
```

```
if f==0:
    if parametros[1] !='expon':
        somaxinum = somaxi[0]
    else:
        somaxinum = somaxi
else:
    somaxinum = somaxi
mtbf = somaxinum / N
finalmtbf.set(str(round(mtbf, 4)))
disponibilidade = (mtbf / (mtbf + mttr)) * 100
finaldisponibilidade.set(str(round(disponibilidade, 4)) + '%')
disponmed_t=calculo.caldispmed(simulaçaodosXi,repsimul,tdispmed)
if f==0:
    if parametros[1] !='expon':
        disponmedt = disponmed_t[0]
    else:
        disponmedt = disponmed_t
else:
    disponmedt = disponmed_t
finaldispmed_t.set(str(round(disponmedt,4))+'%')
dispomed_tot=calculo.caldispmedtot(simulaçaodosXi,repsimul,Tsimul)
if f == 0:
    if parametros[1] != 'expon':
        disponmedtot = dispomed_tot [0]
    else:
        disponmedtot = dispomed_tot
else:
    disponmedtot = dispomed_tot
finaldispmedtotal.set(str(round(disponmedtot,4))+'%')
finalprocesso.set(str(infos[0]))
finalmodelo.set(str(infos[1]))
finaldisrep.set(str(infos[2]))
ordenadasgraf=calculo.enumerarav(N)
#criando graficos
figura = plt.Figure(figsize=(5, 5), dpi=60)
grafico = figura.add_subplot(111)
figura2 = plt.Figure(figsize=(5.8, 5), dpi=60)
grafico2 = figura2.add_subplot(111)
canva = FigureCanvasTkAgg(figura, janela)
canva.get_tk_widget().place(relx=0.26,rely=0.3)
```

```
canva2 = FigureCanvasTkAgg(figura2, janela)
canva2.get_tk_widget().place(relx=0.60, rely=0.3)
grafico.set_title('Número esperado de avarias em função de Ti')
grafico.set_xlabel('Tempo (u.t)')
grafico.set_ylabel('N(T)')
grafico.plot(simul, ordenadasgraf)
grafico2.set_title('Distribuição dos tempos de reparação')
grafico2.set_xlabel('Tempo (u.t)')
grafico2.set_ylabel('f(t) densidade probabilidade')
if parametrosD[1] == 'norm':
    x = np.linspace(stats.norm.ppf(0.01,loc=parametrosD[0][0],scale=parametrosD[0][1]),
                    stats.norm.ppf(0.99,loc=parametrosD[0][0],scale=parametrosD[0][1]), N)
    grafico2.plot(x, stats.norm.pdf(x,loc=parametrosD[0][0],scale=parametrosD[0][1]))
elif parametrosD[1]=='lognorm':
    s=parametrosD[0][0]
    x = np.linspace(stats.lognorm.ppf(0.01,s, loc=parametrosD[0][1], scale=parametrosD[0][2]),
                    stats.lognorm.ppf(0.99,s, loc=parametrosD[0][1], scale=parametrosD[0][2]), N)
    grafico2.plot(x, stats.lognorm.pdf(x,s, loc=parametrosD[0][1], scale=parametrosD[0][2]))
elif parametrosD[1]=='expon':
    x = np.linspace(stats.expon.ppf(0.01, loc=parametrosD[0][0], scale=parametrosD[0][1]),
                    stats.expon.ppf(0.99, loc=parametrosD[0][0], scale=parametrosD[0][1]),N)
    grafico2.plot(x, stats.expon.pdf(x, loc=parametrosD[0][0], scale=parametrosD[0][1]))
elif parametrosD[1]=='gamma':
    a=parametrosD[0][0]
    x = np.linspace(stats.gamma.ppf(0.01,a, loc=parametrosD[0][1], scale=parametrosD[0][2]),
                    stats.gamma.ppf(0.99,a, loc=parametrosD[0][1], scale=parametrosD[0][2]), N)
    grafico2.plot(x, stats.gamma.pdf(x,a, loc=parametrosD[0][1], scale=parametrosD[0][2]))
elif parametrosD[1]=='beta':
    a = parametrosD[0][0]
    b = parametrosD[0][1]
    x = np.linspace(stats.beta.ppf(0.01, a,b,loc=parametrosD[0][2], scale=parametrosD[0][3]),
                    stats.beta.ppf(0.99, a,b,loc=parametrosD[0][2], scale=parametrosD[0][3]), N)
    grafico2.plot(x, stats.beta.pdf(x,a,b, loc=parametrosD[0][2], scale=parametrosD[0][3]))
elif parametrosD[1]=='dweibull':
    c = parametrosD[0][0]
    x = np.linspace(stats.weibull_min.ppf(0.01, c, loc=parametrosD[0][1], scale=parametrosD[0][2]),
                    stats.weibull_min.ppf(0.99, c, loc=parametrosD[0][1], scale=parametrosD[0][2]), N)
    grafico2.plot(x, stats.weibull_min.pdf(x,c, loc=parametrosD[0][1], scale=parametrosD[0][2]))

#declarar variaveis de saida
finaldispmedtotal=StringVar()
finaldispmed_t=StringVar()
finaldisrep=StringVar()
finalmodelo=StringVar()
finalprocesso=StringVar()
finalmttr=StringVar()
finalmtbf=StringVar()
finaldisponibilidade=StringVar()

#label resultado mttr
lb_finalmttr=Label(textvariable=finalmttr, bg='#C4C4C4')
lb_finalmttr.place(relx=0.15, rely=0.325, relwidth=0.1, relheight=0.05)
```

```
#label resultado mtbf
lb_finalmtbf=Label(textvariable=finalmtbf, bg='#C4C4C4')
lb_finalmtbf.place(relx=0.15, rely=0.4, relwidth=0.1, relheight=0.05)
#label resultado disponibilidade instantanea
lb_finaldisponibilidade=Label(textvariable=finaldisponibilidade, bg='#C4C4C4')
lb_finaldisponibilidade.place(relx=0.15, rely=0.475, relwidth=0.1, relheight=0.05)
#label resultado disponibilidade media ate instante t
lb_finaldispomed_t=Label(textvariable=finaldispmed_t, bg='#C4C4C4')
lb_finaldispomed_t.place(relx=0.15, rely=0.550, relwidth=0.1, relheight=0.05)
#label resultado disponibilidade media total
lb_finaldispomed_total=Label(textvariable=finaldispmedtotal, bg='#C4C4C4')
lb_finaldispomed_total.place(relx=0.15, rely=0.625, relwidth=0.1, relheight=0.05)
#label resultado processo
lb_finalprocesso=Label(textvariable=finalprocesso, bg='#C4C4C4')
lb_finalprocesso.place(relx=0.15, rely=0.775, relwidth=0.1, relheight=0.05)
#label resultado modelo
lb_finalmodelo=Label(textvariable=finalmodelo, bg='#C4C4C4')
lb_finalmodelo.place(relx=0.15, rely=0.850, relwidth=0.1, relheight=0.05)
#label resultado distrep
lb_finaldisrep=Label(textvariable=finaldisrep, bg='#C4C4C4')
lb_finaldisrep.place(relx=0.15, rely=0.925, relwidth=0.1, relheight=0.05)

# criar botao limpar
btlimpar = Button(text='Limpar', bd=2, bg='#C1C1CD', command=funçaolimpar)
btlimpar.place(relx=0.85, rely=0.025, relwidth=0.1, relheight=0.05)

# butao ok
btok = Button(text='Confirmar', bd=2, bg='#C1C1CD',command=programafinal)
btok.place(relx=0.85, rely=0.1, relwidth=0.1, relheight=0.05)

janela.mainloop()
```

7.6 ANEXO6- Geração das 3 amostras exemplificativas de HPP e respetivo gráfico

```
from numpy import log as ln
import matplotlib.pyplot
import numpy
def hpp(l,Tsimul):
    t=0
    numeros=list()
    contador=list()
    final=list()
    i=1
    #fixar valores da simulaçao
    numpy.random.seed(1234)
    while t<Tsimul :
        #gerar distribuição uniforme
        u=numpy.random.uniform (0,1)
        #gerar os tempos das ocorrencias
        t=t-(ln(u)/(l))
        numeros.append(t)
        contador.append(i)
        i=i+1
    final.append(numeros)
    final.append(contador)
    return final
#fazer diferentes simulações
a= hpp(0.3,20)
b= hpp(1,20)
c=hpp(3,20)
#representar as simulaçoes no grafico
matplotlib.pyplot.plot(a[0],a[1],'*')
matplotlib.pyplot.plot(b[0],b[1],'*')
matplotlib.pyplot.plot(c[0],c[1],'*')
matplotlib.pyplot.xlim(0,21)
matplotlib.pyplot.xlabel('Tempo (u.t)')
matplotlib.pyplot.ylabel('N(t)')
matplotlib.pyplot.show()
```

7.7 ANEXO7-Geração dos processos exemplificativos NHPP e respetivos histogramas

7.7.1 Modelo lei de potencia (β>1)

```
import numpy
from numpy import log as ln
import matplotlib.pyplot
#função para gerar NHPP modelo lei de potencia
def nhpp(parametros, Tsimul):
    numeros = list()
    # max λ(t)= λ*t**(β)
    l = parametros[1]*Tsimul**(parametros[0])
    t = 0
    #fixar a simulação
    numpy.random.seed(1970)
    while t<Tsimul:
        u = numpy.random.uniform(0, 1)
        #gerar tempos candidados a NHPP
        t = t - (ln(u) / l)
        # a função taxa de falha λ(t)=λ β t^(β-1)
        fun = parametros[1] * parametros[0] * t ** (parametros[0] - 1)
        #verificar se os tempos são aceites
        if u <= fun / l:
            numeros.append(t)
    return numeros
#simular para B>1
parametros=[2.5,0.000064]
x=nhpp(parametros,1000)
#fazer histograma
matplotlib.pyplot.hist(x,5)
matplotlib.pyplot.xlabel('Tempo (u.t)')
matplotlib.pyplot.ylabel('N(T)')
matplotlib.pyplot.show()
```

7.7.2 Modelo lei de potência (β>1)

```
import numpy
from numpy import log as ln
import matplotlib.pyplot
#função para gerar NHPP modelo lei de potencia
def nhpp(parametros, Tsimul):
    numeros = list()
    # max λ(t)= λ*t**(β)
    l = parametros[1]*Tsimul**(parametros[0])
    t = 0
    #fixar a simulação
    numpy.random.seed(1970)
    while t<Tsimul:
        u = numpy.random.uniform(0, 1)
        #gerar tempos candidados a NHPP
        t = t - (ln(u) / l)
        # a função taxa de falha λ(t)=λ β t^(β-1)
        fun = parametros[1] * parametros[0] * t ** (parametros[0] - 1)
        #verificar se os tempos são aceites
        if u <= fun / l:
            numeros.append(t)
    return numeros

#simular para B=1
parametros=[1,0.064]
x=nhpp(parametros,1000)
#fazer histograma
matplotlib.pyplot.hist(x,5)
matplotlib.pyplot.xlabel('Tempo (u.t)')
matplotlib.pyplot.ylabel('N(T)')
matplotlib.pyplot.show()
```

7.7.3 Modelo lei de potência (β<1)

```
import numpy
from numpy import log as ln
import matplotlib.pyplot
#função para gerar NHPP modelo lei de potencia
def nhpp(parametros, Tsimul):
    numeros = list()
    # max λ(t)= λ*t**(β)
    l = parametros[1]*Tsimul**(parametros[0])
    t = 0
    # fixar a simulação
    numpy.random.seed(1952)
    while t<Tsimul:
        u = numpy.random.uniform(0, 1)
        #gerar tempos candidados a NHPP
        t = t - (ln(u) / l)
        # a função taxa de falha λ(t)=λ β t^(β-1)
        fun = parametros[1] * parametros[0] * t ** (parametros[0] - 1)
        #verificar se os tempos são aceites
        if u <= fun / l:
            numeros.append(t)
    return numeros
#simular para B<1
parametros=[0.1,40]
x=nhpp(parametros,1000)
#fazer histograma
matplotlib.pyplot.hist(x,5)
matplotlib.pyplot.xlabel('Tempo (u.t)')
matplotlib.pyplot.ylabel('N(T)')
matplotlib.pyplot.show()
```

7.7.4 Modelo Cox-Lewis ($\alpha_1 > 0$)

```
import numpy
from numpy import log as ln
import matplotlib.pyplot
import math
#função gerar nhpp cox lewis
def nhpp2(parametros, Tsimul):
    numeros = list()
    # max λ(t)= (e^(a0)/(a1))*(e^(a1*t)-1)
    l =((math.exp(parametros[1])/parametros[0]))*(math.exp(parametros[0]*Tsimul)-1)
    t = 0
    #fixar a simulação
    numpy.random.seed(1234)
    while t<Tsimul:
        u = numpy.random.uniform(0, 1)
        # gerar tempos candidados a NHPP
        t = t - (ln(u) / l)
        # a função taxa de falha λ(t)= e^(a0+a1*t)
        fun =math.exp(parametros[1]+parametros[0]*t)
        #verificar se os tempos são aceites
        if u <= fun / l:
            numeros.append(t)
    return numeros
#simular para alfa1>0
parametros=[0.000107,-6.54481]
x=nhpp2(parametros,22000)
#fazer histograma
matplotlib.pyplot.hist(x,5)
matplotlib.pyplot.xlabel('Tempo (u.t)')
matplotlib.pyplot.ylabel('N(T)')
matplotlib.pyplot.show()
```

7.8 ANEXO8- Geração das 3 amostras exemplificativas de PR e respetivo gráfico

```
from scipy import stats
import matplotlib.pyplot
import numpy as np
#função para gerar processo de renovação
def renewalprocess(parametros, Tsimul):
    num = 0
    numeros = list()
    #fixar os valores da silulação
    np.random.seed(1200)
    if parametros[1] == 'norm':
        while num <Tsimul:
            a = stats.norm.rvs(loc=parametros[0][0], scale=parametros[0][1], size=1)
            num = num + a
            numeros.append(num)
        return numeros
    elif parametros[1] == 'dweibull':
        while num <Tsimul:
            k = parametros[0][0]
            a = stats.dweibull.rvs(k, loc=parametros[0][1], scale=parametros[0][2], size=1)
            num = num + a
            numeros.append(num)

        return numeros
 elif parametros[1] == 'lognorm':
    while num < Tsimul:
        k = parametros[0][0]
        a = stats.lognorm.rvs(k, loc=parametros[0][1], scale=parametros[0][2], size=1)
        num = num + a
        numeros.append(num)

    return numeros
 elif parametros[1] == 'gamma':
    while num < Tsimul:
        k = parametros[0][0]
        a = stats.gamma.rvs(k, loc=parametros[0][1], scale=parametros[0][2], size=1)
        num = num + a
        numeros.append(num)

    return numeros
```

```
    elif parametros[1] == 'beta':
        while num < Tsimul:
            k = parametros[0][0]
            j = parametros[0][1]
            a = stats.beta.rvs(k, j, loc=parametros[0][2], scale=parametros[0][3], size=1)
            num = num + a
            numeros.append(num)

        return numeros
#Funçao para enumerar as avarias
def numerototalav(N):
    valores = list()
    for i in range(1,N+1):
        valores.append(i)
    return valores
#simular 3 amostras para processo de renovaçao
parametros=[(8,1),'norm']
parametros1=[(2,0,4),'gamma']
parametros2=[(1,4,1),'lognorm']
x=renewalprocess(parametros,100)
y=renewalprocess(parametros1,100)
z=renewalprocess(parametros2,100)

#contagem do numero das avarias
N1=len(x)
N2=len(y)
N3=len(z)
#enumerar as avarias
a=numerototalav(N1)
b=numerototalav(N2)
c=numerototalav(N3)
#criar grafico para as 3 amostras
matplotlib.pyplot.plot(x,a,'*')
matplotlib.pyplot.plot(y,b,'*')
matplotlib.pyplot.plot(z,c,'*')
matplotlib.pyplot.xlabel('Tempo (u.t)')
matplotlib.pyplot.ylabel('N(T)')
matplotlib.pyplot.show()
```

7.9 ANEXO9- Geração do gráfico exemplificativo dos tempos de reparação com distribuição normal

```
from scipy import stats
import numpy as np
import matplotlib.pyplot

parametrosD=[(3,1),'norm']
np.random.seed(1200)
if parametrosD[1] == 'norm':
    x = np.linspace(stats.norm.ppf(0.01,loc=parametrosD[0][0],scale=parametrosD[0][1]),
                    stats.norm.ppf(0.99,loc=parametrosD[0][0],scale=parametrosD[0][1]), 1000)
    matplotlib.pyplot.plot(x, stats.norm.pdf(x, loc=parametrosD[0][0], scale=parametrosD[0][1]))
    matplotlib.pyplot.xlabel('Tempo (u.t)')
    matplotlib.pyplot.ylabel('f(t) densidade probabilidade')
    matplotlib.pyplot.show()
```