Universidade Federal
do Rio de Janeiro
Escola Politécnica

CÁLCULO DA FREQUÊNCIA DE ACIDENTES EM UMA USINA NUCLEAR EQUIPADA COM UM CANAL DE PROTEÇÃO EM ENVELHECIMENTO PELO MÉTODO MONTE CARLO

Bruno Napoli Warth

Projeto de Graduação apresentado ao Curso de Engenharia Nuclear da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientador: Prof. Paulo Fernando Ferreira Frutuoso e Melo

Rio de Janeiro

Fevereiro de 2019

CÁLCULO DA FREQUÊNCIA DE ACIDENTES EM UMA USINA NUCLEAR EQUIPADA COM UM CANAL DE PROTEÇÃO EM ENVELHECIMENTO PELO MÉTODO MONTE CARLO

Bruno Napoli Warth

PROJETO DE GRADUAÇÃO SUBMETIDA AO CORPO DOCENTE DO CURSO DE ENGENHARIA NUCLEAR DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO NUCLEAR.

Examinado por:

_____

Prof. Paulo Fernando Ferreira Frutuoso e Melo

_____

Prof. Fernando Carvalho da Silva

_____

Prof. Antônio Carlos Marques Alvim

RIO DE JANEIRO, RJ - BRASIL

FEVEREIRO de 2019

Warth, Bruno Napoli

CÁLCULO DA FREQUÊNCIA DE ACIDENTES EM UMA USINA NUCLEAR EQUIPADA COM UM CANAL DE PROTEÇÃO EM ENVELHECIMENTO PELO MÉTODO MONTE CARLO

/ Bruno Napoli Warth, – Rio de Janeiro: UFRJ/ESCOLA POLITÉCNICA, 2019

x, 30 p.: il.; 29.7cm.

Orientador: Paulo Fernando Ferreira Frutuoso e Melo

Projeto de Graduação – UFRJ/ Escola Politécnica/ Curso de Engenharia Nuclear, 2019.

Referências Bibliográficas: p.29-30.

1.Sistemas de Proteção. 2.Instalações Industriais. 3. Frequência de Ocorrência de Acidentes. 4.Envelhecimento . 5.Método de Monte Carlo. I. Frutuoso e Melo, Paulo Fernando Ferreira. II. Universidade Federal do Rio de Janeiro, Escola Politécnica, Curso de Engenharia Nuclear. III. Cálculo Da Frequência De Acidentes Em Uma Usina Equipada Com Um Canal De Proteção Em Envelhecimento Pelo Método Monte Carlo

Agradecimentos

Gostaria de agradecer, em primeiro lugar, à CAPES, pela oportunidade oferecida na forma da bolsa Jovens Talentos para Ciência, da qual fui beneficiário desde o segundo semestre da faculdade. Essa iniciação científica iniciou o projeto que veio a se tornar, anos depois, esse trabalho que escrevi.

Em segundo lugar, um agradecimento com uma pitada de saudades antecipada à Universidade Federal do Rio de Janeiro e à Escola Politécnica, instituições que me trouxeram orgulho e estresse em porções alternadas, apesar de pendendo sempre para o primeiro.

Não posso esquecer, por último, de todo o corpo docente, discente e administrativo do meu curso, em especial ao professor Paulo Fernando, meu orientador e querido amigo. Também agradeço a todos os colegas e amigos que fiz durante o curso, que, sem exceção, sempre mostraram apoio e colaboração dentro de um ambiente tão competitivo como a engenharia.

Resumo do Projeto de Graduação apresentado à Escola Politécnica/ UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro Nuclear.

Cálculo Da Frequência De Acidentes Em Uma Usina Nuclear Equipada Com Um Canal De Proteção Em Envelhecimento Pelo Método Monte Carlo

Bruno Napoli Warth

Fevereiro/2019

Orientador: Paulo Fernando Ferreira Frutuoso e Melo

Curso: Engenharia Nuclear

Este trabalho propõe uma modelagem matemática para quantificar o envelhecimento de sistemas protetores. Dado que a Cadeia Markoviana sozinha não é apta a resolver problemas onde a taxa de falha dos componentes não é constante, outros métodos são procurados para a obtenção de características de interesse da análise de segurança, como a indisponibilidade e a frequência de acidentes. O Método Monte Carlo se apresenta como um candidato para a resolução do problema, por sua versatilidade e precisão. A modelagem matemática é feita de tal maneira que os efeitos do envelhecimento do sistema de proteção sejam contabilizados pela geração de tempos de falha aleatórios que seguem uma distribuição mista, com o envelhecimento surgindo após um tempo de vida útil. O código apresenta uma boa precisão e se mostra adequado para realizar esse cálculo.

Palavras-chave: Frequência de Ocorrência de Acidentes, Envelhecimento, Monte Carlo, Sistemas de Proteção, Usina Nuclear

Abstract of Undergraduate Project presented to POLI/UFRJ as a partial fulfillment of the requirements for the degree of Nuclear Engineer.

Accident Rate Calculation On A Nuclear Power Plant equipped with Aging Single Channel Trip Device Through Monte Carlo Method

Bruno Napoli Warth

February/2019

Advisor: Paulo Fernando Ferreira Frutuoso e Melo

Course: Engenharia Nuclear

This project proposes a mathematical model to quantify the aging of protective systems. Given that the Markovian Chain alone is not able to solve problems when the components' failure rate are not constant, other methods are sought to obtain characteristics of interest for safety analysis, such as unavailability and accident rate. The Monte Carlo Method presents itself as a candidate to solve this problem, because of its versatility and precision. The mathematical modeling is done in such way that the protective system aging effects are accounted for by generating random failure times that follow a mixed distribution, appearing after a useful life period. The code presents good precision and is demonstrated to be adequate for this calculation.

Palavras-chave: Accident Rate, Aging, Monte Carlo Method, Protective Systems, Nuclear Power Plant

Índice

Lista de Figuras

Lista de Tabelas

Nomenclatura

υ	Taxa de demanda (ano^{-1})
μ	Taxa de reparo (ano^{-1})
τ_p	Intervalo entre testes (ano)
γ	Probabilidade de falha no reparo
λ	Taxa de Falha (ano^{-1})
θ	Parâmetro de Escala (ano)
m	Parâmetro de Forma
t_o	Tempo até início do envelhecimento (ano)

1 *Introdução*

O número total de reatores nucleares de potência no mundo ultrapassa a marca de 450 em operação atualmente. Em termos de geração de energia elétrica, isso representa um total de quase 2,5 milhões de TWh, ou aproximadamente 10,5% do total de energia elétrica gerada no mundo no ano de 2017 [1][2]. Dentre essas usinas, 284 completaram 30 anos ou mais até 31 de dezembro de 2017[1]. Levando em conta a preocupação atual com a emissão de gases causadores do aquecimento global [3] e o alto custo das energias renováveis - como solar e eólica, ainda em desenvolvimento tecnológico -, as usinas nucleares ainda são avaliadas como uma opção segura e barata [4] para geração de energia limpa.

Logo, a decisão de estender a vida das plantas [5] tem sido tomada em diversos países, e vem acompanhada pela missão dos órgãos reguladores de entender como o envelhecimento afeta os sistemas e o que deve ser feito em termos de mudanças tecnológicas para manter os requisitos de segurança e sua viabilidade econômica em níveis aceitáveis. De acordo com a *Nuclear Regulatory Commission* (NRC), a agência reguladora estadunidense, "considerações econômicas e antitruste, não limitações da tecnologia nuclear, determinaram originalmente o período de 40 anos para a licença de reatores. Por causa desse período selecionado, alguns sistemas, estruturas e componentes foram projetados com base em uma expectativa de vida de 40 anos" [6], ocorrendo o envelhecimento após esse período.

Nesse contexto, é de suma importância analisar os sistemas de proteção de um reator com base em métodos matemáticos que nos permitam avaliar com maior precisão as consequências do envelhecimento às quais o mesmo está sujeito. No caso dos sistemas de proteção em envelhecimento, o cálculo da frequência de acidentes irá depender da taxa de falha (crescente em função do tempo), da taxa de reparo do sistema, da taxa de demanda para esse sistema e do intervalo entre os testes [7]. A modelagem por cadeia markoviana, desenvolvido no trabalho de OLIVEIRA et al. [8] apresenta um modelo analítico que permite a avaliação da influência das variáveis que governam o valor da indisponibilidade e a frequência de acidentes, além da influência da variação desses parâmetros no resultado final.

No entanto, uma análise com taxas de falha variáveis no tempo implica na não aplicação da cadeia markoviana, sendo necessário recorrer a outros métodos. OLIVEIRA [9], por exemplo, apresenta uma modelagem que utiliza o método das variáveis suplementares, cujos resultados serão usados para fins de comparação neste trabalho, juntamente com os resultados de [8]. Já o artigo de FRUTUOSO E MELO et al. [7] empregou o método de Monte Carlo para a realização do cálculo da frequência de acidente, modelo que aqui será discutido e aprofundado.

Para tanto, no Capítulo 2 é feita uma revisão bibliográfica, exibindo as abordagens exploradas por esses e outros autores em temas correlatos, e como eles contribuíram para a construção deste assunto. É realizada também uma revisão dos conceitos básicos, a delimitação do escopo deste trabalho e a designação de uma notação comum, para facilitar o diálogo entre as diferentes fontes e abordagens.

No Capítulo 3, é apresentado o método de Monte Carlo, incluindo suas características, limitações e diferentes aplicações no cálculo de confiabilidade e disponibilidade. Essa seção será a base teórica na qual se fundamentará a solução para o problema principal do trabalho.

O código em Python escrito para o cálculo da frequência de acidente é retratado no Capítulo 4, juntamente com a validação dessa linguagem de programação para a resolução do problema. A sub-rotina de geração de tempos aleatórios (extremamente necessária para a simulação direta [10]) também é avaliada.

O Capítulo 5 é reservado para a aplicação do programa, primeiramente em casos exponenciais, e sem levar em conta falhas de reparo, como modelado em [8], para validar o código em um caso mais simples. Então, os resultados de [9], que modela casos mais complexos, serão comparados com o programa proposto.

A apresentação dos resultados no Capítulo 6 traz tabelas e gráficos mostrando os resultados do código em diversas condições, incluindo estudo do comportamento para valores extremos, que em [8] foi relatado como um problema para algumas expressões avaliadas.

As conclusões expostas no Capítulo 7 refletem as limitações e particularidades do programa, mas também apontam sugestões para desenvolvimento futuro de um jeito que casos mais gerais possam ser simulados.

2 Revisão Bibliográfica e Conceitos Básicos

Este capítulo tem o objetivo de apresentar os conceitos básicos necessários para a modelagem do problema, aproveitando-se dos conceitos já apresentados por outros autores na literatura sobre modelagem em Monte Carlo e métodos matemáticos aplicados em engenharia, em especial dos artigos e livros voltados para análises mais específicas com foco na engenharia nuclear.

A resolução do problema dependerá de conceitos estruturais das usinas nucleares de potência e de conceitos de análise de segurança. Como já dito no Capítulo 1, a notação a ser trabalhada será determinada de forma a se comunicar facilmente com os trabalhos já apresentados.

2.1 Revisão Bibliográfica

A forma básica do problema possui uma resolução analítica onde a indisponibilidade média do período entre testes é calculada de maneira exata, com taxas de falha, demanda e reparo não variantes no tempo. LEES [11] nos dá, de maneira exata e por simulação usando o método de Monte Carlo, resultados de indisponibilidade média do sistema dependente de suas taxas constantes no tempo. Em sua notação, calcula o que chama de "taxa efetiva de ocorrência de acidentes" (em inglês: *effective hazard rate*) η^*.

A comparação e análise de sensibilidade feita por OLIVEIRA et al. [8] com outros artigos que se propuseram a pesquisar a resolução do problema é bem abrangente. Nele, os autores resolvem analiticamente uma cadeia markoviana simples, representada na Figura 1, e avaliam a influência dos parâmetros no valor da taxa de ocorrência de acidentes da usina. É introduzida nesse artigo a diferença entre reparo com a planta offline e reparo com a planta online. Os resultados dessas duas formas de avaliação são representados por η_2 e η_{23}, respectivamente, por causa dos estados da cadeia markoviana que representam a indisponibilidade do sistema.

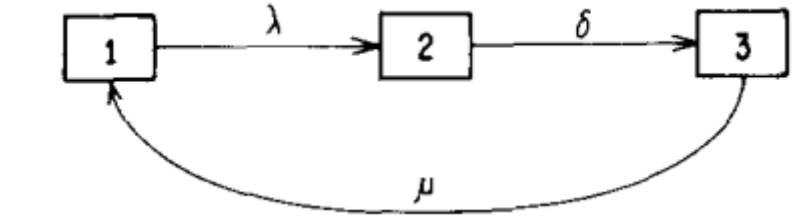

Figura 1 - Diagrama de transição de estados para um canal [8]

Para o reparo com a planta online, a demanda pode ocorrer tanto no estado 2 (falha não revelada) quanto no estado 3 (sistema em reparo). Para o escopo deste trabalho, só serão analisados os casos em que o reparo ocorra com a planta desligada, pelo motivo que será explicado adiante no capítulo.

Tanto os trabalhos de FRUTUOSO E MELO et al. [7] quanto o de OLIVEIRA [9] realizam estudos voltados para distribuições não-exponenciais de tempos em um canal de proteção. Em ambos, a distribuição de Weibull é utilizada para representar taxas de falha crescentes [7]. A diferença entre os dois trabalhos é o método que cada um utiliza: enquanto em [7] vemos o uso do método Monte Carlo para a resolução direta do problema, [9] introduz na cadeia markoviana uma variável complementar, desenvolvendo um sistema de equações integro-diferenciais parciais e ordinárias acopladas que são resolvidas através de método de diferenças finitas.

As simulações através do método Monte Carlo no campo da Engenharia Nuclear, apesar de possuírem uma história que se estende até os anos de 1940, ainda são uma poderosa ferramenta para a resolução de problemas numéricos. SHOOMAN [13], nos anos 60, desenvolvia programas para o cálculo de confiabilidade de sistemas através da simulação direta usando a transformada inversa. ZIO [10] é atualmente um dos autores que tratam de problemas de Engenharia de Confiabilidade e Análise de Riscos (aplicando, entre outros métodos, o Monte Carlo), e apresenta técnicas mais avançadas de cálculo, que podem ser úteis para futuras aplicações.

2.2 Conceitos Básicos

2.2.1 Sistemas de Proteção

Os sistemas de proteção de uma instalação nuclear têm a função de detectar e agir sobre um transiente que possa apresentar perigo para a integridade da usina. Em uma usina nuclear, seu objetivo principal é o de desligar o reator antes que as condições limite

para a operação do núcleo sejam ultrapassadas, e possa ocorrer o derretimento do mesmo. Em um sentido mais geral, existem inúmeros exemplos deste tipo de sistema em qualquer planta industrial, desde s*prinklers* para combate a incêndios até o *trip* de um reator em caso de alto fluxo de nêutrons.

As características de cada sistema de proteção também dependerão de seu arranjo lógico. Para alguns sistemas, são instalados 2 ou mais sensores em uma lógica de votação, ou seus sistemas de desligamento colocados em redundância, para aumentar a confiabilidade da detecção do transiente e de sua atuação sobre ele, assim como diminuir a probabilidade de um acionamento espúrio. Este trabalho foca principalmente em sistemas de proteção de canal único, mas a possibilidade de generalizar o programa para outros tipos de lógica será discutida no Capítulo 7.

Seguindo a terminologia descrita em [7] dos sistemas de proteção, existem 3 estados em que um canal pode se encontrar:

Canal em funcionamento: O transiente é detectado perfeitamente;

Falha não-revelada: A planta continua funcionando, porém o operador não tem conhecimento de que o sistema não está funcionando;

Falha revelada: A falha é detectada a partir de uma demanda não atendida para o ativamento do sistema de proteção, e o mesmo será posto em condição de reparo.

Em termos gerais, pode-se descrever um sistema com $i + j + k$ canais por um terno$< i, j, k >$, onde *i*, *j* e *k* representam o número de canais respectivamente nos estados em funcionamento, em falha não-revelada e em falha revelada. Em termos de um sistema com somente um canal, seus 3 estados possíveis serão $< 1,0,0 >$, $< 0,1,0 >$ e $< 0,0,1 >$.

Para uma descrição mais realista dos sistemas de proteção, também podemos aplicar o conceito de erro humano no reparo do canal de proteção. No caso, o canal que tenha sua falha revelada por uma demanda não atendida deve ser reparado. O reparo terá uma probabilidade γ de induzir falha e, como consequência, o canal não estará pronto para atender à próxima vez que for demandado.

Além disso, como ponto principal do trabalho, temos o fato de que o canal está envelhecendo. Na prática, isso significa que, com o passar do tempo, a sua taxa de falha tende a aumentar, acarretando maior probabilidade de falhas durante a operação. Em termos matemáticos, no entanto, significa que sua modelagem e resolução analítica não poderão ser feitos através de Cadeia Markoviana sem a utilização de métodos associados, como variáveis complementares ou o método dos estágios.

A Figura 2 mostra de maneira gráfica o problema como posto no artigo de FRUTUOSO E MELO et al [7], com todas as complexidades citadas acima sendo levadas em conta.

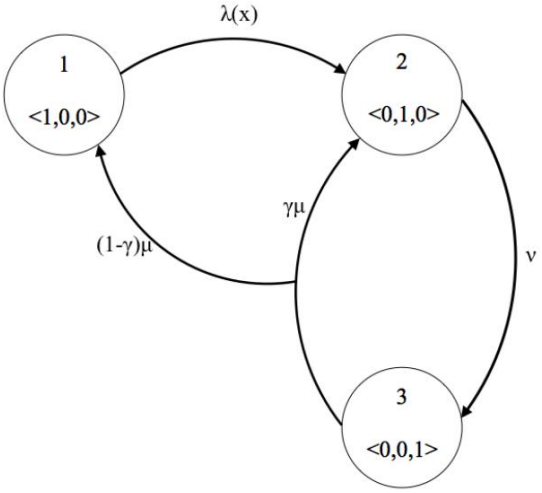

Figura 2 - Diagrama de Transição de Estado com taxa de falha variando no tempo e falha no reparo [9]

2.2.2 Frequência de Ocorrência de Acidentes

A frequência de ocorrência de acidentes (*plant hazard rate*) é uma taxa que quantifica quantos acidentes são previstos em um período de tempo definido para uma dada configuração do sistema. LEES [12], ao tratar de sistemas de proteção de plantas industriais, define a frequência de ocorrência de acidentes η como o produto da taxa de demanda pela indisponibilidade média do intervalo em que se deseja calcular. Ou seja, o acidente ocorrerá quando o sistema for demandado mas por algum motivo não estiver apto a responder corretamente à demanda.

A indisponibilidade média será representada pela letra *U* (do inglês *unavailability*) para seguir a nomenclatura comum à maioria dos artigos usados como referência. Seu significado é a fração de tempo total da missão de um dado sistema em que o mesmo encontra-se em falha não-revelada ou em reparo. Logicamente, seu valor dependerá da taxa de falha λ e da taxa de reparo μ do canal. Pode-se observar também que a indisponibilidade é afetada pela taxa de demanda υ, principalmente para valores >10/ano [8]. Logo, na notação vigente, teremos:

$$\eta = \upsilon U(\lambda, \mu, \upsilon) \tag{2.1}$$

Finalmente, precisamos definir o intervalo entre os testes do sistema de proteção, τ_p. Considera-se neste trabalho que, a cada missão de τ_p anos, a usina é desligada e o canal é testado, sem possibilidade de reiniciar a usina antes de um reparo perfeito. Tratando-se da resolução do problema através da cadeia, sua indisponibilidade seria referente à probabilidade de estar no estado de falha não-revelada (no caso de reparo com a planta *offline*) ou na probabilidade de não estar no estado de funcionamento (no caso de reparo com a planta *online*)[8].

3 *Método de Monte Carlo*

O método de Monte Carlo (MMC), de acordo com HAMMERSLEY & HARDCOMB [14], foi utilizado sistematicamente e em larga escala pela primeira vez para cálculos de blindagem de radiação durante o Projeto Manhattan. Isso faz com que o método tenha mais de 50 anos de história, sempre avançando junto dos progressos da computação. Sua aplicabilidade e flexibilidade atravessa campos como medicina, química, física e finanças, resolvendo os mais variados problemas.

Um exemplo clássico da lógica do método consiste na aproximação de π através da geração de pontos aleatórios dentro de um quadrado com um círculo nele inscrito. A razão entre a área do círculo e a área do quadrado é igual a π/4. Logo, se $N_{círculo}$ é o número de pontos dentro do círculo e $N_{quadrado}$ é o número de pontos dentro do quadrado, podemos aproximar o valor de π como:

$$\pi \approx 4N_{círculo}/N_{total} \tag{3.1}$$

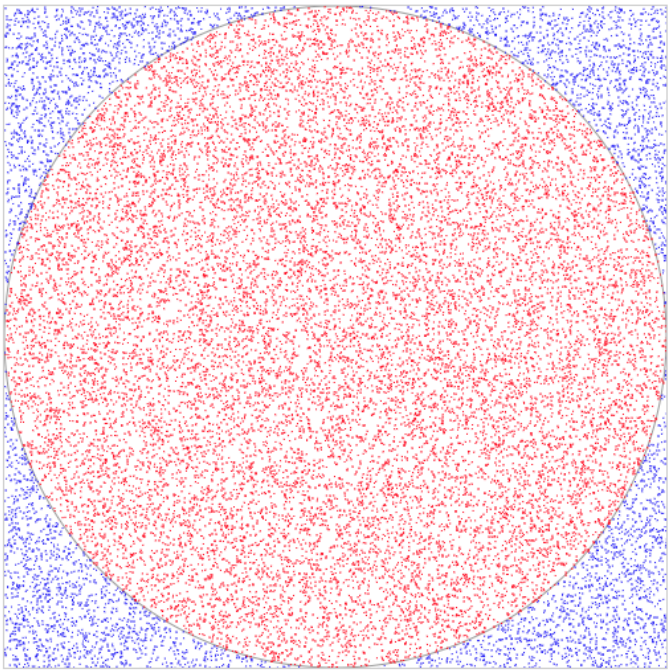

Figura 3 - Exemplo de simulação para obter valor de π [15]

O MMC apresenta um bom método de solução aproximada para problemas sem solução analítica, ou quando a mesma seja muito difícil de obter. As desvantagens do método são a dependência do gerador de números aleatórios, que deve ter um período (quantidade de números gerados antes de repetir a mesma sequência) e aleatoriedade apropriados. Além disso, para simulações complexas ou que exijam grande precisão, o tempo de cálculo também aumenta consideravelmente [10].

3.1 Descrição do Método de Monte Carlo

O MMC consiste na geração de números aleatórios em grande escala, descrevendo certa distribuição probabilística, para simular um fenômeno. Pode-se, por exemplo, usar o método para descrever o caminho aleatório de um nêutron em um reator para calcular sua reatividade. A distribuição dos tempos de falha e demanda dos sistemas de proteção de uma usina podem ser modelados a partir de distribuições de probabilidade conhecidas de acordo com as características desse sistema. Para isso, precisa-se antes definir como serão gerados números aleatórios e quais as distribuições escolhidas.

3.1.1 Geração de números aleatórios

Os números aleatórios (e os métodos para gerá-los) apresentam diversas aplicações nas ciências matemáticas e aplicadas. A natureza da aleatoriedade foge ao escopo do trabalho, mas a discussão sobre como obter resultados análogos (e a confiança que podemos ter neles) será tratada.

O resultado de fenômenos *verdadeiramente* aleatórios, como um cara-ou-coroa com uma moeda não-tendenciosa ou a contagem de emissões de uma fonte radioativa, leva a dois problemas principais: a reprodutibilidade dos experimentos nas mesmas condições fica comprometida; e o tempo de geração para cada número inviabiliza uma simulação em grande escala como necessita o MMC. As tabelas de números aleatórios foram criadas para contornar esses problemas, mas a demora para que o computador acessasse sua memória para retornar um número ainda deixava a desejar para simulações com muitos números [10].

Para tanto, a criação de funções para geração de número *pseudo*-aleatórios, no lugar de números verdadeiramente aleatórios, resolve o problema da reprodutibilidade dos experimentos e aumenta a velocidade de obtenção de números. Vários algoritmos foram criados para tal propósito, sendo para o interesse deste trabalho focar no algoritmo

de Mersenne Twister. Esse é o algoritmo utilizado pela linguagem de programação Pyhton, possui uma vasta literatura, um longo período de geração e cumpre critérios de aleatoriedade para algoritmos de geração de números pseudo-aleatórios [16].

Neste trabalho, a geração de números aleatórios é feita através das funções da classe *random* embutidas no Python 2.7.10, que os geram a partir de dados fornecidos pelo usuário ou por funções que se utilizam do conceito do método de amostragem por transformada inversa, como será explicado na sequência. Para esse programa, as funções *random.expovariate(lambd)* e *random.weibullvariate(alpha,beta)* funcionam gerando números pseudo-aleatórios seguindo respectivamente as distribuições exponencial e de Weibull, a partir de parâmetros que descrevem cada uma delas. A função "expovariate" recebe como entrada o valor da taxa de falha, ou o inverso da média da distribuição exponencial. A função "weibullvariate" recebe dois parâmetros: o parâmetro de forma θ e o parâmetro de escala *m*. O parâmetro de escala mede o envelhecimento do sistema: quanto maior seu valor, mais degradado estará. No limite, com $m=1$ a taxa de falha será constante, de valor igual ao inverso de θ. As funções retornam um valor aleatório, que segue a distribuição desejada. Já as funções da transformada inversa recebem, além dos parâmetros, uma semente, ou seja, um número real pseudo-aleatório gerado pelo Python com distribuição uniforme no intervalo [0,1[, e retornam um tempo de falha aleatório que segue a distribuição desejada.

Em termos de desempenho, comparou-se o tempo de processamento necessário para geração de tempos aleatórios partindo da mesma semente, utilizando as funções fornecidas pelo Python ("expovariate" e "weibullvariate") em comparação com as funções das transformadas inversas, apresentadas mais a frente no capítulo, para 1.000.000 de números gerados. As médias dos tempos gerados, até onde a precisão do computador pôde apresentar, foram as mesmas. Já o tempo de processamento foi avaliado para diferentes parâmetros de distribuição e diferentes sementes. Os resultados podem ser vistos na Tabela 1:

Tabela 1 - Tempo de processamento para diferentes funções, parâmetros e sementes

Sementes		Tempos (segundos)	
		Funções do Python	Transformada Inversa
Exponencial			
0,145669255104130	(λ=1 ano^{-1}) (λ =0,001 ano^{-1}) (λ =10.000 ano^{-1})	0,887113 0,937885 0,924840	0,839753 0,857321 0,846184
0,454927004514021	(λ=1 ano^{-1}) (λ =0,001 ano^{-1}) (λ =10.000 ano^{-1})	0,889686 0,849928 0,894227	0,855923 0,781664 0,831772
0,770783805659022	(λ=1 ano^{-1}) (λ =0,001 ano^{-1}) (λ =10.000 ano^{-1})	0,838361 0,862130 0,857846	0,796624 0,824217 0,846468
Weibull			
0,145669255104130	(θ=1 ano, m=1) (θ=1 ano, m=0,01) (θ=1 ano, m=10.000)	1,231775 1,309497 1,295963	0,695203 0,828095 0,843976
0,454927004514021	(θ=1 ano, m=1) (θ=1 ano, m=0,01) (θ=1 ano, m=10.000)	0,889686 1,370254 1,423671	0,702321 0,768265 0,866731
0,770783805659022	(θ=1 ano, m=1) (θ=1 ano, m=0,01) (θ=1 ano, m=10.000)	1,218880 1,408227 1,362394	0,694939 0,836064 0,834009

As sementes testadas foram geradas aleatoriamente pela função *random.random()*, onde *()* significa que a função não necessita de argumentos - situação na qual o programa utiliza o tempo no relógio do computador para gerar o número aleatório. Os testes foram feitos através do compilador online Repl.it, e estão disponíveis no link [17].

Vemos claramente que, em todos os cenários, o cálculo através da transformada inversa tem velocidade de processamento superior aos geradores do Python, mesmo em casos com valores extremos e resultados com muitas casas decimais, mantendo em todas as situações a precisão requerida para a resolução do problema, e, portanto, a transformada inversa será utilizada na subrotina geradora de tempos.

3.1.2 Modelagem da distribuição

O comportamento das taxas de falha, reparo ou demanda pode ser constante ou variável no tempo. No caso da taxa constante, dizemos que certo componente ou sistema encontra-se em sua vida útil. Já para o caso da taxa de falha crescente (decrescente) com o tempo, teremos um sistema ou componente em fase de envelhecimento (mortalidade infantil), cujos tempos de falha podem em muitos casos ser descritos por uma distribuição de Weibull com parâmetro de forma maior que 1 (menor que 1) [18]. Essas distinções entre as taxas de falha são facilmente vistas na Figura 4:

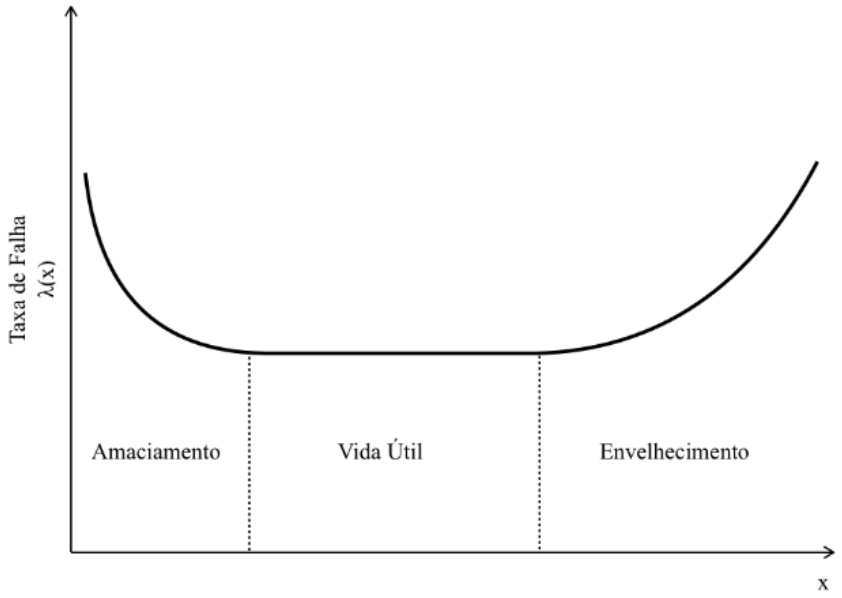

Figura 4 - **“**Curva da Banheira”, reprodução de [9]

Para a simulação direta dos tempos de falha, reparo e demanda, usaremos o método da Amostragem por Transformada Inversa, como descrita por ROSS [19] e ZIO[10]. O método é utilizado para gerar um tempo que siga as características de sua distribuição a partir do inverso de sua função de distribuição cumulativa F(t). Em termos práticos, explicita-se o tempo *t* da distribuição cumulativa em função do valor de F(t), e sorteia-se um valor de F(t) entre 0 e 1, que chamaremos de R. No caso da geração de tempos exponenciais teremos as equações a seguir para F(t) e para a transformada inversa, respectivamente:

$$F(t) = 1 - e^{-\lambda t} \tag{3.2}$$

$$t = -\frac{1}{\lambda} ln(1 - R) \tag{3.3}$$

Nas equações acima, *R* é uma variável aleatória distribuída uniformemente gerada pela função *random.random()*. No caso da distribuição de Weibull, o mesmo procedimento pode ser realizado e as equações abaixo podem ser deduzidas para F(t) e sua transformada inversa:

$$F(t) = 1 - e^{-(\frac{t}{\theta})^m} \tag{3.4}$$

$$t = \theta[-ln(1 - R)]^{1/m} \tag{3.5}$$

A notação acima difere ligeiramente da notação de [10], mas representa a exata função que o Python utiliza para a geração do tempo aleatório. A Eq. (3.3), no entanto, apresenta um problema para a geração de dados. Os valores de tempos baixos que seguem a distribuição de Weibull são maiores que os tempos gerados pelo caso exponencial. Essa distorção não representa o que acontece com um componente na vida real, quando no início de sua vida útil ainda terá uma taxa de falha constante. O envelhecimento do sistema só acontecerá a partir de um certo período t_o de utilização, e se quisermos modelar sistemas que envelheçam dessa maneira precisaremos que a curva da geração de tempos da Weibull coincida com a do caso exponencial em t_o com o intuito de dar continuidade a uma nova distribuição mista. A partir dos parâmetros da vida útil do sistema, devemos encontrar $x_0 \in [0,1[$ onde o valor do tempo será t_o, e uma constante *C* que, acrescentada ao tempo gerado a partir da distribuição de Weibull com x_0, resulte em uma

$$t_0 = -\frac{1}{\lambda} ln(1 - x_0) = \theta[-ln(1 - x_0)]^{1/m} + C \tag{3.6}$$

Podemos, a partir da igualdade esquerda da fórmula, achar o valor de x_o:

$$x_0 = 1 - e^{-\lambda t_0} \tag{3.7}$$

Aplicando esse valor na igualdade a direita para encontrar C, obteremos:

$$C = t_0 - \theta[-ln(1 - x_0)]^{1/m} \tag{3.8}$$

As curvas abaixo foram geradas para exemplificar o deslocamento da distribuição de Weibull. Para facilitar a visualização no gráfico, os valores de λ=0,8/ano, θ=1 ano, m=1,5 e t_o =1 ano foram escolhidos por serem próximos aos reais, mas apresentam curvas com maior distanciamento. Para esses parâmetros, x_0=0,550671 e C=0,138226.

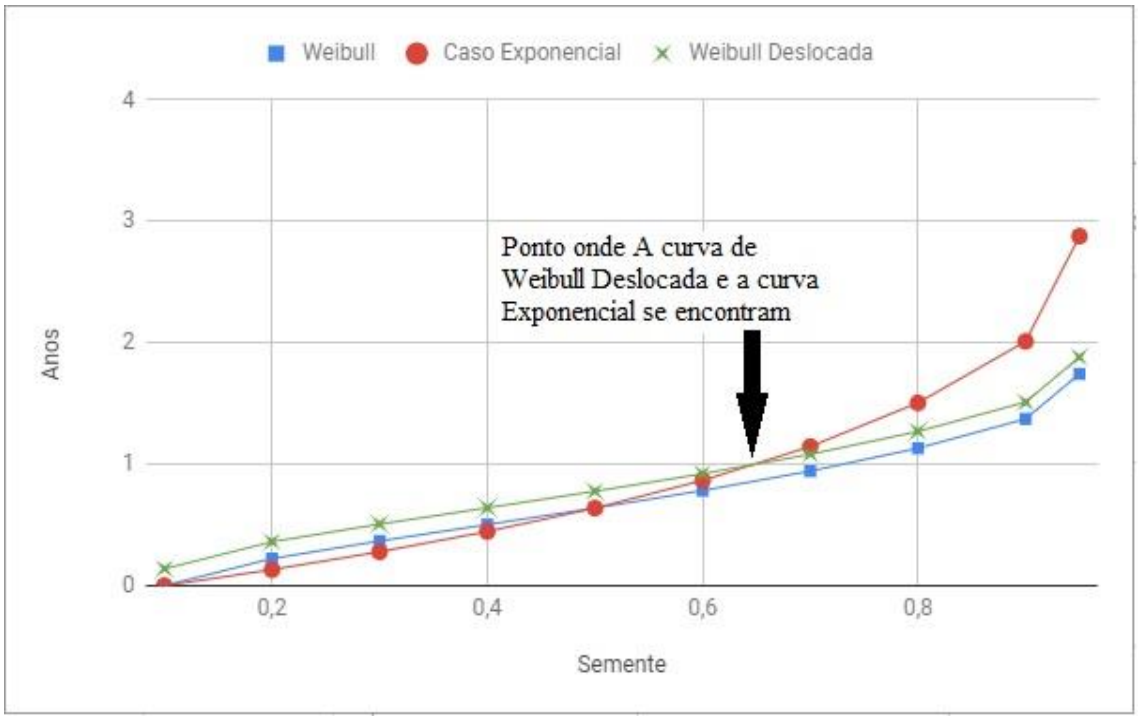

Figura 5 - Comparação entre os tempos gerados pelas transformadas inversas em função da semente x no intervalo [0,1[

Logo, a geração de tempos aleatórios de falha para sistemas em envelhecimento receberá 4 parâmetros: λ, θ, m e t_o. Caso a semente x gerada aleatoriamente pelo programa seja menor que x_0 o tempo gerado seguirá a distribuição exponencial com taxa de falha constante igual a λ. Caso contrário, seguirá a distribuição de Weibull.

4 Aplicação do Método de Monte Carlo ao Problema

Neste Capítulo será utilizada toda a base teórica já apresentada nos capítulos anteriores para compor o programa que simulará o problema. A escolha da linguagem de programação Python para a implantação do código se deu por sua sintaxe e vocabulário relativamente fáceis, em comparação com linguagens como C [20], e por uma extensa coleção de bibliotecas, que permite uma vasta gama de utilizações para essa linguagem, além de ser uma das linguagens mais populares do mundo da programação [21]. Seu desempenho de fato não é tão robusto quanto o de linguagens compiladas, mas sua velocidade está no mesmo nível de outras linguagens utilizadas na engenharia como MATLAB, e à frente do Mathematica em análises estatísticas [22].

4.1 Pseudocódigo

Resumidamente, o programa cria "histórias" que simulam os comportamentos reais do sistema de proteção e da demanda sobre esse mesmo sistema. Em cada uma dessas histórias, é contabilizado o tempo em que o sistema ficou fora de operação (ou *downtime*, em inglês), e comparado com o tempo total de operação da usina para que seja encontrada a indisponibilidade média da usina no período avaliado. A cada história, são sorteados novos tempos aleatórios de demanda, falha e reparo, seguindo suas respectivas distribuições, até termos um número suficientemente alto de simulações para que se alcance uma certa confiabilidade nos resultados.

Em cada história, temos a seguinte sequência de eventos:

1) Os tempos de demanda e falha são gerados e comparados. Como uma demanda com o canal em funcionamento não acarreta em descobrimento de uma falha, são gerados sucessivos tempos de demanda até que uma demanda aconteça após a falha. O estado do canal é modificado de <1,0,0> para <0,1,0> (ou, como aparece no código, de 1 para 2). Caso o tempo de falha tenha atingido o tempo total do teste, encerra-se a história com uma

disponibilidade de 100%. Caso contrário, segue-se para a próxima instrução;

2) Contabiliza-se o tempo entre a falha do canal e a demanda que a revelou, sendo somado ao contador *DT* (do inglês *downtime*, ou "tempo morto"). Caso a demanda tenha ocorrido depois do tempo total testado, será contabilizado o período entre a falha e o fim do período testado e a história é terminada. Caso contrário, segue-se para a próxima instrução;
3) Se a probabilidade de falha no reparo é maior que 0, então é sorteado se o reparo obteve sucesso ou não. Caso falhe, o estado continua como falha não revelada, gera-se um tempo de reparo, e retorna-se para a geração de um novo tempo de demanda a partir do final do reparo. Caso o reparo seja bem-sucedido, segue-se para a próxima instrução;
4) Ao tempo de falha do canal, soma-se o tempo de reparo gerado. Como a usina encontra-se "em parada para reparo", o tempo até o seu "religamento" não é contabilizado no contador *DT*. Gera-se novamente tempos de demanda e falha. Caso eles ainda não tenham ultrapassado o tempo de teste, segue-se para a instrução 1.

A geração de tempos aleatórios pode ser feita através da função *GerarTempo(dist,par)*, que recebe como argumentos a distribuição do tempo que deve ser gerado e seus parâmetros. A ideia é simplificar, ao custo de tempo de processamento, a mudança entre as diferentes distribuições, de modo que o programa atenda ao maior número de configurações possível. A outra opção é a geração dos tempos diretamente no corpo da função principal, o que aumenta a sua eficiência computacional por não necessitar o acesso a uma subrotina externa, mas limita o cálculo a um caso específico do problema. Na Tabela 2 temos a comparação do tempo de execução, em segundos, das duas funções.

Tabela 2 - Tempo de processamento para diferentes funções e número de histórias

Número de Histórias	Função com subrotina (segundos)	Função sem subrotina (segundos)
100	0,003829	0,002935
1000	0,037494	0,030893
10000	0,365964	0,299979
100000	3,746855	3,194608
1000000	37,023367	30,862621

Claramente, a função funciona mais rapidamente sem a subrotina de geração de tempos aleatórios, e será utilizada para casos em que as distribuições de falha, demanda e reparo sejam exponenciais. Porém, ela será mantida no corpo do programa para utilizações onde seja necessária a utilização de outros tipos de distribuição de tempos. Além da Weibull e exponencial, que são aplicadas neste trabalho, estão disponíveis na subrotina as distribuições normal e lognormal para aplicações futuras.

A Figura 6 demonstra graficamente um exemplo de história. A partir do tempo t=0, são gerados o tempo T_F de falha e o tempo T_D de demanda. No fim de T_F, vemos que sistema passa do estado 1 para o 2. O tempo em que o sistema fica no estado 2, como já mencionado anteriormente, é o contabilizado no final das histórias para o cálculo da indisponibilidade. Em seguida o sistema passa por um reparo bem-sucedido de duração T_R, passando pelo estado 3. Voltando para o estado 1 novamente, serão gerados novos tempos de demanda e de falha a partir do ponto onde o sistema volta a operar. Como, no exemplo, o tempo de falha ultrapassa o período entre testes τ_p, não é nem necessário gerar mais tempos de demanda. Todos os casos possíveis estão previstos para que nenhum

tempo no estado 2 fiquem de fora da contabilidade total, muito menos que seja contado mais que o necessário.

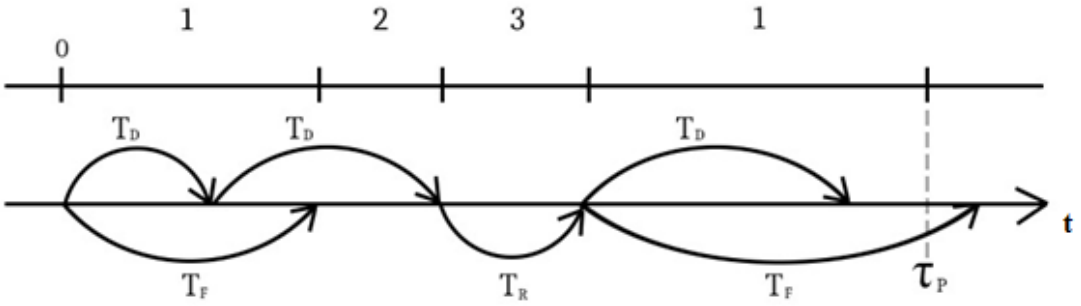

Figura 6 - Exemplo de história

4.2 Número de Histórias

O assunto a ser tratado nessa seção diz respeito ao número de simulações necessárias para a obtenção de um resultado confiável. A importância de estudar o número mínimo de histórias diz respeito não só à economia computacional (e consequentemente ao tempo de processamento do programa) como também à confiabilidade que podemos dar ao resultado. O MMC demanda um alto poder de processamento computacional, dada a quantidade de números aleatórios gerados, e sua eficiência é diretamente ligada ao poder de processamento da máquina que está lendo o código. Tendo definido a maneira mais eficiente de gerar os tempos aleatórios, necessários para o código, definiremos então o número mínimo de histórias para que a simulação convirja.

SHOOMAN [13] afirma que, para as simulações utilizando o MMC, o erro varia de acordo com a seguinte relação empírica:

$$\varepsilon_{MC} \sim 1/\sqrt{n} \tag{4.1}$$

Ou seja, o erro máximo relativo é proporcional ao inverso da raiz quadrada do número *n* de simulações. Isso implica ao mesmo tempo em uma vantagem e uma fraqueza do método para resolução de problemas complexos. A vantagem é que, se tratado corretamente, o modelo irá convergir para o valor exato da solução analítica, se o número de simulações tender ao infinito. É evidente que a economia computacional e o poder de processamento dos computadores atuais tornam proibitiva uma simulação com número

altíssimo de histórias, quando bastaria um número suficientemente alto para que a margem de desvio do resultado do código em relação ao resultado analítico fosse aceitável.

Por outro lado, sua desvantagem é a velocidade com que o erro se aproxima de zero. Derivando a Eq. (4.1) em relação a *n*, vemos uma desaceleração da redução do erro quando o número de simulações é muito grande. O incremento do número de histórias não leva a uma redução considerável quando esse número já é muito grande, o que torna o MMC dependente de um tempo cada vez maior de processamento para obter valores cada vez menores de erro.

GOLDFELD & DUBI [23] apresentam uma fórmula geral para a estimativa do número de histórias necessárias para alcançar certo grau de erro máximo no resultado:

$$NPS = 10^4/(P\varepsilon^2) \tag{4.2}$$

Na fórmula (4.2), NPS é o número estimado de histórias necessárias para atingir um grau ε de erro (em porcentagem) para um evento com probabilidade de ocorrência igual a P. Vale ressaltar que o artigo [23] foi publicado em 1987, quando a capacidade de processamento de computadores científicos não era da mesma magnitude dos computadores pessoais atuais, e, de acordo com o artigo, uma simulação de um milhão de histórias (que, no compilador online Repl.it, leva em média pouco mais de 8 segundos) demorava meia hora.

Para a faixa de valores tratada por OLIVEIRA et al. [8] para o período de testes τ_p igual a 1 ano e taxa de reparo μ igual a 52 ano^{-1}, a frequência de ocorrência de acidentes não assume valores menores que 0,004. Aplicado à fórmula, para um erro de no máximo 1%, obtemos uma recomendação para 2,5x10^6 histórias.

5 *Validação do modelo de solução*

Neste capítulo trataremos da conformidade do modelo proposto para os casos analisados anteriormente por outros autores. Inicialmente, para a validação do código, utilizaremos o caso exponencial que é passível de uma resolução analítica exata por cadeia markoviana, assim como tratado por OLIVEIRA et al. [8], comparando os seus valores com os obtidos pelo código proposto. Nessa etapa, não será levado em conta o tempo computacional de cálculo, somente a comprovação da convergência do código para o resultado analítico. Para tanto, utilizamos $7{,}5 \times 10^6$ histórias em cada simulação, segundo a lógica da Eq. (4.1), com um número grande de simulações para reduzir a variância do resultado, como pode ser conferido na Tabela 3.

Tabela 3 – Comparação de valores entre o MMC e os resultados de [8]

λ (ano^{-1})	υ (ano^{-1})	η [8] (ano^{-1})	η (ano^{-1})	Erro Relativo (%)
0,1	0,1	4,6824E-03	4,6885E-03	-0,130
	1,0	3,5756E-02	3,5746E-02	0,028
	10,0	8,9060E-02	8,9295E-02	-0,264
	15,0	9,2598E-02	9,2511E-02	0,094
1,0	0,1	3,5756E-02	3,5758E-02	-0,007
	1,0	2,8259E-01	2,8272E-01	-0,045
	10,0	8,1394E-01	8,1390E-01	0,005
	15,0	8,6467E-01	8,6504E-01	-0,043
	0,1	8,9060E-02	8,9063E-02	-0,003
	1,0	8,1394E-01	8,1398E-01	-0,005
	10,0	4,3610	4,3609	0,003
	15,0	5,1970	5,1977	-0,014

Todos os resultados foram obtidos através do código para o período de testes τ_p igual a 1 ano e taxa de reparo μ igual a 52 ano^{-1}, sem considerar falha de reparo e considerando apenas o reparo *offline* do sistema, em uma faixa de valores para a taxa de demanda e falha selecionados diretamente de [8]. Pode-se observar na Tabela 3 que o resultado gerado pelo código claramente converge para o resultado analítico no caso exponencial, e pode ser aplicado nessa faixa de valores sem maiores problemas. O erro relativo dos casos descritos acima pode ser associado à geração dos números pseudo-aleatórios, intrínsecos ao MMC.

A comparação com [9], por sua vez, já apresenta dificuldades por tratarem com visões conceitualmente diferentes. No primeiro, o autor trata da taxa de falha do sistema como um todo como sendo constante até um tempo t_o *no calendário* e crescente a partir daí, o que de fato pode ser aplicado em situações onde, por exemplo, o ambiente se degrada de uma maneira a gerar piora nas condições de funcionamento do sistema. Não faz diferença se um componente substituto começa a atuar logo antes de t_o, pois ele irá entrar em envelhecimento mesmo sendo novo, apesar de que na prática é mais comum que seja feito um reparo de tal componente. No trabalho aqui apresentado, cada componente novo na história terá que atingir o tempo t_o *de funcionamento* para a partir daí começar a envelhecer. Isso é o mesmo que dizer que o reparo, quando bem-sucedido, nos dará sempre um componente tão bom quanto um novo, o que, por sua vez, não necessariamente retrata a realidade. Espera-se, então, que os valores de frequência de acidentes obtidos através do código aqui apresentado sejam ligeiramente menores do que os obtidos em [9].

6 *Resultados*

Neste capítulo serão exibidos alguns resultados calculados através do código, de forma a demonstrar as características do método desenvolvido neste trabalho. Para tanto, em primeiro lugar devemos escolher as variáveis a serem utilizadas no programa. A escolha será feita com base em valores semelhantes aos apresentados em [9] para facilitar a comparação entre os dois estudos, mesmo já tendo sido explicitada a diferença de metodologia utilizada pelos dois programas (pelo mesmo motivo, as nomenclaturas dos parâmetros podem diferir, e recomenda-se a utilização da relação de nomenclaturas após o índice).

A Tabela 4 mostra os valores utilizados para cada parâmetro, que foram inseridos no código a fim de realizar os cálculos apresentados a seguir. O programa foi executado uma vez para cada ponto; a distância entre dois pontos consecutivos foi fixada em 0,1 ano para a visualização do comportamento temporal da frequência de acidentes.

Tabela 4 – Valores dos parâmetros utilizados como dados de entrada do programa

Parâmetro	**Valor**
υ	10 ano^{-1}
μ	52 ano^{-1}
τ_p	2 anos
γ	10%
λ	1 ano^{-1}
θ	1 ano
m	2,5
t_o	1 ano
Número de simulações	$2{,}5 \times 10^6$

Os valores escolhidos se assemelham aos de uma usina nuclear típica, de acordo com [8]. Os cálculos para a comparação entre o caso exponencial e o caso com envelhecimento demoraram cerca de 16 minutos para $2{,}5 \times 10^6$ simulações em cada ponto, realizados online através do site Repl.it. Caso rodados no Python, demoram aproximadamente 10 minutos em um MacBook Pro 2013, com 4 GB de memória RAM. Os resultados são apresentados na Figura 7.

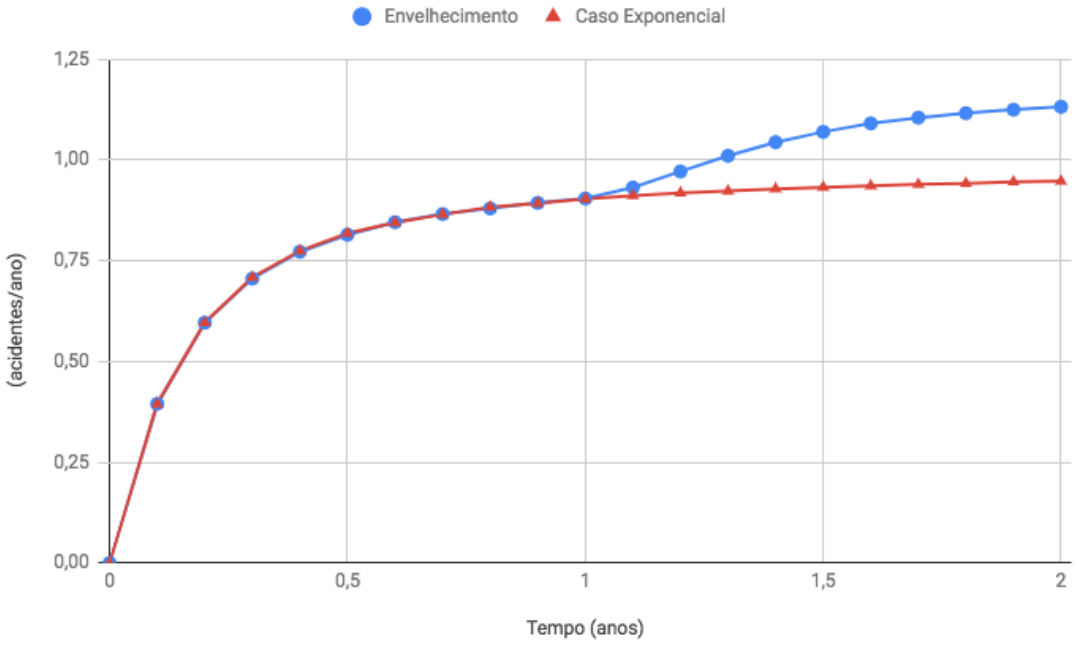

Figura 7 - Comparação entre caso exponencial e caso com envelhecimento

É flagrante que, para tempos menores do que o tempo que leva o sistema de proteção a sofrer o processo de envelhecimento, os valores da frequência de acidentes irão coincidir com os valores do caso de taxa de falha constante (t<1 ano no gráfico). Após esse período, as curvas se separam mostrando nitidamente o aumento da frequência de acidentes em sistemas envelhecidos. No caso dos valores utilizados, a diferença chega a quase 20% em relação ao caso exponencial, ao final de 2 anos. A aleatoriedade dos resultados faz com que haja uma diferença mesmo nos casos em que as frequências de acidente deveriam coincidir, não passando, no entanto, de 0,41%, como visto na Tabela 5.

Tabela 5 – Apresentação dos valores obtidos pelo código

Tempo (ano)	Envelhecimento (acidentes/ano)	Caso Exponencial (acidentes/ano)	Diferença Percentual (%)
0	0,0000	0,0000	-
0,1	0,3948	0,3942	0,16
0,2	0,5960	0,5962	-0,03
0,3	0,7054	0,7083	-0,41
0,4	0,7719	0,7744	-0,33
0,5	0,8140	0,8174	-0,41
0,6	0,8451	0,8442	0,11
0,7	0,8654	0,8643	0,12
0,8	0,8793	0,8828	-0,39
0,9	0,8930	0,8918	0,13
1	0,9039	0,9027	0,14
1,1	0,9311	0,9110	2,20
1,2	0,9709	0,9177	5,79
1,3	1,0100	0,9224	9,49
1,4	1,0436	0,9278	12,48
1,5	1,0692	0,9319	14,73
1,6	1,0901	0,9352	16,56
1,7	1,1044	0,9389	17,63
1,8	1,1154	0,9412	18,51
1,9	1,1242	0,9453	18,93
2	1,1312	0,9470	19,45

Para esse caso, ainda vale notar que o valor obtido para t = 2 anos é ligeiramente inferior ao valor obtido por [9]. No final do capítulo anterior foi explicado o porquê de o valor esperado ser menor utilizando o código proposto aqui.

Partindo agora para uma análise de sensibilidade do programa, é apresentada na Fig. 8 uma variação de taxas de demanda do sistema de proteção com o intuito de analisar o comportamento e evolução da frequência de acidentes alterando esse parâmetro. Todos os outros dados de entrada do programa foram mantidos nos três casos.

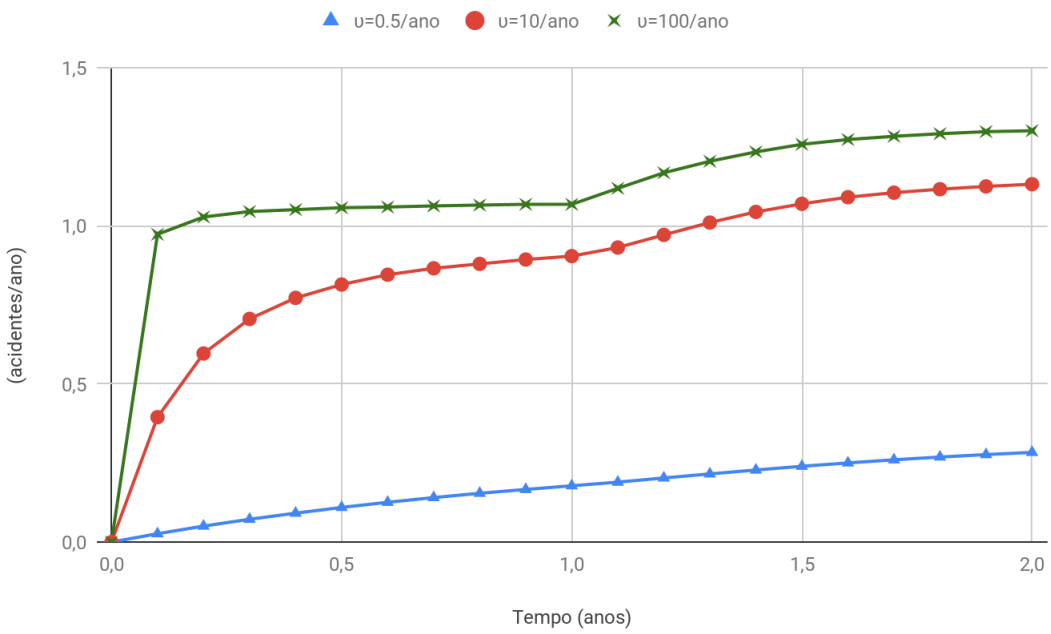

Figura 8 - Comparação da frequência de acidentes para diferentes taxas de demanda

O comportamento das curvas de acordo com a variação da taxa de demanda segue o padrão esperado, aumentando conforme aumenta υ. A forma da curva vista na Fig. 7 também se manteve, mesmo que a mudança na curva devido ao envelhecimento após t = 1 ano seja imperceptível no gráfico para valores baixos de taxa de demanda.

Como mais um teste de sensibilidade do modelo, veremos agora como o parâmetro de forma m influencia o valor da frequência de acidentes. O fator de forma m pode ser interpretado, no caso da geração de tempos de falha para o sistema, como a velocidade ou força do envelhecimento sobre o desempenho do sistema. Pode-se ver, na Fig. 9, em escala logarítmica para facilitar a visualização, os resultados do gerador de tempos aleatórios para diferentes valores de m.

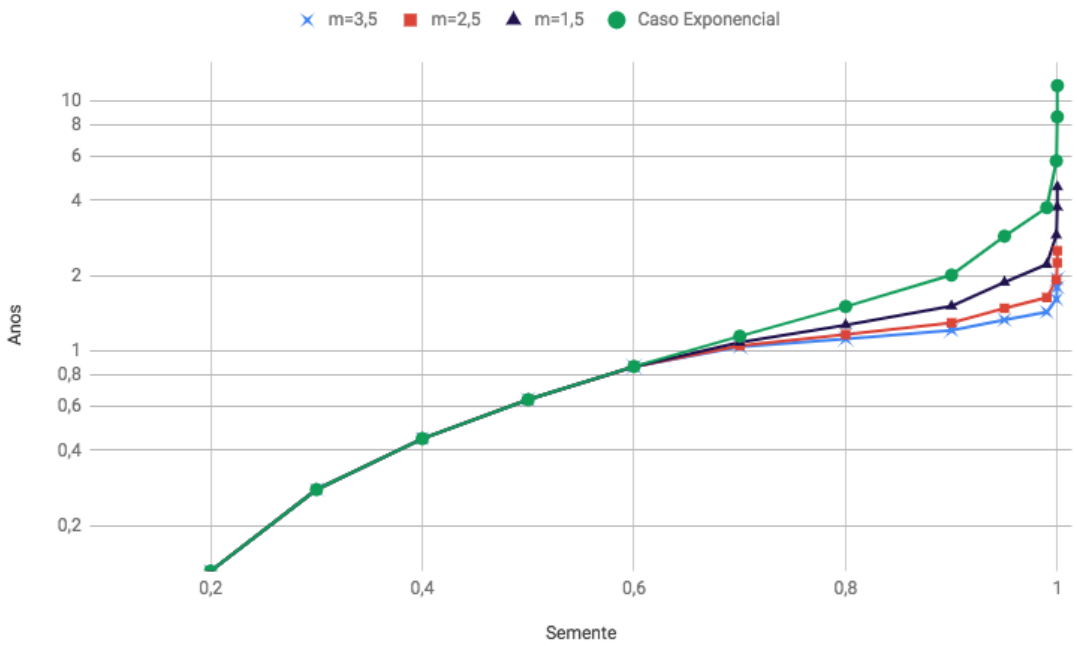

Figura 9 - Tempos gerados pela subrotina geradora de tempos de falha em função da semente, para diferentes *m*

Como os tempos gerados são menores enquanto o fator *m* aumenta, podemos concluir que a frequência de acidentes será mais alta para sistemas com um *m* elevado. De fato, essa intuição é confirmada pelos resultados apresentados na Fig. 10.

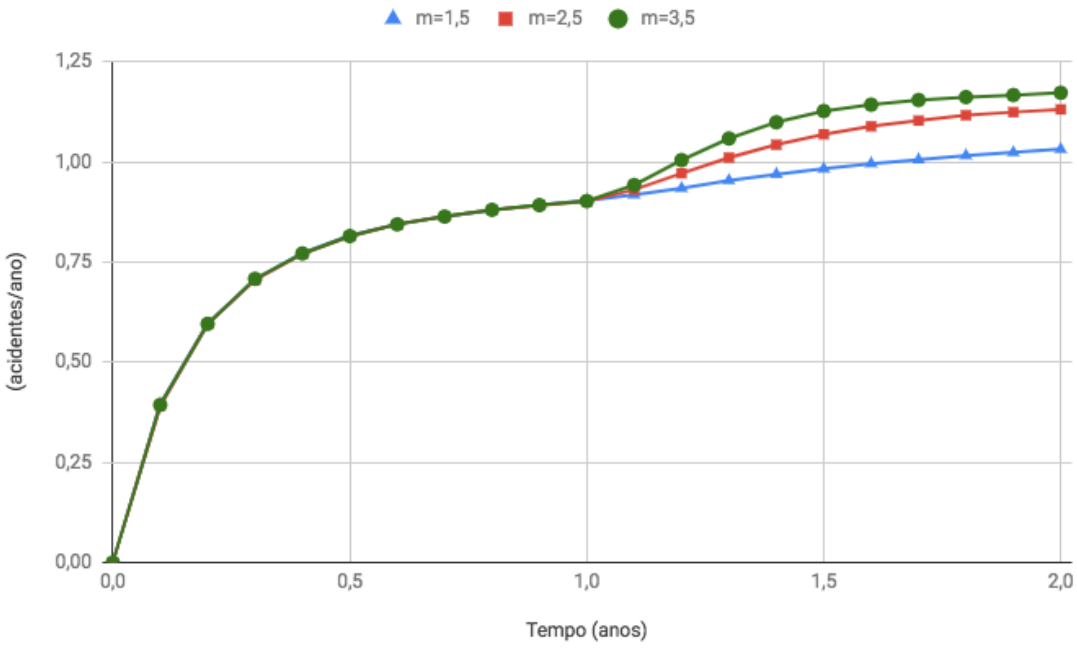

Figura 10 - Frequência de acidente por tempo, para diferentes fatores de forma *m*

E, finalmente, passando para a análise da interferência da variação da taxa de falha inicial λ e o fator de escala (vida característica) θ na frequência de acidentes, precisamos primeiro dissecar a relação entre esses dois parâmetros. Fica claro, dentro do escopo do trabalho, que só faz sentido tratar como um caso de *envelhecimento* quando há, de fato uma piora das condições do sistema de proteção, levando a um aumento da indisponibilidade do sistema. Para tanto, não faz sentido trabalhar em um mesmo cálculo com um fator de escala que gere, dentro do programa, uma média de tempos iguais ou superiores aos obtidos pela geração de tempos com o sistema em vida útil. Podemos então fazer uma limitação de que o fator de escala seja igual ou inferior ao inverso da taxa de falha inicial. Essa limitação garante que haja um aumento da frequência de acidentes após t_o.

Na Fig. 12 podem ser vistas as curvas geradas para três casos, onde $\theta = \frac{1}{\lambda}$ varia entre 0,5, 1 e 10 anos.

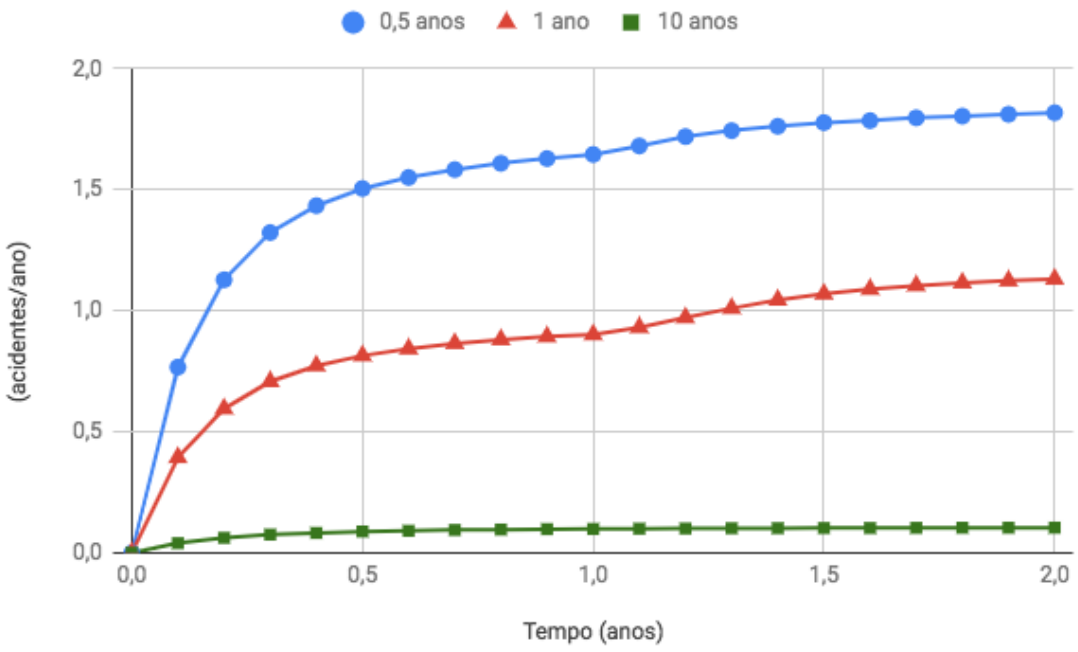

Figura 11 - Frequência de acidente por tempo, para diferentes valores de θ

7 Conclusões

O método desenvolvido neste trabalho foi construído com base em conceitos já bem sólidos, e apresenta resultados dentro do esperado, com uma faixa de erro pequena e com um tempo de processamento bom. Se aplicado, o código pode servir de base para cálculos de frequência de acidentes de usinas em envelhecimento que desejem estudar uma extensão de vida, como Angra 1 e 2, vitais para a economia brasileira, provavelmente o farão. Não obstante, outras áreas industriais que sofram situações análogas a simulada neste trabalho também poderão ser beneficiadas.

Para tanto, o caminho a ser seguido pelo código é a extensão de seu escopo de um sistema de proteção para *n* canais de proteção, modelando uma situação mais parecida com situações reais da indústria. Além disso, outra prática bem estabelecida é a de sistemas de proteção com lógica de votação em seus alarmes. Isso significa que um alarme ou uma ação só serão desencadeados caso um número k<n de canais acuse um problema ou transiente. Pequenas modificações no código podem contabilizar essas configurações mais complexas e mais realistas, contando inclusive com outras variáveis como probabilidade de falha em perigo e probabilidade de ativação espúria.

Apesar de não ter sido aplicado a nenhum caso estudado no trabalho, a subrotina que gera os tempos de falha e demanda também aceita outras distribuições, como normal e lognormal. Essa característica também poderá ser melhor estudada mais adiante em casos cujas distribuições de falha sejam essas.

Referências

[1]https://www-pub.iaea.org/MTCD/Publications/PDF/RDS-2-38_web.pdf, Acesso em: 01 nov. 2018

[2]http://www.world-nuclear.org/information-library/facts-and-figures/reactor-database.aspx Acesso em: 01 nov. 2018

[3] https://unfccc.int/sites/default/files/english_paris_agreement.pdf, Acesso em: 02 nov. 2018

[4]https://www.forbes.com/sites/jamesconca/2012/06/10/energys-deathprint-a-price-always-paid/#3a997150709b , Acesso em: 21 jan. 2019

[5]https://www.iaea.org/newscenter/news/going-long-term-us-nuclear-power-plants-could-extend-operating-life-to-80-years, Acesso em: 03 nov. 2018

[6]https://www.nrc.gov/reading-rm/doc-collections/fact-sheets/fs-reactor-license-renewal.html, traduzido pelo autor, Acesso em: 04 nov. 2018

[7] Frutuoso e Melo, P. F. & G. Teixeira, D & C. M. Alvim, A. (2017). A Monte Carlo evaluation of the accident rate of a plant equipped with an aging single-channel trip device. AIP Conference Proceedings. 1863. 560060. 10.1063/1.4992743.

[8] Oliveira, L.F., Netto, J.D.A., 1987, "Influence of the Demand Rate and Repair Rate on the Reliability of a Single-Channel Protective System", Reliability Engineering,17, pp.267-276.

[9] Oliveira, L.G., Cálculo da Frequência de Ocorrência de Acidente de um Sistema de Proteção de um Canal de uma Instalação Nuclear sujeito a Envelhecimento – Rio de Janeiro: UFRJ/ESCOLA POLITÉCNICA, 2018

[10] Zio, E., The Monte Carlo Simulation Method for System Reliability and Risk Analysis (Springer, London, 2013).

[11] Lees, F.P., 1982, "A General Relation for the Reliability of a Single-Channel Trip System", Reliability Engineering, 3(1), p.1.

[12] Lees, F.P., 1980, Loss Prevention in the Process Industries - Hazard Identification, Assessment and Control, 2. ed, Oxford, Butterworth Heineman

[13] Shooman, M.L., Probabilistic Reliability: An Engineering Approach, Robert E. Krieger Publishing Co., Malabar, FL, 1990

[14] Hammersley, J. M., D. C. Handscomb: Monte Carlo Methods. Methuen & Co., London, and John Wiley & Sons, New York, 1964

[15] https://academo.org/demos/estimating-pi-monte-carlo/, Acesso em: 07 nov. 2018.

[16] https://docs.python.org/2.7/library/random.html, Acesso em: 14 jan. 2019.

[17] https://repl.it/@bnwarth/Testes-dos-Geradores-de-numeros-aleatorios/, Acesso em: 14 jan. 2019.

[18] Notas de Aula da matéria de Engenharia de Confiabilidade do professor Paulo Fernando Frutuoso e Melo

[19] Ross, S.M., Simulation, Elsevier, New York, 2012

[20] Fangohr H. (2004) A Comparison of C, MATLAB, and Python as Teaching Languages in Engineering. In: Bubak M., van Albada G.D., Sloot P.M.A., Dongarra J. (eds) Computational Science - ICCS 2004. ICCS 2004. Lecture Notes in Computer Science, vol 3039. Springer, Berlin, Heidelberg

[21] Bissyandé, T.F., Thung, F., Lo, D., Jiang, L. and Réveillère, L., "Popularity, Interoperability, and Impact of Programming Languages in 100,000 Open Source Projects," *2013 IEEE 37th Annual Computer Software and Applications Conference*, Kyoto, 2013, pp. 303-312. 10.1109/COMPSAC.2013.55

[22] S. Borağan Aruoba, Jesús Fernández-Villaverde, A comparison of programming languages in macroeconomics, Journal of Economic Dynamics and Control, Volume 58, 2015, Pages 265-273, ISSN 0165-1889.

[23] Goldfeld, A. & Dubi, A. (1987). Monte Carlo methods in reliability engineering. Quality and Reliability Engineering International. 3, 83 - 91.