Arthur Sorrentino Ferreira
Higor David Oliveira
Tulio Brunoro de Souza

Segundo Trabalho Prático

Trabalho acadêmico desenvolvido como obtenção parcial de nota na disciplina de Algoritmos Numéricos I do Curso de Graduação em Engenharia Elétrica da Universidade Federal do Espírito Santo

Vitória ES
2020

Sumário

1 INTRODUÇÃO 3

2 OBJETIVOS 4

3 METODOLOGIA 5

4 RESULTADOS E AVALIAÇÃO 8

REFERÊNCIAS 16

APÊNDICE A - CÓDIGO *Solve2_1* 16

APÊNDICE B - CÓDIGO *Solve2_1_1* 18

APÊNDICE C - CÓDIGO *meu_geornd* 19

APÊNDICE D - CÓDIGO *Solve3_1* 21

APÊNDICE E - CÓDIGO *Solve3_1_1e3_1_2* 23

APÊNDICE F - CÓDIGO *Solve3_2_1e3_2_2* 27

APÊNDICE G - CÓDIGO *main* 28

1 Introdução

Desde seu surgimento, o Universo vem sendo moldado por eventos aleatórios e suas consequências. Chama-se se evento aleatório aqueles que produzem resultados diferentes, mesmo que submetidos as mesmas condições. Sendo assim, torna-se impossível prever o próximo resultado de tal evento, mesmo que todos os parâmetros sejam controlados, define-se como espaço amostral o conjunto de todos os resultados possíveis para um dado experimento. (MAYER, 2020)

Para quantizar a chance de um evento ocorrer, utiliza-se de um número de zero a um, chamado de probabilidade. Este define a facilidade ou dificuldade de um dado resultado ocorrer em meio aos demais resultados do espaço amostral. (MAYER, 2020)

Semelhantemente, a distribuição de probabilidade é a associação feita entre os resultados numéricos de um dado evento. Ela tem como objetivo tentar modelar eventos aleatórios, visto que conseguem prover um comportamento geral para tal evento. De posse dessa informação, torna-se possível analisar a concentração das possibilidades. (MAYER, 2020)

As variáveis analisadas na probabilidade podem ter natureza contínua ou discreta, tendo tratamento diferente dependendo do caso analisado. Uma variável é dita discreta quando possui uma quantidade finita de valores possíveis. De modo análogo, uma variável contínua possui uma quantidade infinita de valores para assumir. (PPGEP, 2020)

Diversas distribuições são conhecidas na literatura. Algumas notáveis são o modelo de Bernoulli, o modelo binomial e modelo de Poisson para distribuições discretas e o modelo de Gauss ou modelo normal para distribuições contínuas. (MAYER, 2020)

2 Objetivos

Baseado na ideia de fazer afirmações sobre o processo de histórico de falhas de algum processo de natureza estocástica, este trabalho tem como objetivo uma série de objetivos pontuais, entre eles:

- A geração de dados, sujeitos a uma distribuição estatística, com o objetivo de simular o aparecimento de certos eventos;
- Estimar os parâmetros associados a uma certa distribuição, usando as amostras disponíveis;
- Calcular probabilidades, integrando distribuições em caso de variáveis contínuas, ou somando, em caso de variáveis discretas;
- Fazer afirmações sobre o processo, por exemplo, dizer quando um equipamento falhará com 80% de probabilidade.

Além dos objetivos pontuais, podemos citar o aprendizado de ferramentas de engenharia moderna, como o Octave, principal programa de desenvolvimento deste trabalho, e Matlab, que possui sintaxe de código similar ao Octave.

3 Metodologia

Para a realização deste trabalho, foi utilizado o software GNU Octave, um software livre empregado para a solução de problemas numéricos, simulações e outras diversas capacidades. Tais objetivos são alcançados fazendo-se uso de uma linguagem de programação científica dotado de diversos algoritmos matemáticos, bem como funções para visualização de tais resultados.

Para alcançar os objetivos propostos neste trabalho, foram estabelecidos vários desafios, que exigem a utilização de diferentes técnicas e teorias. Para a solução de cada um desses desafios, foram criados códigos individuais e modularizados. Então, por fim, para fácil execução, foi criada uma *main.m*, que chama os métodos e os organiza para que apareçam na ordem que os desafios aparecem na documentação. Todos os códigos usados estão no final deste documento.

O primeiro problema proposto a ser resolvido trata-se de reproduzir o gráfico da curva da Função Massa de Probabilidade (FMP), que consiste na associação do valor de uma probabilidade dada a sua ocorrência de forma aleatória. A função é responsável por corresponder para cada valor de x do espaço de resultados do evento aleatório para um y real, que corresponde a probabilidade de x dentro do espaço X. Além disso, o desafio também inclui a reprodução do gráfico da Função Distribuição Acumulada (FDA), que consiste no acumulativo da FMP, ou seja, é o acumulado das probabilidades de acontecimento do evento aleatório para o valor definido de x. A função de FMP é definida como,

$$P(X = x) = (1 - P)^n P \tag{3.1}$$

e FDA,

$$F(x) = \int_{-\infty}^{x} p(u)du \tag{3.2}$$

Para esta primeira solução, foi desenvolvido o código *Solve2-1.m*. O código se estrutura em três partes diferentes: definição das funções de FMP e FDA, geração dos dados para diferentes possibilidades, ($P_1 = 0.2$, $P_2 = 0.5$, $P_3 = 0.8$), e, por fim, a impressão dos dados na forma de três gráficos: FMP, FDA e FDA em escala de log no eixo y.

O segundo problema proposto trata-se de desenvolver uma função capaz de retornar quantos insucessos se tem antes de um sucesso n vezes dado uma certa probabilidade de uma evento aleatório acontecer. Esse código pode ser útil para se ter uma estimativa de,

sabendo a possibilidade de falha em um equipamento, em média o quanto se espera uma falha antes de um sucesso ou quantos sucessos se espera antes de uma falha.

Para esta segunda solução conforme orientado, foi criado uma função, *meu_geornd(P, n)*, que gera uma distribuição geométrica, filtra de acordo com a probabilidade dada e então faz a contagem de quantos insucessos ocorreram antes do primeiro sucesso. Esses dados são armazenados em um vetor, até n posições e, uma vez completo o vetor, é usado como retorno da função. Por último, através de um *if()*, é escolhido se será feito a avaliação da probabilidade estimada que o vetor de saída "equivaleria".

O terceiro problema proposto foi a reprodução das curvas de distribuição de Weibull, distribuição bastante usada na engenharia de confiabilidade, buscando analisar o tempo de vida médio e as taxas de falhas de dada população. Para gerar as curvas da distribuição, foi criado o código *Solve3-1.m*. Ele funciona de forma similar à solução do primeiro problema: Primeiramente é feito a declaração da função de Weibull, a Função de Densidade de Probabilidade (FDP), e da sua FDA; depois são feitos os cálculos relacionados a função; e por fim, os plots. Para a geração dos dados são usados os parâmetros de $\lambda = 1.0$ para $k = 0.5,\ 1.0,\ 1.5$ e 5.0 e $\lambda = 2.0$ para $k = 1.0\ e\ 5.0$. A função de FDP de Weibull pode ser definida como,

$$p(x; k, \lambda) = \frac{k}{\lambda}(\frac{x}{\lambda})^{k-1} e^{-(x/\lambda)^k} \tag{3.3}$$

e sua integral, FDA,

$$F(x; k, \lambda) = 1 - e^{-(x/\lambda)^k} \tag{3.4}$$

O quarto problema é também relacionado a distribuição de Weibull. Como nem sempre é possível obter a solução analítica da distribuição, foi proposto conseguir a FDA usando métodos computacionais de integração. Então, para a integração, foram usados duas técnicas diferentes, uma por Trapézios, alterando o número de triângulos para cada iteração da função e outra, pelo método de Simpson 1/3, alterando também o número de iterações para cada cálculo do valor da integral. Por fim, para efeitos de comparação, é proposto fazer a comparação dos valores em $x = 1$ e dos gráficos para até $x = 2.5$. Os valores usados na distribuição são $\lambda = 1.0$ e $k = 5.0$. Para resolver tal problema, foi criado o código *Solve3_1_1 e 3_1_2.m*. De forma similar ao primeiro caso, este código também executa as etapas de definição dos métodos, cálculos e por fim os plots. Existe somente o adicional desse método pedir a tabela de comparação, que se encontra um pouco antes do plot dos gráficos. Como todo o código está comentado, os detalhes podem ser vistos diretos no código, nos apêndices desse trabalho.

O quinto e então último problema proposto se trata de tentar obter uma reestimativa dos parâmetros da distribuição de Weibull por meio de um processo iterativo, dado um conjunto de amostras inicial de uma distribuição de Weibull gerada aleatoriamente com

parâmetros $\lambda = 1.0$ e $k = 5.0$ iniciais. Para gerar as amostras, foi usada a função pronta *wblrnd-octave*. Para conseguir a reestimativa usou-se as funções abaixo:

$$k^{(j+1)} = \phi(k^{(j)}) \quad \& \quad \frac{1}{\phi(k)} = \frac{\overline{x^k ln(x)}}{\overline{x^k}} - \overline{ln(x)}; \tag{3.5}$$

$$\hat{\lambda} = \sqrt[k]{\overline{x^k}}; \tag{3.6}$$

4 Resultados e Avaliação

Com a implementação das soluções dos problemas propostos foram obtidos vários resultados. Começando pelo primeiro problema, a reprodução de gráficos de FMP e FDA para diferentes probabilidades, temos os gráficos abaixo, conforme Figuras 1, 2 e 3.

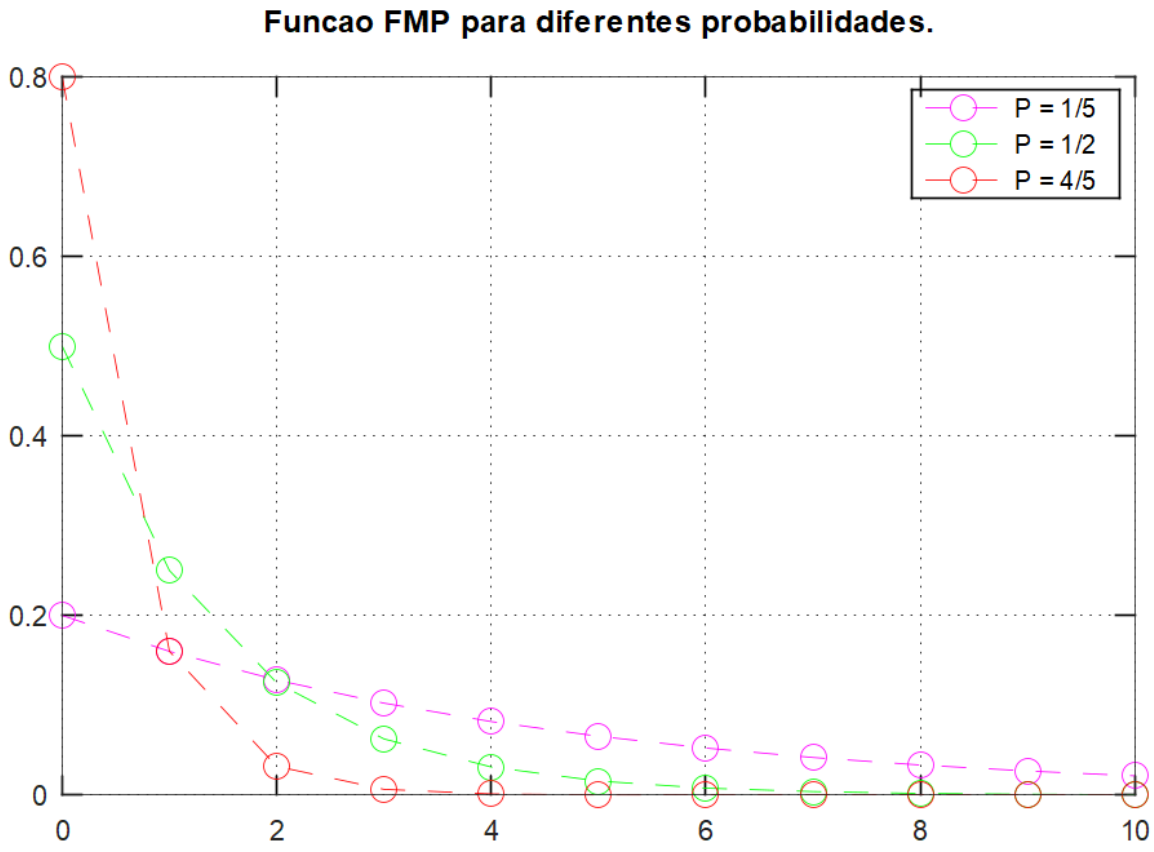

Figura 1 – Gráfico da Função FMP para diferentes probabilidades com eixo y decimal

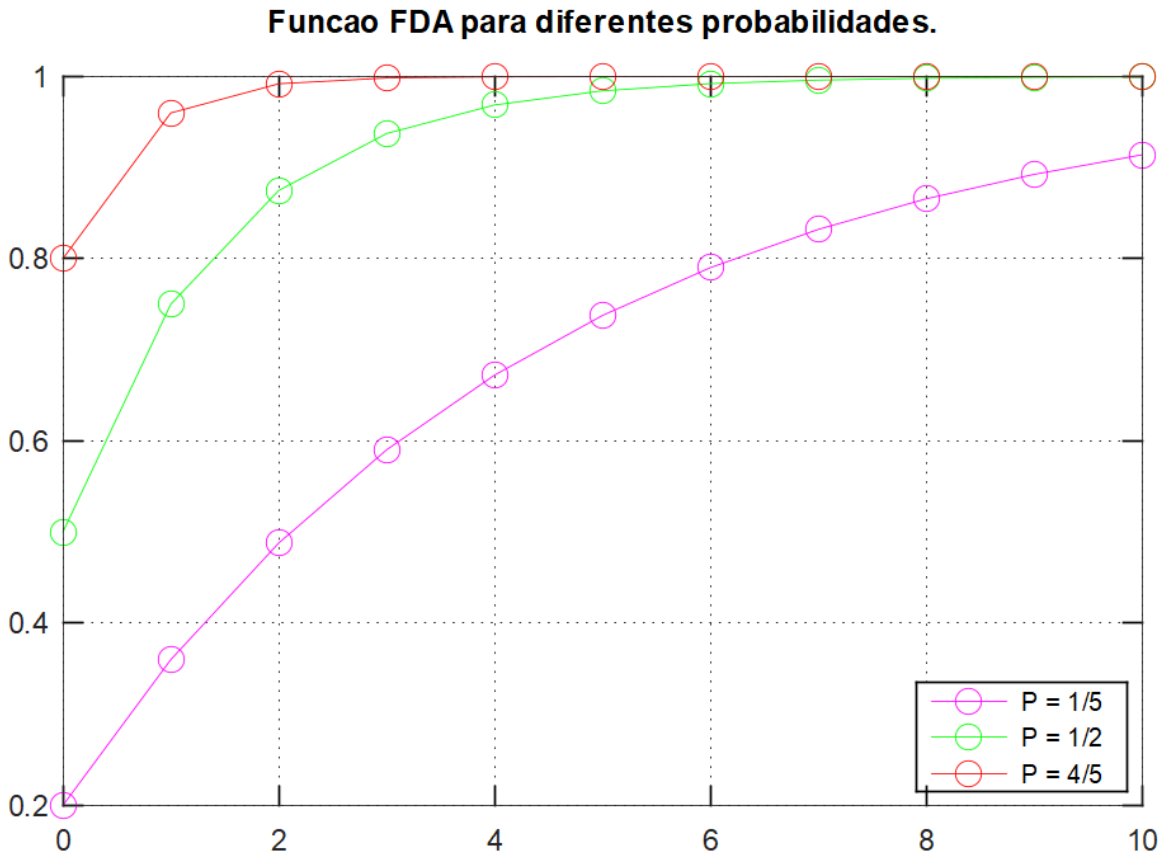

Figura 2 – Gráfico da Função FDA para diferentes probabilidades com eixo y decimal

É possível observar que os dois gráficos acima se aproximam bem dos da Wikipédia, que serviram de referência. (WIKIPEDIA, 2020a) Assim, podemos dizer que o desafio proposto foi concluído. Além disso, para analisar a curva característica da FMP, foi plotado o gráfico na escala de log em y. O gráfico obtido se encontra abaixo, conforme Figura 3.

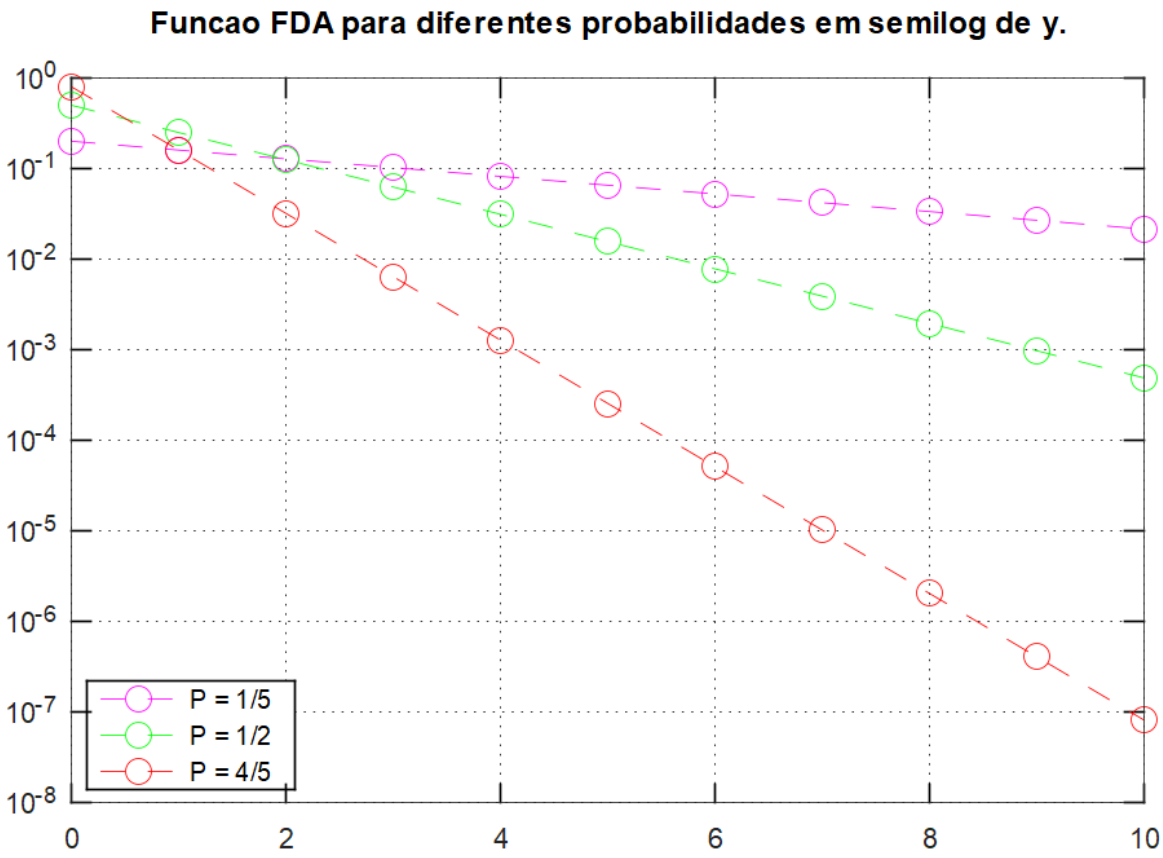

Figura 3 – Gráfico da Função FDA para diferentes probabilidades com eixo y logarítmico

Colocando a escala de log em y é possível perceber que a curva tem comportamento exponencial de primeira ordem, pois, uma vez aplicado o log, obteve-se uma reta para cada uma das possibilidades tratadas.

Continuando, temos então os resultados do desafio proposto de reprodução das curvas de Distribuição de Weibull abaixo, conforme Figuras 4 e 5.

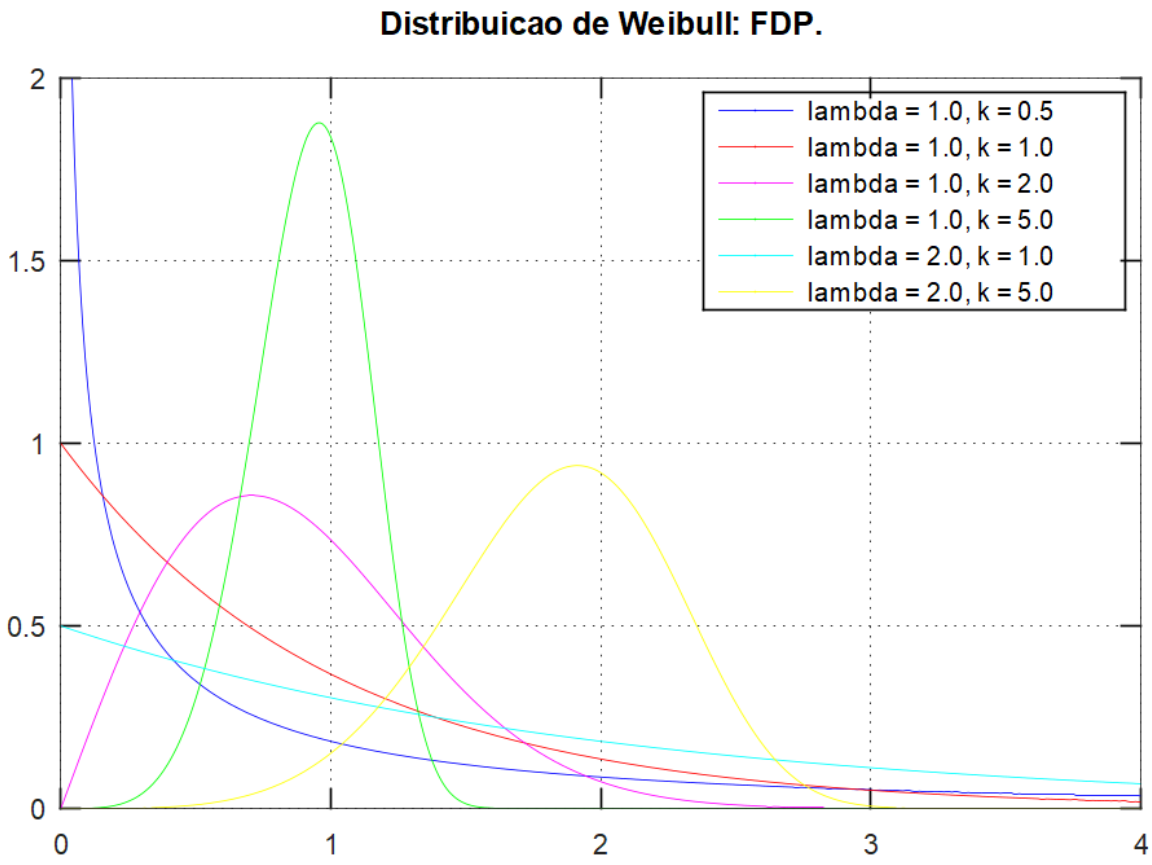

Figura 4 – Gráfico da FDP da Distribuição de Weibull

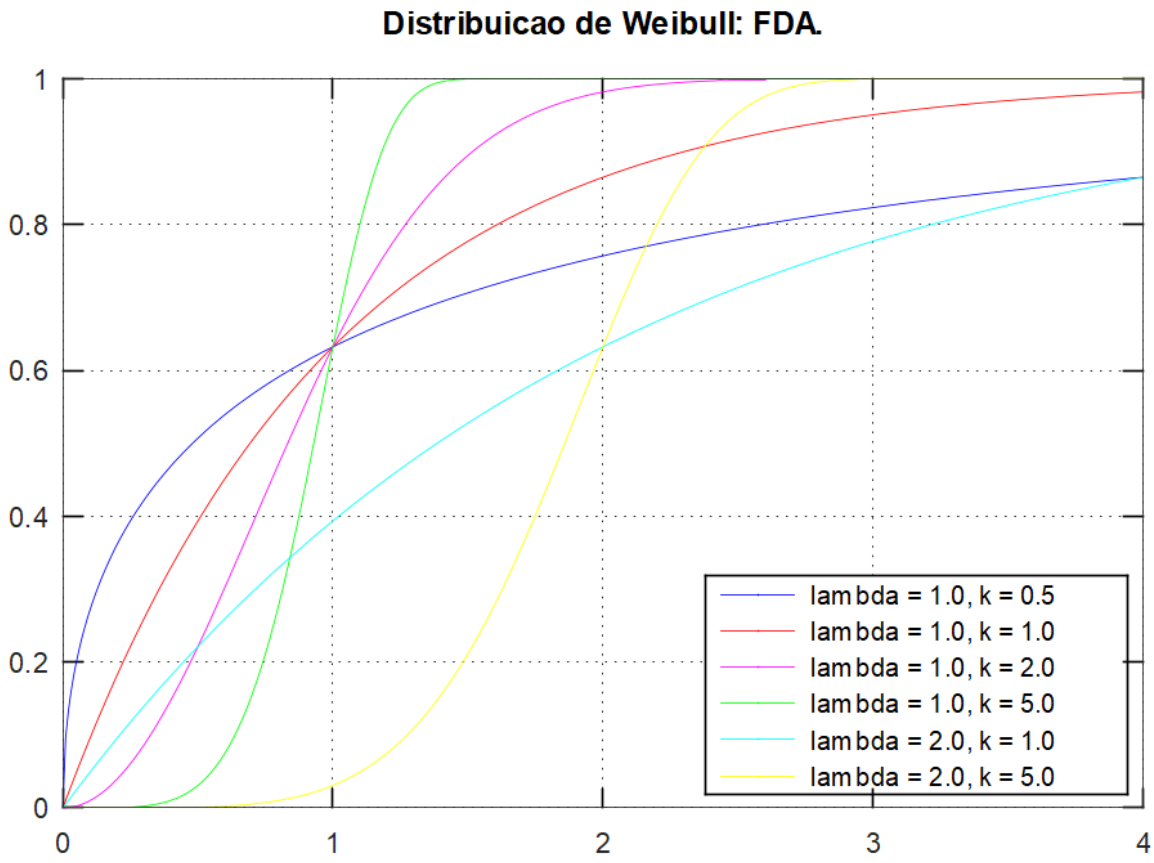

Figura 5 – Gráfico da FDA da Distribuição de Weibull

Essas são as curvas características da distribuição de Weibull para diferentes valores de Λ e k. Comparando os gráficos obtidos com os gráficos da Wikipédia, usados como referência, percebe-se que eles são idênticos.(WIKIPEDIA, 2020b) Podemos então considerar o objetivo do desafio proposto concluído.

Depois da tentativa de reprodução da distribuição de Weibull por meio da integração, temos, primeiro pelo método de Trapézios, conforme Figura 6:

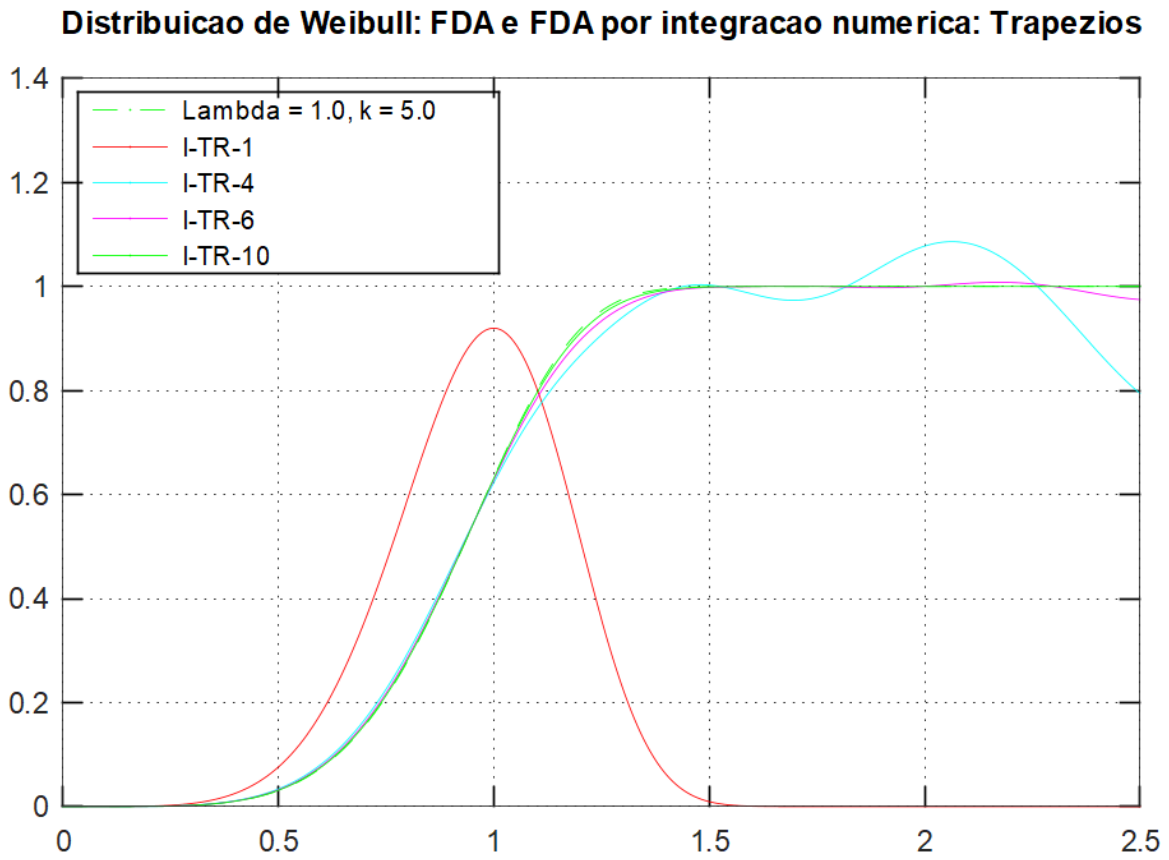

Figura 6 – Gráfico da FDA da Distribuição de Weibull e aproximações usando o método dos Trapézios

E então, pelo método Simpson 1/3, conforme Figura 7.

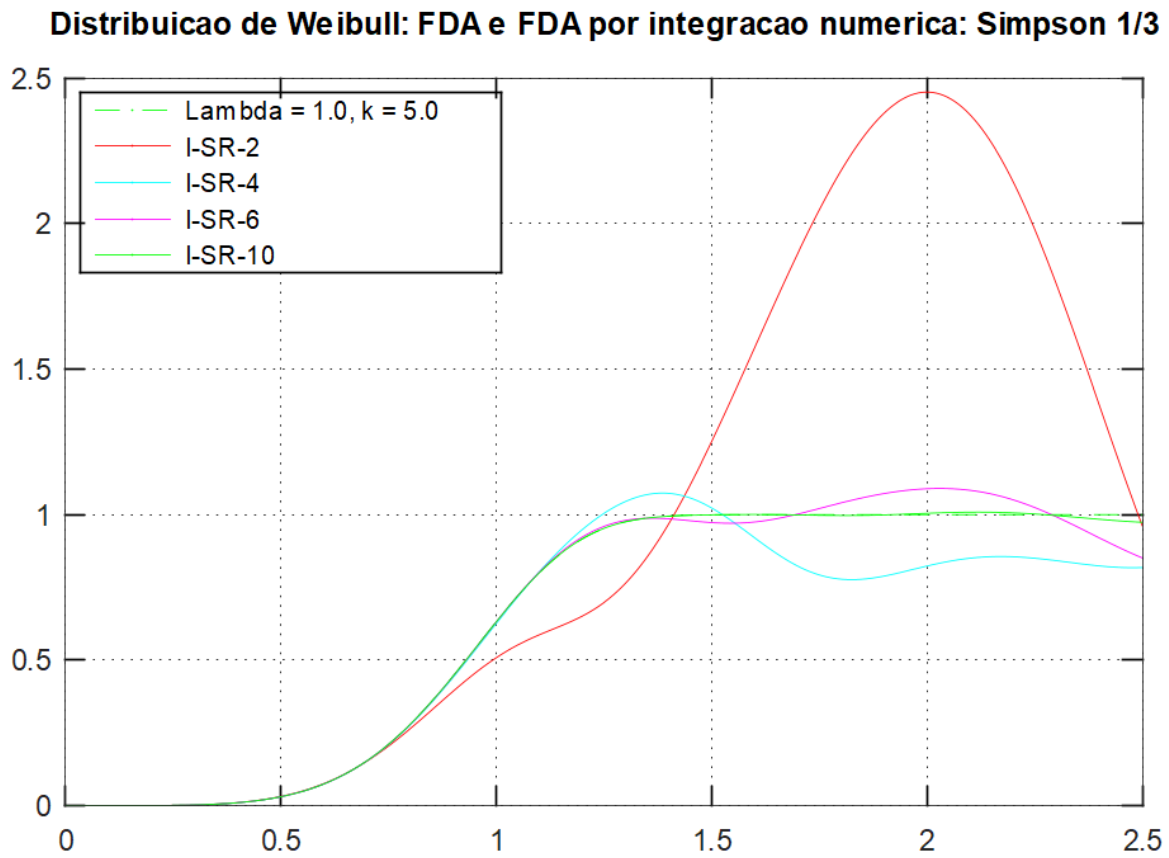

Figura 7 – Gráfico da FDA da Distribuição de Weibull e aproximações usando o método de Simpson 1/3

Pelos gráficos é possível observar que os métodos com menor número de iterações na integral, *I-TR-1* e *I-SR-2*, não obtiveram curvas próximas a curva real da FDA de Weibull, extrapolando para baixo no caso de Trapézios e extrapolando para cima no caso de Simpson. Observando os outros métodos, é possível notar que os métodos com 4 (*I-TR-4* e *I-SR-4*) e 6 iterações (*I-TR-6* e *I-SR-6*) obtiveram as curvas mais próximas da real, mas ainda variando muito, uma vez que o valor da função estabiliza em $x = 1.5$. E então, ambos os métodos de Trapézios e de Simpson para 10 iterações (*I-TR-10* e *I-SR-10*) obtiveram resultados bem próximos da curva real.

Para complementar essa análise, vamos analisar o valor para $x = 1$, ponto no meio da curva dos gráficos. Com o programa obtivemos a tabela abaixo:

```
----------------------------------------
Valor exato em x = 1        6.321e-01
Metodo        Valor         Erro
----------------------------------------
I_TR_1        9.197e-01     2.876e-01
I_TR_4        6.225e-01     -9.639e-03
I_TR_6        6.278e-01     -4.288e-03
I_TR_10       6.306e-01     -1.537e-03
I_SR_2        5.085e-01     -1.236e-01
I_SR_4        6.262e-01     -5.910e-03
I_SR_6        6.319e-01     -2.286e-04
I_SR_10       6.321e-01     1.109e-05
----------------------------------------
```

Figura 8 – Tabela obtida em $x = 1$ para cada método de integração, seu valor e o erro.

Os resultados mostrados confirmam o que foi dito anteriormente, onde os métodos com poucas iterações na integral apresentam erros muito grandes e o valor do erro desce consideravelmente aumentando o número de iterações.

E por fim, para a reestimativa dos valores de λ e k, como resultado é obtido um valor diferente toda vez que se roda o código, já que a amostra é aleatória. Sendo assim, monta-se uma tabela com valores de retorno de 5 iterações diferentes.

#	λ	k
0	1.00	5.0
1	1.00	5.22
2	1.00	4.81
3	1.00	5.39
4	1.00	4.80
5	0.99	4.86

Tabela 1 – Retorno do script de Solve3-2-1e3-2-2.

Assim, observa-se que apesar do valor ser diferente para toda vez que o programa roda, os valores da estimação são próximos dos valores reais da amostra ($\lambda = 1.0$ e $k = 5.0$).

Referências

MAYER, F. de P. *Variáveis Aleatórias e Distribuições de Probabilidade*. 2020. UFPR. Disponível em: <http://leg.ufpr.br/~fernandomayer/aulas/ce001n-2016-01/05_Variaveis_aleatorias/05_Variaveis_Aleatorias.pdf>. Acesso em: 06 dez 2020. Citado na página 3.

PPGEP. *Distribuições de Probabilidade*. 2020. UFRGS. Disponível em: <http://www.producao.ufrgs.br/arquivos/disciplinas/489_estaind005_distprob.pdf>. Acesso em: 06 dez 2020. Citado na página 3.

WIKIPEDIA. *Cumulative distribution function*. 2020. Wikipedia. Disponível em: <https://en.wikipedia.org/wiki/Cumulative_distribution_function>. Acesso em: 06 dez 2020. Citado na página 9.

WIKIPEDIA. *Weibull distribution*. 2020. Wikipedia. Disponível em: <https://en.wikipedia.org/wiki/Weibull_distribution>. Acesso em: 06 dez 2020. Citado na página 12.

Apêndice A - Código "Solve2_1"

```
% Trabalho 2 da matéria de Algoritmos Númericos para Engenharia feito em
% Octave.
% O trabalho é a reprodução de curvas estastíticas características,
% como FDA, FMP
% Usando Symbolic v2.9.0 e SymPy v1.5.1.
% Por Arthur Sorrentino, Higor Oliveira, Tulio Brunoro
% Este é um dos documentos de solução das questões.

FProb = @(n,p) ((1-p).^n)*p;
% Definição da função de probabilidade pra facilitar no futuro.
n = 0:1:10;
% Vetor de valores de amostras, será usado também futuramente nos plots

FMP = zeros(length(n), 3); % Matriz resultados das funções.
%Colunas são os resultados para facilitar o print futuramente

FMP(:,1) = FProb(n, 1/5);
% resultado da aplicação da função para os 3 diferentes casos
FMP(:,2) = FProb(n, 1/2);
FMP(:,3) = FProb(n, 4/5);
leg = {"P = 1/5", "P = 1/2", "P = 4/5"};

figure ('name','Funcao FMP para diferentes probabilidades.');
% Primeiros plots
plot(n, FMP(:,1), 'om--');
hold on;
plot(n, FMP(:,2), 'og--');
plot(n, FMP(:,3), 'or--');
title('Funcao FMP para diferentes probabilidades.');
legend(leg);

fprintf(" Plotando os graficos de FMP\n");
pause(0.1); % Pause para forçar render dos gráficos

for j = 1:length(FMP(1,:))
```

```
% Calculo das probabilidades acumuladas de cada função.
    for i = 1:length(FMP(:,j))
        FDA(i,j) = sum(FMP(1:i,j));
        % Sum equivale a integral para períodos discretos
    end
end

figure ('name', 'Funcao FDA para diferentes probabilidades.');
% Segundos plots
plot(n, FDA(:,1), 'om-');
hold on;
plot(n, FDA(:,2), 'og-');
plot(n, FDA(:,3), 'or-');
title('Funcao FDA para diferentes probabilidades.');
legend(leg, 'Location','southeast');

fprintf(" Plotando os graficos de FDA\n");
pause(0.1); % Pause para forçar render dos gráficos

figure ('name','Funcao FMP em escala semilog para diferentes probabilidades.');
% Terceiro, da escala em semilog
semilogy(n, FMP(:,1), 'om--');
hold on;
semilogy(n, FMP(:,2), 'og--');
semilogy(n, FMP(:,3), 'or--');
title('Funcao FDA para diferentes probabilidades em semilog de y.');
legend(leg, 'Location','southwest');

fprintf(" Plotando os graficos de FDA em escala de semilog.\n");
pause(0.1); % Pause para forçar render dos gráficos
```

Apêndice B - Código "Solve2_1_1"

```
% Trabalho 2 da matéria de Algoritmos Númericos para Engenharia feito
% em Octave.
% O trabalho é a reprodução de curvas estastíticas características,
% como FDA, FMP
% Usando Symbolic v2.9.0 e SymPy v1.5.1.
% Por Arthur Sorrentino, Higor Oliveira, Tulio Brunoro
% Este é um dos documentos de solução das questões.

printf(' Chamando funcao meu_geornd com P = 0.5 e n = 20.\n');
printf(' meu_geornd(0.5,20)\n');
k = meu_geornd(0.5,20);
printf(' Retorno: ');
k
```

Apêndice C - Código "meu_geornd"

```
% Função meugeornd do trabalho 2 da matéria de Algoritmos Númericos
% para Engenharia feito em Octave.
% Usando Symbolic v2.9.0 e SymPy v1.5.1.
% Por Arthur Sorrentino, Higor Oliveira, Tulio Brunoro

% Sobre a função:
% Retorna n amostras de tentativas de Bernoulli com quantidade de insucessos
% antes do primeiro sucesso, dada a probabilidade P de sucesso da
% tentativa de Bernoulli.

function k = meu_geornd(P, n)

if(P <= 0 || n <= 0)
    printf('\nValores de P e n tem de ser positivos, inteiros e não nulos.\n')
    k = NaN;
    return;
end

semente = 1234567890; % Seed escolhida pelo professor
rand('seed', semente);

m = 1000+n; % Número de amostras para a distribuição uniforme
amostras = rand(ceil(sqrt(m)));
% geração da matriz de valores aleatorios. O round é só para caso
% a pessoa escolha um valor com valor da raiz não inteira
% boll_vec = zeros(1,m); % Desnecessário a prealocação do vetor

bool_vec = amostras < P; % Filtro lógico do octave
k = zeros(1,n); % Pré alocagem de k

i = 1; j = 1; % i é o index que percorre o vetor de amostras e j é a
% posição no vetor de saída.
while (i < m && j < n)
% Não extrapolar o vector e para quando atingir o número de sucessos n
    if (bool_vec(i))
```

```
        k(j) = k(j) + 1;
    else
        j = j + 1;
    end
    i = i + 1;
end

% print da comparação entre a probabilidade do resultado em comparação
% com a probabilidade real do evento.
if(1)
    P_estimado = n./(n+sum(k));
    printf(" Diferenca entre o valor estimado e real da probabilidade:\n");
    printf(" Valor estimado: %.5f", P_estimado);
    printf(" Valor real: %.5f", P);
    printf(" Erro: %.3e\n", abs(P_estimado - P));
end

return;
```

Apêndice D - Código "Solve3_1"

```
% Trabalho 2 da matéria de Algoritmos Númericos para Engenharia
% feito em Octave.
% O trabalho é a reprodução de curvas estastíticas características,
% como FDA, FMP
% Usando Symbolic v2.9.0 e SymPy v1.5.1.
% Por Arthur Sorrentino, Higor Oliveira, Tulio Brunoro
% Este é um dos documentos de solução das questões.

FDPWeibull = @(x, k, lambda) (k/lambda).*((x/lambda).^(k-1)).*exp(-(x/lambda).^k);
% Definição das funções de FDP e FDA de Weibull
FDAWeibull = @(x, k, lambda) 1 - exp(-(x/lambda).^k);

lambda = [1.0 1.0 1.0 1.0 2.0 2.0];
k =      [0.5 1.0 2.0 5.0 1.0 5.0];
x = 0:0.01:4;

for i = 1:length(lambda)
    leg3{i} = sprintf([sprintf('lambda = %.1f', lambda(i))
    sprintf(', k = %.1f', k(i))]);
end

FDPW = zeros(length(x), length(lambda));
% Pré-alocagem da matriz com os valores de FMP da distribuição de Weibull

for i = 1:length(lambda)
    FDPW(:, i) = FDPWeibull(x, k(i), lambda(i));
end

figure ('name', 'Distribuicao de Weibull: FDP.')
% Plots da FDP de Weibull
plot(x(2:length(x)), FDPW(2:length(FDPW(:,1)),1), '-b-');
% Pulando o primeiro termo, que deve ser infinito
hold on;
plot(x, FDPW(:,2), '-r-');
plot(x, FDPW(:,3), '-m-');
```

```
plot(x, FDPW(:,4), '-g-');
plot(x, FDPW(:,5), '-c-');
plot(x, FDPW(:,6), '-y-');
hold off;
title('Distribuicao de Weibull: FDP.');
legend(leg3);
ylim([0, 2]);

fprintf(" Plotando os graficos de FDP da distribuicao de Weibull.\n");
pause(0.1); % Pause para forçar render dos gráficos

for i = 1:length(lambda)
    FDAW(:, i) = FDAWeibull(x, k(i), lambda(i));
end

figure ('name', 'Distribuicao de Weibull: FDA.') % Plots da FDA de Weibull
plot(x, FDAW(:,1), '-b-');
hold on;
plot(x, FDAW(:,2), '-r-');
plot(x, FDAW(:,3), '-m-');
plot(x, FDAW(:,4), '-g-');
plot(x, FDAW(:,5), '-c-');
plot(x, FDAW(:,6), '-y-');
hold off;
title('Distribuicao de Weibull: FDA.');
legend(leg3,'Location','southeast');

fprintf(" Plotando os graficos de FDA da distribuicao de Weibull.\n");
pause(0.1); % Pause para forçar render dos gráficos
```

Apêndice E - Código "Solve3_1_1e3_1_2"

```
% Trabalho 2 da matéria de Algoritmos Númericos para Engenharia
% feito em Octave.
% O trabalho é a reprodução de curvas estastíticas características,
% como FDA, FMP
% Usando Symbolic v2.9.0 e SymPy v1.5.1.
% Por Arthur Sorrentino, Higor Oliveira, Tulio Brunoro
% Este é um dos documentos de solução das questões.

FDPWeibull = @(x, k, lambda) (k/lambda).*((x/lambda).^(k-1)).*exp(-(x/lambda).^k);
FDAWeibull = @(x, k, lambda) 1 - exp(-(x/lambda).^k);

a = 0; b = 1;
lambda2 = 1.0;
k2 =      5.0;
h = 0.01;
x = a:h:2.5;
n = length(x);
x_igual_1 = a + 1/h + 1;

wrapper_FDPWeibull = @(x) FDPWeibull(x, k2, lambda2);

% Valores reais da função
FDAW2 = FDAWeibull(x, k2, lambda2);

% Definição dos vetores resultantes das integrações
TR_1 = zeros(1,n);
TR_4 = zeros(1,n);
TR_6 = zeros(1,n);
TR_10 = zeros(1,n);
SR_2 = zeros(1,n);
SR_4 = zeros(1,n);
SR_6 = zeros(1,n);
SR_10 = zeros(1,n);

coluna = {"I-TR-1", "I-TR-4", "I-TR-6", "I-TR-10", "I-SR-2",
```

```
"I-SR-4", "I-SR-6", "I-SR-10"};
leg1 = {"Lambda = 1.0, k = 5.0", coluna{1:4}};
leg2 = {"Lambda = 1.0, k = 5.0", coluna{5:8}};

% Calculo dos valores aproximados
for i = 1:n
% Método 1: integração por Trapézios usando a função pronta do professor
TR_1(i) = integralTrapeziosRepetidaFunc(wrapper_FDPWeibull, a, x(i), 1, false);
TR_4(i) = integralTrapeziosRepetidaFunc(wrapper_FDPWeibull, a, x(i), 4, false);
TR_6(i) = integralTrapeziosRepetidaFunc(wrapper_FDPWeibull, a, x(i), 6, false);
TR_10(i) = integralTrapeziosRepetidaFunc(wrapper_FDPWeibull, a, x(i), 10, false);

% Método 2: integração por Simpson 1/3 usando a função pronta do professor
SR_2(i) = integralSimpsonRepetidaFunc(wrapper_FDPWeibull, a, x(i), 2, false);
SR_4(i) = integralSimpsonRepetidaFunc(wrapper_FDPWeibull, a, x(i), 4, false);
SR_6(i) = integralSimpsonRepetidaFunc(wrapper_FDPWeibull, a, x(i), 6, false);
SR_10(i) = integralSimpsonRepetidaFunc(wrapper_FDPWeibull, a, x(i), 10, false);
end
resultados = [TR_1; TR_4; TR_6; TR_10; SR_2; SR_4; SR_6; SR_10;];

fprintf(" Tabela com resultados das aproximacoes de FDA com
metodos de integracao\n");
% Definição da divisão
sep = repmat(['-'], 1, 50);
% Print da divisão
fprintf("%.40s\n", sep);
% Print da primeira linha
p1 = sprintf("%.28s", sprintf("%-28s", "Valor exato em x = 1"));
p2 = sprintf("%.14s", sprintf("%-12s", sprintf("%.3e",FDAW2(x_igual_1))));
linha = [p1 p2 '\n'];
fprintf(linha);
% Print da segunda linha
p1 = sprintf("%.14s", sprintf("%-14s", "Metodo"));
p2 = sprintf("%.14s", sprintf("%-14s", "Valor"));
p3 = sprintf("%.14s", sprintf("%-14s", "Erro"));
linha = [p1 p2 p3 '\n'];
fprintf(linha);
% Print da divisão
fprintf("%.40s\n", sep);
```

```
% Print das linhas de resultado
for i = 1:8
nome = sprintf(coluna{i});
p1 = sprintf("%.14s", sprintf("%-14s", nome));
p2 = sprintf("%.14s", sprintf("%-14s",
sprintf("%.3e",-resultados(i, x_igual_1))));
p3 = sprintf("%.14s", sprintf("%-14s",
sprintf("%.3e",-resultados(i, x_igual_1)+FDAW2(x_igual_1))));
linha = [p1 p2 p3 '\n'];
fprintf(linha)
end
% Print da divisão
fprintf("%.40s\n\n", sep);

% Plots dos gráficos
figure ('name', 'FDA e FDA por integracao numerica: Trapezios')
% Plots da FDA de Weibull
plot(x, FDAW2, '-g--');
hold on;
plot(x, TR_1, '-r-');
plot(x, TR_4, '-c-');
plot(x, TR_6, '-m-');
plot(x, TR_10, '-g-');
hold off;
title("Distribuicao de Weibull: FDA e FDA por integracao
numerica: Trapezios");
legend(leg1,'Location','northwest');

fprintf(" Plotando os graficos de FDA de Weibull e
aproximacoes de trapezios.\n");
pause(0.1); % Pause para forçar render dos gráficos

figure ('name', 'FDA e FDA por integracao numerica: Simpson 1/3')
% Plots da FDA de Weibull
plot(x, FDAW2, '-g--');
hold on;
plot(x, SR_2, '-r-');
plot(x, SR_4, '-c-');
plot(x, SR_6, '-m-');
```

```
plot(x, SR_10, '-g-');
title("Distribuicao de Weibull: FDA e FDA por integracao
numerica: Simpson 1/3");
legend(leg2,'Location','northwest');
hold off;

fprintf(" Plotando os graficos de FDA de Weibull e aproximacoes
de Simpson.\n");
pause(0.1); % Pause para forçar render dos gráficos
```

Apêndice F - Código "Solve3_2_1e3_2_2"

```
% Trabalho 2 da matéria de Algoritmos Númericos para Engenharia feito
% em Octave.
% O trabalho é a reprodução de curvas estatísticas características,
% como FDA, FMP
% Usando Symbolic v2.9.0 e SymPy v1.5.1.
% Por Arthur Sorrentino, Higor Oliveira, Tulio Brunoro
% Este é um dos documentos de solução das questões.

% A definição equação de phi será dividido em várias partes
% para calcular cada parte individualmente
phi = @(x, k) ( mean(x.^k.*log(x))/mean(x.^k) - mean(log(x))).^-1;
lambda = @(x, k) nthroot(mean(x.^k),k);

n = 1000; % numero de termos da primeira distribuição
vec_k(1) = 5; % k dado
vec_lambda(1) = 1; % lambda dado

fprintf(" Primeiro valor de lambda = %.2f, e de
k = %.2f\n", vec_lambda(1), vec_k(1));

X = wblrnd_octave(vec_lambda(1), vec_k(1), ceil(sqrt(n)));
% Gerado as amostras. n^1/2 pois o resultado é matrix nxn
x = X(1:1000);
% Converte a matriz na brutalidade para vector.
% A função precisa receber um vetor, n matrix

vec_k(2) = phi(x, vec_k(1)); % nova estimativa
vec_lambda(2) = lambda(x, vec_k(1));

fprintf(" Estimativa do valor de lambda = %.2f, e de
k = %.2f na 2 iteracao\n", vec_lambda(2), vec_k(2));
```

Apêndice G - Código "main"

```
% Trabalho 2 da matéria de Algoritmos Númericos para Engenharia
% feito em Octave.
% O trabalho é a reprodução de curvas estastíticas características,
% como FDA, FMP
% Usando Symbolic v2.9.0 e SymPy v1.5.1.
% Por Arthur Sorrentino, Higor Oliveira, Tulio Brunoro

close all; % Fecha janelas de plot abertas
clear all; % Limpeza do ambiente
%close all hidden % Limpeza mais detalhada

addpath(['../' '../' 'util'], ['../' '../' 'edo'], ['../' '../' 'integral']);
% Inserção das funções auxiliares
available_graphics_toolkits();
% Adiciona um pacote de alteração do gráfico

graphics_toolkit gnuplot;

% ------------------------------ 2.1 ------------------------------
fprintf("\n--------------- Solucao - 2.1 ---------------\n");
Solve2_1;

fprintf("\nAperte enter para continuar");
pause();
% ------------------------------ 2.1.1 ------------------------------
fprintf("\n--------------- Solucao - 2.1.1 -------------\n");
Solve2_1_1

fprintf("\nAperte enter para continuar");
pause();
% ------------------------------ 3.1 ------------------------------
fprintf("\n--------------- Solucao - 3.1 ---------------\n");
Solve3_1

fprintf("\nAperte enter para continuar");
```

```
pause();
% ------------------------------ 3.1.1 e 3.1.2 ------------------------------
fprintf("\n--------------- Solucao - 3.1.1 e 3.1.2 ---------------\n");
Solve3_1_1e3_1_2

fprintf("\nAperte enter para continuar");
pause();
% ------------------------------ 3.2.1 e 3.2.2 ------------------------------
fprintf("\n--------------- Solucao - 3.2.1 e 3.2.2 ---------------\n");
Solve3_2_1e3_2_2

fprintf("\n\n Finalizando... \n");
```