UNIVERSIDADE FEDERAL DE PELOTAS
Centro de Desenvolvimento Tecnológico
Programa de Pós-Graduação em Computação

Dissertação

Extensão intervalar para as variáveis aleatórias com distribuições Uniforme, Normal, Gama, Exponencial e Pareto

Alice Fonseca Finger

Pelotas, 2014

Alice Fonseca Finger

Extensão intervalar para as variáveis aleatórias com distribuições Uniforme, Normal, Gama, Exponencial e Pareto

Dissertação apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal de Pelotas, como requisito parcial à obtenção do título de Mestre em Ciência da Computação

Orientadora: Profa. Dra. Aline Brum Loreto

Pelotas, 2014

Dados de catalogação na fonte:
Ubirajara Buddin Cruz – CRB-10/901
Biblioteca de Ciência & Tecnologia – UFPel

F497e Finger, Alice Fonseca

Extensão intervalar para as variáveis aleatórias com distribuições Uniforme, Normal, Gama, Exponencial e Pareto / Alice Fonseca Finger. – Pelotas, 2014. – 68 f: gráf. – Dissertação (Mestrado em Ciência da Computação) – Programa de Pós-Graduação em Computação. Universidade Federal de Pelotas. Pelotas, 2014. – Orientador: Aline Brum Loreto.

1. Aritmética intervalar. 2. Estatística intervalar. 3. Complexidade computacional. 4. Qualidade do intervalo. I. Loreto, Aline Brum. II. Título.

CDD: 511.352

AGRADECIMENTOS

Agradeço primeiramente a Deus, por ter colocado pessoas maravilhosas na minha vida e por me proporcionar fazer escolhas certas e chegar até onde cheguei. Agradeço aos meus pais, com certeza eles são grandes responsáveis por mais essa etapa concluída na minha vida. Agradeço aos exemplos de pai e de mãe maravilhosos e exemplos de profissionais, nos quais sempre me espelho e levo como ensinamento pro resto da vida. Obrigada por me proporcionarem a oportunidade de além de concluir uma graduação, ainda concluir o mestrado com todas as ”coisas boas” de viver na casa dos pais. Obrigada minha irmã Paula, por nossa amizade, nosso companheirismo e amor incondicional que sentimos. Tu és um exemplo pra mim, pela tua força, pelo teu trabalho e pela pessoa que és. Tenho certeza que todo teu estudo e trabalho são exemplos para mim e renderão muitos frutos. Agradeço a Deus pela família maravilhosa que tenho, que me apoia e que tem grande contribuição no sucesso que foram esses 2 anos de mestrado. Obrigada! Ao meu namorado Julio, obrigada por fazer a diferença nesses 5 anos que estás comigo. Pessoa especial, que ao longo desses 2 anos de mestrado foi fundamental com o apoio, amor, companheirismo e força que me deu. Espero retribuir pelo menos a metade de todo apoio e exemplo que fostes pra mim nesses anos de muita dedicação. Com certeza temos muitas comemorações ainda pela frente, e essa é apenas uma etapa que vencemos juntos. Te amo! Já não bastasse a orientação maravilhosa na graduação, tive a sorte de ter a melhor orientadora no mestrado. Nunca pensei e nunca ouvi falar que um mestrado poderia ser tão bom. Foi maravilhoso conviver mais esses 2 anos contigo. Cada conselho, cada conversa, cada risada, cada chimarrão, cada ”puxão de orelha”, eu só tenho a agradecer. Se já éramos amigas durante a orientação na graduação, agora tenho a certeza que tenho uma amiga maravilhosa pro resto da vida. Aline, minha admiração por ti é imensa. Obrigada por todo incentivo e por acreditar em mim. Me tornei uma pessoa capaz de concluir o mestrado, capaz de passar em um concurso público e capaz de fazer pro resto da vida o que eu sempre quis, ser professora. Fostes fundamental em todas essas etapas da minha vida. Sentirei tua falta, mas tenho certeza que nos encontraremos muito ainda e teremos muitos trabalhos juntas. Agradeço a minha instituição, UFPel, pelas oportunidades que me proporcionou. Além dos anos de mestrado, ainda pude ministrar aulas durante 1 ano e meio, o que me deixou mais certa da carreira que seguirei. Agradeço ao pessoal do mestrado, meus colegas que no primeiro semestre foram fundamentais durante nossas várias horas de estudos, risadas e trabalhos. Obrigada Rodolfo, Vakaria, Deives e Leonardo. A nossa querida secretaria Martha, que sempre esteve prestativa e ajudando em todos os momentos que precisei, muito obrigada. Aos meus colegas de grupo TEIa, Lucas, Maurício, Mariline e Vinícius, agradeço a convivência maravilhosa e aos frutos que renderam nossos trabalhos. Ao Maurício, um agradecimento especial. Fostes muito importante nessa reta final de dissertação. Obrigada pela paciência, por seres prestativo e por toda ajuda que sempre me ofereceu. Muito obrigada!

”A persistência é o menor caminho do êxito.”

— CHARLES CHAPLIN

RESUMO

FINGER, Alice Fonseca. **Extensão intervalar para as variáveis aleatórias com distribuições Uniforme, Normal, Gama, Exponencial e Pareto**. 2014. 68 f. Dissertação (Mestrado em Ciência da Computação) – Programa de Pós-Graduação em Computação, Universidade Federal de Pelotas, Pelotas, 2014.

Quando se trabalha com números de ponto flutuante o resultado é apenas uma aproximação de um valor real e erros gerados por arredondamentos ou por instabilidade dos algoritmos podem levar a resultados incorretos. Não se pode afirmar a exatidão da resposta estimada sem o auxílio de uma análise de erro. Utilizando-se intervalos para representação dos números reais, é possível controlar a propagação desses erros, pois resultados intervalares carregam consigo a segurança de sua qualidade. Para obter o valor numérico das funções densidade de probabilidade das variáveis aleatórias contínuas com distribuições Uniforme, Exponencial, Normal, Gama e Pareto se faz necessário o uso de integração numérica, uma vez que a primitiva da função nem sempre é simples de se obter. Além disso, o resultado é obtido por aproximação e, portanto, afetado por erros de arredondamento ou truncamento. Neste contexto, o presente trabalho possui três principais objetivos: i) definir funções densidade de probabilidade das variáveis com distribuições Gama e Pareto na forma intervalar, completando as definições já existentes na literatura para as distribuições Uniforme, Exponencial e Normal; ii) analisar a qualidade dos intervalos encapsuladores para as variáveis aleatórias com distribuições Uniforme, Exponencial, Normal, Gama e Pareto, através de resultados gerados a partir de algoritmos implementados na linguagem Python, utilizando o pacote intervalar IntPy; iii) analisar a complexidade computacional para computar as funções densidade de probabilidade com distribuições Uniforme, Exponencial, Normal, Gama e Pareto nas formas real e intervalar. Através desses objetivos, certifica-se que, ao utilizar aritmética intervalar para o cálculo da função densidade de probabilidade das variáveis aleatórias com distribuições, é possível ter um controle automático de erros com limites confiáveis, e, no mínimo, manter o esforço computacional existente nos cálculos que utilizam a aritmética real.

Palavras-chave: Aritmética intervalar, estatística intervalar, complexidade computacional, qualidade do intervalo.

ABSTRACT

FINGER, Alice Fonseca. **Interval extension for random variables with Uniform, Normal, Gamma, Exponential and Pareto distributions**. 2014. 68 f. Dissertação (Mestrado em Ciência da Computação) – Programa de Pós-Graduação em Computação, Universidade Federal de Pelotas, Pelotas, 2014.

When working with floating point numbers the result is only an approximation of a real value and errors generated by rounding or by instability of the algorithms can lead to incorrect results. We can't affirm the accuracy of the estimated answer without the contribution of an error analysis. Using intervals for the representation of real numbers, it is possible to control this error propagation, because intervals results carry with them the security of their quality. To obtain the numerical value of the probability density functions of continuous random variables with distributions Uniform, Exponential, Normal, Gamma and Pareto is necessary to use numerical integration, once the primitive of the integral do not always is simple to obtain. Moreover, the result is obtained by approximation and therefore affected by truncation or rounding errors. In this context, this paper has three main objectives: i) define probability density function of variables with Gamma and Pareto distributions in the interval form completing the already existing definitions in the literature for the Uniform, Exponential and Normal distributions; ii) analyze the quality of wrappers intervals for random variables with distributions Uniform, Exponential, Normal, Gamma and Pareto, through results generated from algorithms implemented in the Python language, using the interval package IntPy; iii) analyze the computational complexity of computing the probability density functions with Uniform, Exponential, Normal, Gamma and Pareto distributions in the real and interval forms. Through these goals, make sure that by using interval arithmetic to calculate the probability density function of the random variables with distributions, it is possible to have an automatic error control with reliables boundaries, and, at least, keep the existing computational effort in the calculation using the real arithmetic.

Keywords: interval arithmetic, statistical interval, computational complexity, quality of the interval.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Na matemática intervalar, o valor real x é aproximado por um intervalo $\mathbf{x}$, que possui $\underline{x}$ e $\bar{x}$ como limites inferior e superior, de forma que o intervalo contenha x. O tamanho deste intervalo pode ser usado como medida para avaliar a qualidade de aproximação (RATSCHEK; ROKNE, 1988). Os cálculos reais são substituídos por cálculos que utilizam a aritmética intervalar (MOORE, 1966).

A análise intervalar surgiu com o objetivo inicial de controlar a propagação de erros numéricos em procedimentos computacionais. Mas, aparentemente, a matemática intervalar duplica o problema de representação dos números reais em processadores numéricos, uma vez que ao invés de operar com um número real, operam-se com dois. Entretanto, sua realização é feita por meio de números de ponto flutuante, isto é, os extremos do intervalo $\mathbf{x}$ são números de máquina $\underline{x}_{pf}$ e $\bar{x}_{pf}$ (MOORE, 1966)(MOORE, 1979)(MOORE; KEARFOTT; CLOUD, 2009).

Intervalos automatizam a análise do erro computacional. Através da utilização de intervalos, tem-se um controle automático de erros com limites confiáveis.

A aritmética intervalar utiliza intervalos reais para representar valores infinitos, valores desconhecidos ou para representar valores contínuos que podem ser conhecidos ou não. Os intervalos servem para representar dados inexatos, aproximações e erros de truncamento de procedimentos (LORETO, 2006).

É importante ressaltar que existem questões do tipo: “Por que utilizar técnicas intervalares se existem na maioria dos sistemas de computadores bibliotecas matemáticas avançadas e eficientes, que resolvem a maioria dos problemas?”.

A justificativa do uso de técnicas intervalares, segundo Rastchek e Rokne (RATSCHEK; ROKNE, 1988), inicia pelo fato de que os computadores empregam aritmética chamada de ponto flutuante ou ponto fixo. Portanto, são gerados erros quando um valor real de entrada é aproximado por um número de máquina, resultados intermediários são gerados na execução de cada operação e vão se acumulando, ou ainda, um outro tipo de erro relacionado com a incerteza dos dados de entrada, o que acontece muito em casos de experimentos físicos e químicos onde os dados de entrada são incertos.

No processo de resolução de problemas podem ser constatadas fontes de erros, tais como: propagação dos erros nos dados iniciais, arredondamento e erros de truncamento, causados ao se truncar sequências de operações aritméticas, após um número finito de etapas. Neste contexto, percebe-se a importância de técnicas intervalares. Ressalta-se que uma resposta intervalar carrega com ela a garantia de sua incerteza. Um valor pontual não carrega medidas de sua incerteza. Mesmo quando uma análise de sondagem do erro é executada, o número resultante é somente uma estimativa do erro que pode estar presente.

Segundo Kearfott (KEARFOTT, 1996), são muitas as aplicações de intervalos e nas mais diversas áreas, tais como: programação matemática, manipulação de equações, análise e projeto de circuitos elétricos, psicologia matemática, estatística, equações diferenciais, física e muitos outros.

No estudo das variáveis aleatórias contínuas sobre o conjunto dos números reais, $\mathbb{R}$, um dos problemas é o cálculo de probabilidades, visto que é necessário resolver uma integral definida da função densidade que, na maioria das vezes, não possui primitiva explícita ou cuja primitiva não é simples de se obter. Considerando que integrais de funções densidade de probabilidade sejam resolvidas analiticamente, seu valor numérico é dado por aproximação e, portanto, afetado por erros de arredondamento ou truncamento.

Computar probabilidades em situações práticas envolve números e, consequentemente, problemas numéricos. Problemas numéricos na computação científica originam-se primordialmente da impossibilidade de se operar com números reais diretamente, pois tem-se que representar uma grandeza contínua (a reta real) de forma discreta (palavras de máquina) (CAMPOS, 1997). O sistema de ponto flutuante é uma aproximação prática dos números reais.

O método para implementação de operações e algoritmos intervalares em máquinas é realizado por meio do critério de semimorfismo proposto por Kulisch (KULISCH, 2008) (KULISCH; MIRANKER, 1981). Considerando que o controle do erro numérico é feito através do uso de intervalos ao invés de números reais, Kulisch (KULISCH, 2008) e Kulisch e Miranker (KULISCH; MIRANKER, 1981) propuseram que a implementação da aritmética intervalar seja realizada através da chamada aritmética de exatidão máxima, o que significa a busca para que resultados numéricos ou sejam um número de ponto flutuante ou estejam entre dois números de ponto flutuantes consecutivos.

Considerando que métodos númericos devem ser usados para o cálculo de integrais a idéia é que estes sejam suportados pela matemática intervalar e a aritmética de exatidão máxima, o que implica que cálculos numéricos em computadores sejam realizados por meio das linguagens ou bibliotecas que tenham definidos o tipo intervalo e as operações sobre o tipo, usualmente denominadas de linguagens XSC (eXtended Scientific Computation).

O objetivo deste trabalho é definir a função densidade de probabilidade das variáveis aleatórias contínuas com distribuições Gama e Pareto com entradas intervalares. Para isso, utiliza-se o método de extensão intervalar, baseado na matemática intervalar (MOORE, 1979) e na aritmética de exatidão máxima (KULISCH, 2008)(KULISCH; MIRANKER, 1981). Após definidas, realiza-se a análise da qualidade dos intervalos encapsuladores para as funções densidade de probabilidade das variáveis aleatórias com distribuições Uniforme, Exponencial, Normal, Gama e Pareto. A análise é feita a partir de resultados obtidos com as implementações utilizando o pacote de programação intervalar IntPy (INTPY, 2014).

Dentre a diversas soluções ou implementações para determinado problema, muitas vezes é preciso escolher a mais eficiente. Para isso, existe a análise de algoritmos, a qual tem como objetivo melhorar, se possível, seu desempenho e escolher, entre os algoritmos disponíveis, o melhor. Existem vários critérios de avaliação de um algoritmo como: quantidade de trabalho requerido, quantidade de espaço requerido, simplicidade, exatidão de resposta e otimalidade (TOSCANI; VELOSO, 2001).

Preocupados se a quantidade de trabalho dispendido pelo algoritmo aumenta ao utilizar intervalos para controlar a propagação de erros, o presente trabalho realiza a análise da complexidade dos algoritmos elaborados para computar as funções densidade de probabilidade das variáveis aleatórias com distribuições Uniforme, Exponencial, Normal, Pareto e Gama, nas formas real e intervalar. Através desta análise, justifica-se que ao utilizar intervalos para calcular tais funções é possível ter um controle automático de erros com limites confiáveis e manter o esforço computacional (ou quantidade de trabalho requerido) quando calculado com as funções na sua forma real.

1.1 Objetivos

O objetivo principal desta dissertação é comprovar que utilizando-se a matemática intervalar para calcular a densidade de probabilidade das variáveis aleatórias contínuas com distribuições Uniforme, Exponencial, Normal, Pareto e Gama é possível obter resultados com um controle automático de erros e manter o mesmo esforço computacional quando se utiliza a aritmética com entradas reais.

A fim de alcançar o objetivo principal, abaixo são listados e detalhados objetivos específicos que devem ser executados.

- Definir as funções densidade de probabilidade das distribuições Gama e Pareto com entradas intervalares: Na literatura encontram-se as definições das funções densidade de probabilidade das variáveis aleatórias contínuas com distribuições Uniforme, Exponencial e Normal. Assim, a fim de completar a definição intervalar para as funções densidade de probabilidade das variáveis aleatórias, realiza-se a definição intervalar para as distribuições Gama e Pareto, nas quais é aplicado um dos métodos da aritmética intervalar para definir, através de uma função real, a sua correspondente função intervalar, chamado método de extensão intervalar.

- Analisar a qualidade dos intervalos encapsuladores: No presente trabalho são apresentados os resultados dos intervalos encapsuladores para as funções densidade de probabilidade com distribuições Uniforme, Exponencial e Pareto. Tais resultados estão presentes no trabalho desenvolvido por Varjão (VARJÃO, 2011), onde é possível encontrar uma comparação dos resultados obtidos com implementação de cada função intervalar, utilizando o método de Simpson Intervalar definido por Caprani *et al* (CAPRANI; MADSEN; NIELSEN, 2002), com o pacote de programação intervalar IntPy (INTPY, 2014) e os resultados obtidos por Santos (SANTOS, 2010) através do uso do pacote IntLab (MATLAB, 2014). Serão apresentadas, quando possível, duas formas de obter o intervalo, uma utilizando o método de Simpson Intervalar para integração numérica e outra a partir da primitiva da função. Apresentam-se os resultados de qualidade para cada função, utilizando as medidas de erros absoluto e erro relativo, bem como o cálculo do diâmetro do intervalo. Salienta-se que no trabalho desenvolvido por Varjão (VARJÃO, 2011) não se encontra a análise da qualidade dos intervalos encapsuladores. Assim, também é realizada a análise da qualidade dos intervalos obtidos por Varjão, a fim de comparar com os desenvolvidos na presente dissertação.

- Analisar a complexidade computacional: Para a distribuição Uniforme, utiliza-se a primitiva da função para gerar solução com entradas reais e intervalares. Já para as distribuições Exponencial e Pareto, implementam-se duas soluções com entradas reais e duas com entradas intervalares. Uma das soluções é através da primitiva da função, e a outra através da solução da integral utilizando o método de Simpson. Para as distribuições Normal e Gama, a solução gerada com entradas real e intervalar utiliza o método de Simpson. Para cada solução apresentada para gerar o intervalo encapsulador das funções densidade de probabilidade são elaborados algoritmos para serem implementados na linguagem Python (PYTHON, 2014), utilizando o pacote intervalar IntPy (INTPY, 2014), o qual possibilita a programação com entradas intervalares. Destes algoritmos, realiza-se uma análise da complexidade computacional de cada solução para cada distribuição. Ressalta-se que na literatura não se encontrou análise da complexidade das funções com entradas reais, justificando a realização da mesma nesse trabalho. Com a análise de complexidade realizada para cada solução das funções densidade de probabilidade com distribuições, confirma-se que, ao utilizar intervalos para representar essas funções, é possível controlar a propagação de erros e, no mínimo, manter o esforço computacional da forma real.

1.2 Organização do Trabalho

A dissertação organiza-se da seguinte maneira:

O Capítulo 2 apresenta uma fundamentação teórica sobre aritmética intervalar, citando seus principais conceitos e métodos para computar um intervalo solução, e sobre a teoria da probabilidade. Após, descrevem-se os trabalhos já desenvolvidos com relação a estatística e probabilidade intervalar, apresentando os principais resultados nas áreas da estatística computacional e complexidade computacional de problemas com entradas intervalares.

No Capítulo 3 descrevem-se as funções densidade de probabilidade das variáveis aleatórias contínuas com distribuições nas duas formas, com entradas reais e intervalares, apresentando suas respectivas fórmulas.

No Capítulo 4 realiza-se a análise da qualidade dos intervalos com base nos resultados numéricos oriundos de aplicações em exemplos utilizando entradas reais e intervalares.

O Capítulo 5 é uma análise da complexidade computacional das funções densidade de probabilidade com distribuições na sua forma real e intervalar.

No Capítulo 6 apresentam-se as conclusões obtidas na dissertação, bem como propostas de trabalhos futuros.

Por fim são apresentadas as referências bibliográficas.

2 FUNDAMENTAÇÃO TEÓRICA

O presente capítulo apresenta conceitos da matemática intervalar, juntamente com suas definições, que serão utilizados no desenvolvimento dessa dissertação. Descreve-se e exemplifica-se o método de extensão intervalar, o qual é utilizado para definir de forma intervalar as funções densidade de probabilidade das variáveis aleatórias com distribuições Gama e Pareto. Além disso, apresenta-se uma seção com a teoria da probabilidade com o objetivo de, na seção seguinte, apresentar a teoria da probabilidade intervalar definida por Campos (CAMPOS, 1997). Nesta seção, ainda são destacados os principais trabalhos relacionados com a dissertação, mostrando o que já existe na área de estatística intervalar e probabilidade intervalar, com foco nas variáveis aleatórias contínuas. Por fim, encerra-se o capítulo com a definição intervalar do Método de Simpson, a qual foi desenvolvida por Caprani *et al* (CAPRANI; MADSEN; NIELSEN, 2002) e utilizada em soluções de algumas das funções apresentadas no presente trabalho.

2.1 Matemática Intervalar

A aritmética intervalar (MOORE, 1966)(MOORE, 1979)(MOORE; KEARFOTT; CLOUD, 2009) é baseada no uso de intervalos fechados $[x_1, x_2]$ de números reais como elementos básicos e sua idéia do ponto de vista computacional, é: dada uma função $f(x)$ de variável real x pertencente a um intervalo $\mathbf{x}$=$[x_1, x_2]$ onde $x_1, x_2 \in \mathbb{R}$, a imagem da f é dada por

$$f(\mathbf{x}) = \{y \mid y = f(x), x_1 \leq \mathbf{x} \leq x_2\},$$

onde esta em geral não é representada exatamente, mas é sempre possível determinar um intervalo $\mathbf{y} = [y_1, y_2]$ tal que $f(\mathbf{x}) \subseteq \mathbf{y}$, isto é $y_1 \leq f(x) \leq y_2$. Pode-se então definir uma função intervalar F associada a f pela transformação do intervalo $[x_1, x_2]$ em $[y_1, y_2]$, isto é:

$$f(x) \subseteq F(\mathbf{x}) = \mathbf{y}.$$

Esta função F, chamada extensão intervalar de f, deve ser aquela que possui o mínimo possível de diferença da imagem $f(\mathbf{x})$. O erro obtido no cálculo de $f(x)$ a partir do intervalo $\mathbf{x}$ é calculado através do diâmetro $w(F(\mathbf{x})) = y_2 - y_1$.

O cálculo de expressões na aritmética intervalar consiste em usar extensão das operações aritméticas junto com um conjunto de funções *standards*.

A aritmética intervalar utiliza um arredondamento especial, chamado arredondamento direcionado, o que significa que os resultados são arredondados para o menor e para o maior número de máquina que contém o resultado das operações, obtendo-se com isso um intervalo de máquina, com diâmetro mínimo, no qual a solução se situa.

Uma distinção importante na aritmética intervalar é entre imagem intervalar de uma função e avaliação intervalar da função.

Definição 1. *A imagem intervalar de uma função f, contínua no intervalo $\mathbf{x}$, é definida como o intervalo limitado pelo mínimo da imagem de $f(x)$ e pelo máximo da imagem de $f(x)$, sendo x um elemento do intervalo $\mathbf{x}$.*

$$\mathbf{Im} = I(f,\mathbf{x}) = [min\{f(x)|x \in \mathbf{x}\}, max\{f(x)|x \in \mathbf{x}\}].$$

Segundo Ferson *et al* (FERSON; GINZBURG; KREINOVICH, 2002), historicamente, o primeiro método para computar o intervalo solução é a extensão intervalar (MOORE, 1979), ou avaliação intervalar (OLIVEIRA; DIVERIO; CLAUDIO, 1997). Este método está baseado no fato que em um computador, todo algoritmo consiste de operações elementares (aritméticas e lógicas). Para cada operação elementar $f(a,b)$, se são conhecidos os intervalos $\mathbf{a}$ e $\mathbf{b}$ para a e b, pode-se computar a imagem exata $f(\mathbf{a},\mathbf{b})$ através da aritmética intervalar definida por Moore (MOORE, 1966). Na extensão intervalar, repete-se a computação formando o programa f passo-a-passo, substituindo cada operação elementar de números reais pela correspondente operação da aritmética intervalar.

A extensão intervalar é definida da seguinte forma:

Definição 2. *A função $F : \mathbb{I}(\mathbb{R}) \to \mathbb{I}(\mathbb{R})$ é uma extensão intervalar de uma função $f : \mathbb{R} \to \mathbb{R}$, se para todo $x \in \mathbb{R}$, $F([x,x])$ = $[f(x), f(x)]$ (SANTIAGO; BEDREGAL; ACIÓLY, 2006).*

Exemplo 1. *Seja $f(x) = 3x + x$ e um intervalo degenerado $\boldsymbol{x} = [2,2] \subseteq Dom(f) = \mathbb{R}$. Substituindo x real por $\boldsymbol{x}$ intervalo, tem-se: $F(\boldsymbol{x}) = [3,3] \times [2,2] + [2,2] = [6,6] + [2,2] = [8,8]$. Sabendo-se que $f(2) = 6 + 2 = 8$, então tem-se que $F([2,2]) = [f(2), f(2)]$. Logo, F é uma extensão intervalar da função real f (DUARTE, 2007).*

Na computação intervalar, pode-se calcular o intervalo solução Y = $[\underline{y}, \bar{y}] = f(\mathbf{x}_1, \cdots, \mathbf{x}_n)$ através de métodos de aproximação, técnicas de otimização, extensão intervalar e, ainda, por métodos considerados mais sofisticados (KREINOVICH et al., 1998).

2.2 Teoria da Probabilidade

A Teoria da Probabilidade é o ramo da matemática relacionado com fenômenos aleatórios, os quais apresentam as seguintes características: (i) podem ser repetidos sob as mesmas condições; (ii) apresentam um conjunto de possíveis resultados; (iii) quando realizado um grande número de vezes, apresentam uma regularidade. A teoria da probabilidade desenvolve uma estrutura matemática formal para o tratamento dos fenômenos aleatórios (CAMPOS, 1997).

As variáveis aleatórias contínuas podem ser de grande utilidade na abordagem de problemas práticos. Os modelos probabilísticos para variáveis aleatórias contínuas também são denominados funções densidades de probabilidades, e podem envolver mais de um parâmetro.

Para calcular probabilidades, atribuímos um peso de probabilidade $P(x)$ a cada elemento do espaço amostral, de modo que o peso represente o que acreditamos ser a probabilidade relativa desse resultado. Existem duas regras na atribuição de pesos: i) os pesos devem ser números não negativos; ii) a soma dos pesos de todos os elementos em um espaço amostral deve ser 1. Definimos a probabilidade $P(E)$ do evento *E* como a soma dos pesos dos elementos de *E*. Algebricamente, escrevemos (STEIN; DRYSDALE; BOGART, 2013):

$$P(E) = \sum_{x:x \in E} P(x). \tag{1}$$

Uma variável aleatória para um experimento com um espaço amostral *S* é uma função que atribui um número a cada elemento de *S*. Normalmente, em vez de usar *f* para representar tal função, usamos *X* (STEIN; DRYSDALE; BOGART, 2013).

Suponhamos que a cada ponto de um espaço amostral se atribua um número. Teremos então uma função definida no espaço amostral. Esta função é chamada variável aleatória (ou variável estocástica), ou mais precisamente, função aleatória (ou função estocástica). Denota-se em geral por uma letra maiúscula X, Y, ... Em geral, uma variável aleatória tem algum significado físico, geométrico, ou outro qualquer (SPIEGEL, 1978).

Uma variável aleatória que toma um número finito, ou um número infinito enumerável de valores, é chamada *variável discreta*, ao passo que uma variável aleatória que toma um número infinito (não-enumerável) de valores, é uma *variável aleatória contínua*.

2.3 Estado da Arte

2.3.1 Probabilidade Intervalar

Do ponto de vista matemático, a idéia de probabilidade intervalar consiste em dada uma probabilidade, um número real P(A) = *p*, *p* é substituído por um intervalo que o contém. Do ponto de vista da implementação, *p* sendo não representável em um específico sistema de ponto flutuante, é substituído pelo menor intervalo que o contém propriamente. Com respeito a ambos pontos de vista, o intervalo não é uma probabilidade, apenas contém o valor da probabilidade, isto é, é uma aproximação da probabilidade exata (CAMPOS, 1997).

Intervalos constituem uma estrutura algébrica diferente da estrutura dos números reais ($\mathbb{R}$), e, devido a essa característica, não se pode substituir simplesmente um valor real por um intervalo e as operações reais pelas operações intervalares para resolver problemas de probabilidade. Portanto, deve-se introduzir condições adicionais para, durante o cálculo de probabilidade intervalar, preservar a semântica da função probabilidade.

A probabilidade intervalar proposta por Campos (CAMPOS, 1997) resolve problemas numéricos associados com o cálculo de probabilidades reais. A probabilidade intervalar é uma extensão da probabilidade real, mas que mantém toda a consistência semântica do cálculo probabilístico, buscando para isso recursos da análise intervalar (MOORE, 1976) (MOORE, 1979) e da aritmética de exatidão máxima (KULISCH; MIRANKER, 1981) (CAMPOS, 1997).

A probabilidade intervalar é realizada através de uma peculiar composição de funções, envolvendo extensões intervalares de funções reais. Inicialmente a um evento é associada a probabilidade real *p*. Em seguida, a *p* é associado um intervalo que a contém. Da mesma forma que uma extensão canônica para um número real é um intervalo que o contenha, assim também uma extensão natural para a probabilidade real é a probabilidade intervalar. Nesse sentido, a probabilidade real é a restrição da probabilidade intervalar e esta a extensão da probabilidade real. Então, se necessário, a probabilidade real é estendida para a probabilidade intervalar ou a probabilidade intervalar é restrita para a real (CAMPOS, 1997).

Segundo Campos (CAMPOS, 1997), a seguinte definição é válida:

Definição 3. *Um espaço de probabilidade intervalar é uma sextupla* $\langle \mathscr{A}, \mathbb{R}, \mathbb{IR}, \Omega, P, P_{\mathbb{IR}} \rangle$, *onde* $\mathscr{A}$ *é uma família de conjuntos,* Ω *é um elemento destacável de* $\mathscr{A}$, $\mathbb{R}$ *é o conjunto dos números reais,* $\mathbb{IR}$ *é o espaço dos intervalos,* *P uma função probabilidade e* $P_{\mathbb{IR}} : \mathscr{A} \rightarrow \mathbb{IR}$ *é uma função satisfazendo as seguintes propriedades:*

(A1) $\forall$ *A* $\in \mathscr{A}$ *e P(A)* $\in \mathbb{R}$*, então* $P_{\mathbb{IR}}(A) \in \mathbb{IR}$

(A2) *1* $\in P_{\mathbb{IR}}(\Omega)$

(A3) *0* $\in P_{\mathbb{IR}}(\varnothing)$

(A4) *Se* A_1, A_2*, ... é uma sequência de elementos de* $\mathscr{A}$ *com* $A_i \cap A_j = \varnothing$ *para todo* $i \neq j$*, então* $P_{\mathbb{IR}}(\bigcup_{n=1}^{\infty} A_n) = \sum_{n=1}^{\infty} P_{\mathbb{IR}}(A_n)$

A função $P_{\mathbb{IR}}$ satisfazendo as propriedades A1, A2, A3 e A4 acima é denominada uma probabilidade intervalar ou uma extensão intervalar de *P* ou ainda uma validação para *P*.

A distribuição de probabilidade de uma variável aleatória contínua, é caracterizada por sua função densidade de probabilidade, a qual satisfaz às propriedades (FELLER, 1968) (MEYER, 1983):

i) $f(x) \geq 0,$

ii) $\int_a^b f(x)dx = P(a \leq x \leq b),\ a < b,$

iii) $\int_{-\infty}^{\infty} f(x)dx = 1.$

O item (ii) indica que a probabilidade da variável aleatória assumir valor em um intervalo é dada pela integral da função nesse intervalo. Entretanto, o cálculo dessa probabilidade implica em resolver dois tipos de problemas: (i) encontrar primitivas na forma analítica, o que é possível no caso de algumas distribuições, mas não para todas, como no caso da Normal e (ii) o valor da probabilidade, em geral, é real, não necessariamente representável em computadores.

2.3.2 Estatística Descritiva Intervalar

No trabalho desenvolvido por Loreto (LORETO, 2006) foram definidos indicadores estatísticos com abordagem intervalar. Em sua tese, Loreto reuniu indicadores já definidos de forma intervalar, como a média intervalar, moda intervalar, variância intervalar, desvio padrão intervalar, covariância intervalar e coeficiente de correlação intervalar e definiu coeficiente de variação intervalar, mediana intervalar, quartil, decil e percentil intervalares.

A Figura 1 contém um quadro com todas as expressões dos indicadores intervalares definidos na tese de Loreto (LORETO, 2006).

Média Intervalar:	$\mathbf{ME}_v = [\underline{me}, \overline{me}] = \frac{1}{n}[\sum_{i=1}^{n} \underline{x}_i, \sum_{i=1}^{n} \overline{x}_i]$
Mediana Intervalar:	$\mathbf{MD}_v = [\underline{md}, \overline{md}] = \begin{cases} \frac{(\mathbf{x}_{(\frac{n}{2})} + \mathbf{x}_{(\frac{n}{2}+1)})}{2}, & \text{se } n \text{ for par} \\ (\mathbf{x}_{(\frac{n+1}{2})}), & \text{se } n \text{ for ímpar.} \end{cases}$
Moda Intervalar:	$\mathbf{MO}_v = [\underline{mo}, \overline{mo}] = [mo_{1\leq i\leq n}\{\underline{x}_i\}, mo_{1\leq i\leq n}\{\overline{x}_i\}]$
Amplitude total Intervalar:	$\mathbf{AT}_v = [\underline{at}, \overline{at}] = \begin{cases} [\underline{ma} - \overline{mi}, \overline{ma} - \underline{mi}], & \text{se } \underline{ma} \geq \overline{mi} \\ [0, \overline{ma} - \underline{mi}], & \text{se } \underline{ma} < \overline{mi}. \end{cases}$
Variância Intervalar:	$\mathbf{VA}_v = [\underline{va}, \overline{va}] = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x_i} - \mathbf{ME})^2$
Desvio padrão Intervalar:	$\mathbf{DP}_v = [\underline{dp}, \overline{dp}] = \sqrt{\mathbf{VA}} = +\sqrt{[\underline{va}, \overline{va}]} = [+\sqrt{\underline{va}}, +\sqrt{\overline{va}}]$
Coeficiente de variação Intervalar:	$\mathbf{CV}_v = [\underline{cv}, \overline{cv}] = \frac{\mathbf{DP}}{\mathbf{ME}} = \frac{[\underline{dp}, \overline{dp}]}{[\underline{me}, \overline{me}]}$
Covariância Intervalar:	$\mathbf{CO}_v = [\underline{co}, \overline{co}] = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x_i} - \mathbf{ME_X})(\mathbf{y_i} - \mathbf{ME_Y})$
Coeficiente de correlação Intervalar:	$\mathbf{CC}_v = [\underline{cc}, \overline{cc}] = \frac{\mathbf{CO}}{\mathbf{DP_X DP_Y}} = (\frac{[\underline{co}, \overline{co}]}{[\underline{dp}_X, \overline{dp}_X][\underline{dp}_Y, \overline{dp}_Y]})$
Separatrizes Intervalares:	$\mathbf{Q}_v = [\underline{q}, \overline{q}], \mathbf{D}_v = [\underline{d}, \overline{d}], \mathbf{P}_v = [\underline{p}, \overline{p}], pos = (n-1)\alpha + 1$

Figura 1: Formulário dos Indicadores Estatísticos Intervalares

Para verificar se a utilização da extensão intervalar retorna como resposta intervalos que englobam a resposta real exata, Loreto (LORETO, 2006) utilizou um estudo de caso onde comparou os resultados reais com os intervalares, para cada um dos indicadores. A verificação da qualidade de aproximação nos intervalos solução ocorreu através de cálculos das medidas de erros. Em todos os exemplos considerados no trabalho, verificou-se que a qualidade no intervalo solução foi mantida.

2.3.3 Intervalos Encapsuladores para Probabilidades de Variáveis Aleatórias Contínuas

Santos (SANTOS, 2010) utilizou o métodos de Simpson intervalar, definido por Caprani *et al* (CAPRANI; MADSEN; NIELSEN, 2002), para calcular probabilidades intervalares ou intervalos encapsuladores para as variáveis aleatórias contínuas. Foram desenvolvidas expressões teóricas de como calcular os intervalos encapsuladores para as variáveis aleatórias estudadas.

Para as variáveis Exponencial e Normal os intervalos encapsuladores ou as probabilidades validadas foram obtidos com o método de Simpson intervalar. Para a Uniforme o intervalo encapsulador foi definido a partir da função densidade.

A verificação dos resultados obtidos com a definição intervalar dessas variáveis foi feita através de exemplos, onde calculou-se os resultados de probabilidade real para cada variável e comparou-se com as probabilidades intervalares resultantes, calculadas através da implementação utilizando IntLab (MATLAB, 2014). Os resultados obtidos por Santos (SANTOS, 2010) foram utilizados por Varjão (VARJÃO, 2011) para fins de validação das implementações feitas por ele utilizando IntPy (INTPY, 2014).

2.3.3.1 Uniforme

A distribuição Uniforme possui densidade com primitiva na forma analítica, entretanto valores da probabilidade podem não ser representáveis em computadores.

A densidade de uma Uniforme X no intervalo $A = [a,b]$ é

$$f(x) = \begin{cases} \dfrac{1}{b-a}, & x \in [a,b], \\ 0, & x \notin [a,b]. \end{cases}$$

Seja $B = [c,d] \subset \mathbb{R}$. Para calcular $P(c < X \leq d)$ tem-se as seguintes situações possíveis:

(i) Se $d \leq a$,
$$P(c < X \leq d) = 0$$

(ii) Se $c < a \leq d \leq b$,
$$P(c < X \leq d) = P(a \leq X \leq d) = \int_a^d \tfrac{1}{b-a}dx = \tfrac{d-a}{b-a}.$$

(iii) Se $a < c \leq b \leq d$,
$$P(c < X \leq d) = P(c \leq X \leq b) = \int_c^b \tfrac{1}{b-a}dx = \tfrac{b-c}{b-a}.$$

(iv) Se $a \leq c < d \leq b$,
$$P(c < X \leq d) = P(c \leq X \leq d) = \int_c^d \tfrac{1}{b-a}dx = \tfrac{d-c}{b-a}.$$

(v) Se $c \leq a < b \leq d$,
$$P(c < X \leq d) = P(a \leq X \leq b) = \int_a^b \tfrac{1}{b-a}dx = \tfrac{b-a}{b-a} = 1.$$

(vi) Se $c \geq b$,
$$P(c < X \leq d) = 0.$$

Analisando as probabilidades apresentadas para a distribuição Uniforme com entradas reais, observa-se que os cálculos anteriores poderiam ter sido realizados através da seguinte definição:

Definição 4.

$$P(c < X \leq d) = \begin{cases} \dfrac{w([c,d] \cap [a,b])}{b-a}, & [c,d] \cap [a,b] \neq 0, \\ 0, & [c,d] \cap [a,b] = 0. \end{cases}$$

onde w é o diâmetro do intervalo.

A densidade da distribuição Uniforme tem primitiva na forma analítica, portanto, pelo Teorema Fundamental do Cálculo Integral, pode-se calcular qualquer integral definida, por exemplo

$$P(c < X \leq d) = \int_c^d f(x)dx,$$

diretamente desta primitiva.

Entretanto, a existência da primitiva na forma analítica não impediu a ocorrência de problemas numéricos relacionados com o cálculo das integrais, justificando a busca para um intervalo encapsulador para a probabilidade real.

A definição intervalar da função densidade de probabilidade com distribuição Uniforme, é uma proposta de como calcular um intervalo encapsulador para probabilidades da distribuição Uniforme no intervalo $[a,b]$.

2.3.3.2 Exponencial

Uma variável aleatória contínua com função densidade dada por

$$f(x) = \begin{cases} \alpha e^{-\alpha x}, & 0 \leq x \leq \infty, \\ 0, & x < 0, \end{cases}$$

com $\alpha \in \mathbb{R}_+^*$ (conjunto dos Números Reais excluindo os negativos e o zero), é uma variável aleatória Exponencial.

Seja $A = [a,b] \subset \mathbb{R}$. Para calcular $P(a \leq x \leq b)$, tem-se as seguintes situações possíveis:

i) Se $b \leq 0$, então

$$P(a < X \leq b) = 0.$$

ii) Se 0 ∈ (a, b), então

$$P(a < X \leq b) = P(0 \leq X \leq b) = \int_0^b \alpha e^{-\alpha x} dx.$$

iii) Se $a \geq 0$, então

$$P(a \leq X \leq b) = \int_a^b \alpha e^{-\alpha x} dx.$$

A função f tem primitiva na forma analítica, então, pelo Teorema Fundamental do Cálculo Integral, pode-se calcular a integral definida de f diretamente desta primitiva. O número e e suas potências, de expoentes não nulos, são irracionais, sendo seus valores aproximados, portanto sujeitos a erros de arredondamento (SANTOS, 2010).

Para gerar o intervalo encapsulador da exponencial, Santos (SANTOS, 2010) utilizou o método de Simpson Intervalar, o qual é uma alternativa para encontrar um intervalo que aproxime o valor desta integral, com a finalidade de obter um intervalo com amplitude tão pequena quanto possível que contenha a integral. Para a aplicação do método é necessário determinar extensões intervalares para a densidade f e sua derivada de quarta ordem,

$$f^{(4)}(x) = \begin{cases} \alpha^5 e^{-\alpha x}, & 0 < x < \infty, \\ 0, & x < 0. \end{cases}$$

Como a probabilidade da variável aleatória X assumir valores negativos é 0 pode-se restringir, na aplicação do método de Simpson intervalar, o domínio de f e de $f^{(4)}$ ao conjunto $\mathbb{R}_+$. Sejam $f|\mathbb{R}_+$ e $f^{(4)}|\mathbb{R}_+$ as restrições de f e de $f^{(4)}$, isto é

$$f|\mathbb{R}_+(x) = \alpha e^{-\alpha x}, 0 \leq x < \infty,$$

e

$$f^{(4)}|\mathbb{R}_+ = \alpha^5 e^{-\alpha x}, 0 < x < \infty,$$

são decrescentes e as funções intervalares

$$F_E(X) = \alpha[e^{-\alpha\bar{x}}, e^{-\alpha\underline{x}}], \quad \underline{x} > 0$$

e

$$G_E(X) = \alpha^5[e^{-\alpha\bar{x}}, e^{-\alpha\underline{x}}],$$

como extensão intervalar para a função densidade $f_X|\mathbb{R}_+$ e a derivada $f_X^{(4)}|\mathbb{R}_+$, respectivamente.

2.3.3.3 Normal

A função $f(x) = \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}}$ não tem primitiva na forma analítica portanto, não é possível aplicar o Teorema Fundamental do Cálculo Integral. Contudo, métodos de integração numérica podem ser empregados para calcular integrais da forma acima e gerar tabelas de probabilidades (SANTOS, 2010).

O método proposto por Santos (SANTOS, 2010) consiste em utilizar extensões intervalares F e G para f e $f^{(4)}$, respectivamente, que, quando computados, produzam intervalos encapsuladores para a probabilidade procurada.

2.3.4 Computação Científica Auto Validável em Python

Varjão (VARJÃO, 2011) desenvolveu um pacote para a linguagem de programação Python (PYTHON, 2014), chamado IntPy, o qual implementa o tipo intervalo e as operações sobre este tipo.

A última versão do pacote implementa intervalos com arredondamentos direcionados, reconhece entrada de intervalos como strings e realiza as operações sobre intervalos (VARJÃO, 2011).

Foram implementadas funções matemáticas específicas e o cálculo de probabilidades intervalares para as variáveis aleatórias contínuas. Após, realizaram-se comparações de desempenho com rotinas que já haviam sido implementadas em outro pacote de desenvolvimento intervalar, o Intlab.

Em intervalos, o suporte para realização das operações aritméticas é o uso de arredondamentos direcionados (KULISCH; MIRANKER, 1981) nos extremos dos intervalos, ou seja, o limite inferior é arredondado para baixo ($\triangledown$) e o superior para cima ($\triangle$). Por exemplo, o intervalo $[2.71828, 3.14159]$ se arredondado, utilizando a técnica de arredondamento direcionado, para 4 casas decimais, resulta no intervalo $[2.7182, 3.1416]$. Esta forma de arredondamento foi implementada por Varjão (VARJÃO, 2011).

Abaixo serão mostradas algumas funções desenvolvidas pro Varjão (VARJÃO, 2011) para o pacote intervalar de Python.

A Figura 2 mostra exemplos de execução da função potência, onde são apresentadas chamadas em Python, a qual recebe um intervalo como entrada e um n, o qual é a potência a ser utilizada.

Chamada em Python	Exemplo 1: X = [-3, 2], n = 2
pow([x1, x2], n)	>>> X = IReal(-3, 2) ou >>> pow (IReal(-3,2), 2) >>> pow(X, 2)
	Resultado: [0.0, 9.0]
	tempo de processamento = 0.0001 s
	Exemplo 2: X = [0.25, 0.27], n = 4
	>>> X = IReal(0.25, 0.27) ou >>> pow (IReal(0.25, 0.27), 4) >>> pow(X, 4)
	Resultado: [0.00390625, 0.005314410000000002]
	tempo de processamento = 0.0001 s
	Exemplo 3: X = [1.0, 2.0], n = 2
	>>> X = IReal(1.0, 2.0) ou >>> pow(IReal(1.0, 2.0), 2) >>> pow(X, 2)
	Resultado: [1.0, 4.0]
	tempo de processamento = 0.0001 s
	Exemplo 4: X = [2.7182818284590451, 3.1415926535897931], n =2
	>>> X = IReal(2.7182818284590451, 3.1415926535897931) >>> pow(X, 2)
	Ou
	>>> pow(IReal(2.7182818284590451, 3.1415926535897931),2)
	Resultado: [7.3890560989306495, 9.869604401089358]
	tempo de processamento = 0.0001 s

Figura 2: Exemplos da Função Potência

Os resultados obtidos mostram o intervalo solução e o tempo de processamento da chamada da função para cada um dos exemplos.

Assim como a função mostrada acima, também foram implementadas as funções exponencial, logarítmica, raiz quadrada, seno, arco seno, seno hiperbólico, cosseno, arco cosseno, cosseno hiperbólico, tangente, arco tangente e tangente hiperbólica.

Neste mesmo trabalho foi utilizado IntPy para calcular intervalos encapsuladores para as variáveis aleatórias Uniforme, Exponencial e Pareto.

A Figura 3 mostra exemplos de execução da função desenvolvida para a variável Uniforme.

Chamada em Python	Exemplo 1: A = [0.0, 3.0], B = [1.0,2.0]
import dunif dunif(a, b, c, d)	dunif(0.0, 3.0, 1.0, 2.0)
	[0.33333333333331, 0.33333333333337]
	tempo de processamento = 0.0005s
	Exemplo 2: A = [0.0, 3.0], B = [1.0/2.0, 4.0/7.0]
	dunif(0.0, 3.0, 1.0/2.0, 4.0/7.0)
	[0.02380952380952,0.02380952380953]
	tempo de processamento = 0.003s

Figura 3: Exemplos do Cálculo de Intervalos Encapsuladores para Uniforme

Na Figura 4 são apresentados os intervalos encapsuladores (Ip) e os tempos de processamento (TIp) comparados nos ambientes IntPy e Intlab. Salienta-se que todos os resultados apresentados no Intlab foram obtidos por Santos (SANTOS, 2010) e citados por Varjão (VARJÃO, 2011) para fins de comparação.

Probabilidades	Biblioteca	Ip	TIp
P(1≤ X ≤ 2)	IntPy	[0.33333333333331, 0.33333333333337]	0.0005s
	Intlab	[0.33333333333333, 0.33333333333334]	0.0460s
P(1/2 ≤ T ≤ 4/7)	IntPy	[0.02380952380952, 0.02380952380954]	0.0003s
	Intlab	[0.02380952380952,0.02380952380953]	0.0630s

Figura 4: Exemplos do Cálculo de Intervalos Encapsuladores para Uniforme

Pela Figura 4 é possível perceber que o tempo de processamento do pacote desenvolvido por Varjão (VARJÃO, 2011) é menor do que o tempo do Intlab.

A Figura 5 mostra exemplos de execução da função desenvolvida para a variável Exponencial.

Chamada em Python	Exemplo 1: X = [-inf, 50], c = 0.01, p = 50
import dexp dexp(x1, x2, c, p)	dexp(-inf, 50, 0.01, 50)
	[0.39346934028736, 0.39346934028737]
	tempo de processamento = 0.0002s
	Exemplo 2: X = [20, 50], c = 0.01, p = 50
	dexp(20, 50, 0.01, 50)
	[0.2122009336534, 0.21220009336535]
	tempo de processamento = 0.0052s

Figura 5: Exemplos do Cálculo de Intervalos Encapsuladores para Exponencial

Na figura a seguir são apresentados os intervalos encapsuladores (Ip) para a distribuição Exponencial e os tempos de processamento (TIp) comparados nos ambientes IntPy e Intlab.

Probabilidades	Biblioteca	Ip	TIp
P(T < 50)	IntPy	[0.39346934028736, 0.39346934028737]	0.0002s
	Intlab	[0.39346934028736, 0.39346934028737]	0.0150s
P(20 ≤ T ≤ 50)	IntPy	[0.21220009336534, 0.21220009336535]	0.0052s
	Intlab	[0.2122009336534, 0.21220009336535]	1.1410s

Figura 6: Exemplos do Cálculo de Intervalos Encapsuladores para Exponencial

Pela Figura 6 é possível perceber que o tempo de processamento do pacote desenvolvido por Varjão (VARJÃO, 2011) é menor do que o tempo do Intlab.

Na Figura 7 são apresentados exemplos de execução da função desenvolvida em Python para a variável Pareto.

Chamada em Python	Exemplo 1: X = [1.0, 2.0], c = 0.25 , d = 1.0
	dpareto(1.0, 2.0, 0.25, 1.0)
	[0.15910358474628, 0.15910358474629]
	tempo de processamento = 0.0001s
	Exemplo 2: X = [1.0, 2.0], c = 0.50, d = 1.0
import dpareto dpareto(x1, x2, c, d)	dpareto(1.0, 2.0, 0.50, 1.0)
	[0.29289321881345, 0.29289321881346]
	tempo de processamento = 0.001s
	Exemplo 3: X = [1.0, 2.0], c = 0.75, d = 1.0
	dpareto(1.0, 2.0, 0.75, 1.0)
	[0.40539644249863, 0.40539644249864]
	tempo de processamento = 0.001s
	Exemplo 4: X = [1.0, 2.0], c = 0.1, d = 1.0
	dpareto(1.0, 2.0, 0.1, 1.0)
	[0.5, 0.5]
	tempo de processamento = 0.002s
	Exemplo 5: X = [1.0, 2.0], c = 0.25, d = 0.5
	dpareto(1.0, 2.0, 0.25, 0.5)
	[0.24367151233022033, 0.24367151233022039]
	tempo de processamento = 0.002s
	Exemplo 6: X = [1.0, 2.0], c = 5.0, d = 1.3
	dpareto(1.0, 2.0, 5.0, 1.3)
	[0.88397093759998, 0.88397093750000022]
	tempo de processamento = 0.001s

Figura 7: Exemplos do Cálculo de Intervalos Encapsuladores para Pareto

A figura a seguir apresenta os intervalos encapsuladores (Ip) para a distribuição de Pareto e os tempos de processamento (TIp) comparados nos ambientes IntPy e Intlab.

Probabilidades	Biblioteca	Ip	TIp
P (1 < X < 2)	IntPy	[0.15910358474628, 0.15910358474629]	0.0001s
	Intlab	[0.15910358474628, 0.15910358474629]	0.0310s
P (1 < X < 2)	IntPy	[0.29289321881345, 0.29289321881346]	0.0001s
	Intlab	[0.29289321881345, 0.29289321881346]	0.0310s
P (1 < X < 2)	IntPy	[0.40539644249863, 0.40539644249864]	0.0001s
	Intlab	[0.40539644249863, 0.40539644249864]	0.0780s
P (1 < X < 2)	IntPy	[0.5, 0.5]	0.0002s
	Intlab	[0.50000000000000, 0.50000000000000]	0.0790s
P (1 < X < 2)	IntPy	[0.24367151233022033,0.24367151233022039]	0.0002s
	Intlab	[0.24367151233022, 0.24367151233023]	0.0310s
P (1 < X < 2)	IntPy	[0.88397093759998, 0.88397093750000022]	0.0001s
	Intlab	[0.88397093749998, 0.88397093750002]	0.0310s

Figura 8: Exemplos do Cálculo de Intervalos Encapsuladores para Pareto

Pela Figura 8 é possível perceber que o tempo de processamento do pacote desenvolvido por Varjão (VARJÃO, 2011) para a função com distribuição Pareto é menor do que o tempo do Intlab.

Salienta-se que no trabalho desenvolvido por Varjão (VARJÃO, 2011) foi implementado e calculado o intervalo encapsulador para Uniforme, Exponencial e Pareto, também utilizadas na presente dissertação. Porém, não havia sido verificada a qualidade dos intervalos gerados pela implementação no IntPy e no Intlab, o que então é realizado no presente trabalho. Analisar a qualidade de um intervalo solução é de fundamental importância porque garante a exatidão da resposta, bem como a aproximação em relação a solução exata.

2.3.5 Método de Simpson Intervalar para Integração Numérica

O método de Simpson com entradas reais pode ser encontrado e melhor detalhado em Ruggiero (RUGGIERO; LOPES, 1996). A seguir, apresenta-se a definição intervalar existente para o método, desenvolvida por Caprani *et al* (CAPRANI; MADSEN; NIELSEN, 2002) e aplicada na presente dissertação.

Dada a função contínua $f : \mathbb{R} \to \mathbb{R}$ sobre o intervalo A = [a,b], o objetivo é calcular

$$S = \int_a^b f(x)dx. \tag{2}$$

Como f é contínua, f é integrável a Riemann. Se G é uma primitiva de f em [a,b] tem-se, pelo Teorema Fundamental do Cálculo Integral, que

$$S = G(b) - G(a)$$

$G(a)$ ou $G(b)$ podem não ter uma representação exata no computador, o que implica que seu valor tem de ser aproximado por um número de máquina. Se não é possível encontrar uma expressão analítica para uma primitiva de f, métodos numéricos são essenciais para encontrar uma solução numérica do problema (SANTOS, 2010). A idéia básica da integração numérica é a substituição da função f por um polinômio que a aproxime razoavelmente no intervalo [a,b], assim o problema se resume a resolver uma integral de polinômio.

O método de Simpson Intervalar (CAPRANI; MADSEN; NIELSEN, 2002) é uma extensão intervalar do método de Simpson real (RUGGIERO; LOPES, 1996). O método na forma intervalar é fundamentado na propriedade aditiva da integral definida e no teorema do valor médio para integrais.

O método de Simpson Intervalar descrito em Caprani *et al* (CAPRANI; MADSEN; NIELSEN, 2002) admite conhecidas as extensões intervalares inclusões monotônicas F e G de f e $f^{(4)}$, respectivamente.

Como $f(x) \in F(X)$, para todo $x \in (X \in \mathbb{R})$ então,

$$IS = \sum_{i=1}^{n} \int_{a_{i-1}}^{a_i} f(x)dx = \sum_{i=1}^{n} w(A_i)F(A_i).$$

Assim, tem-se:

$$IS_i = \int_{a_{i-1}}^{a_i} f(x)dx = \frac{w(A_i)}{6}(F(a_{i-1}) + 4F(m(A_i)) + F(a_i)) - \frac{(w(A_i))^5}{2880}G(A_i).$$

onde $F(a_{i-1})$, $F(m(A_i))$ e $F(a_i)$ são intervalos degenerados de números reais. Portanto,

$$S \subseteq IS = \sum_{i=1}^{n} IS_i. \tag{3}$$

Assumindo que G é uma extensão intervalar linear de $f^{(4)}$, tem-se

$$w(IS_i) = \frac{1}{2880}w(A_i)^5 w(G(A_i)) \leq \frac{1}{2880}w(A_i)^5 K_G w(A_i) = K_2 w(A_i)^6.$$

Se $w(A_i) = w(A)/n$ tem-se que

$$w(IS) = \sum_i w(IS_i) \leq K_3/n^5,$$

pode-se esperar que dobrando-se o valor de n reduz-se o erro de truncamento por um fator de 32. O intervalo IS depende das extensões intervalares usadas e do valor de n (SANTOS, 2010).

O presente capítulo introduziu conceitos da matemática intervalar importantes para o desenvolvimento da dissertação. Também destacou os principais trabalhos relacionados, apresentando o estado da arte na área da estatística intervalar. Conclui-se o capítulo apresentando a definição do método de Simpson Intervalar, utilizado como método para resolução de integral com entradas intervalares. O capítulo a seguir traz as funções densidade de probabilidade das variáveis aleatórias contínuas estudadas e implementadas na presente dissertação.

3 FUNÇÕES DENSIDADE DE PROBABILIDADE DAS VARIÁVEIS ALEATÓRIAS CONTÍNUAS

O capítulo apresenta definições e fórmulas de cada uma das distribuições exploradas na dissertação. Primeiramente, são mostradas as funções com entradas reais, para as quais utiliza-se as definições encontradas em Naghettini (NAGHETTINI; PINTO, 2007). Na seção que segue, apresentam-se as fórmulas intervalares para cada função. Para Uniforme, Exponencial e Normal as definições encontram-se no trabalho desenvolvido por Santos (SANTOS, 2010). Já as definições para Gama e Pareto foram realizadas na presente dissertação como um dos objetivos da mesma.

3.1 Forma Real

Nesta seção serão apresentadas as funções densidade de probabilidade na sua forma real. Cada distribuição está descrita em uma subseção.

3.1.1 Distribuição Uniforme

A distribuição uniforme é a mais simples distribuição contínua, entretanto é muito utilizada dentro da teoria de probabilidade. Esta distribuição é aplicada quando a probabilidade de acontecer um fenômeno é constante em intervalos de mesma dimensão.

Uma variável aleatória contínua X, cujos valores possíveis x encontram-se restritos à condição a $\leq$ x $\leq$ b, é distribuída uniformemente se a probabilidade de que ela esteja compreendida em qualquer intervalo [m, n], contido em [a, b], for diretamente proporcional ao comprimento (m-n). Se a constante de proporcionalidade for denotada por ρ, então,

$$P(m \leq X \leq n) = \rho(m-n) \quad se \quad a \leq m \leq n \leq b$$

Uma vez que P(a $\leq$ X $\leq$ b) = 1, é fácil verificar que ρ = 1/(b-a). Portanto, para qualquer a $\leq$ x $\leq$ b, a função de distribuição acumulada da distribuição Uniforme é dada por

$$F_x(x) = \frac{x-a}{b-a}. \tag{4}$$

Se x $<$ a, $F_x(x)$ = 0 e, se x $>$ b, $F_x(x)$ = 1. A função densidade da distribuição Uniforme decorre da diferenciação da Equação 4 e tem a seguinte expressão:

$$f_x(x) = \frac{1}{b-a} \quad se \quad a \leq x \leq b.$$

Assim, é possível calcular a probabilidade do valor calculado estar compreendido entre a e b.

3.1.2 Distribuição Exponencial

Esta função de densidade de distribuição é frequentemente utilizada para determinar a vida útil de equipamentos eletrônicos e do tempo entre ocorrências de eventos sucessivos, como por exemplo, o tempo entre chegadas de clientes a uma agência bancária (VILCHES, 2011).

A distribuição Exponencial possui inúmeras outras aplicações em diversas áreas do conhecimento humano e, em particular, às variáveis hidrológicas (NAGHETTINI; PINTO, 2007). Define-se a função densidade de probabilidade da distribuição Exponencial de parâmetro α por:

$$f(x) = \begin{cases} \alpha e^{-\alpha x}, & 0 \leq x \leq \infty, \\ 0, & x < 0. \end{cases}$$

com $\alpha > 0$.

A probabilidade de que um número $x \in$ (a, b) é:

$$P(a \leq x \leq b) = \alpha \int_a^b e^{-\alpha x} dx = e^{-a\alpha} - e^{-b\alpha}$$

Sendo a aplicação da integral a resolução de um número x estar compreendido no intervalo entre os valores a e b.

3.1.3 Distribuição Normal

A distribuição Normal também é conhecida como de Gauss, em referência ao emprego pioneiro dessa distribuição no tratamento dos erros aleatórios de medidas experimentais, atribuído ao matemático alemão Karl Friedrich Gauss (1777-1855). A distribuição Normal é utilizada para descrever o comportamento de uma variável aleatória que flutua de forma simétrica em torno de um valor central. Além disso, a distribuição Normal está na origem de toda a formulação teórica acerca da construção de intervalos de confiança, testes estatísticos de hipóteses, bem como da teoria de regressão e correlação (NAGHETTINI; PINTO, 2007).

Defini-se a função de densidade de probabilidade da distribição Normal por:

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}, -\infty < x < +\infty,$$

onde $\mu \geq 0$ e $\sigma > 0$.

Diz-se que X é normalmente distribuída com média μ e desvio padrão σ, ou, sinteticamente, que $X \sim N(\mu, \sigma)$ (NAGHETTINI; PINTO, 2007).

3.1.4 Distribuição de Pareto

A distribuição de Pareto recebeu esse nome devido ao economista italiano Vilfredo Pareto que utilizou esse distribuição, no início do século XX, para descrever a distribuição da riqueza na Itália, onde uma minoria possuía a maior parte da riqueza e a maioria da população possuía uma pequena parte. Mais precisamente, ele reparou que 20% da população possuía 80% da riqueza. Em seguida, constatou que esta assimetria se verificava nas mesmas proporções em outros países. Foi baseado neste fato que, mais tarde, o gestor de negócios, Joseph M. Juran, estabeleceu o Princípio de Pareto ou a regra dos 80-20 que diz que, em muitos fenômenos, 80% dos efeitos vem de 20% das causas. Por exemplo, num negócio de vendas, 80% das vendas provêm de 20% dos clientes (LAPPONI, 1995).

Define-se a função de densidade de probabilidade da distribuição de Pareto, por (VILCHES, 2011):

$$f(x) = \begin{cases} \frac{\alpha\beta^\alpha}{x^{\alpha+1}}, & se \quad x \geq \beta \\ 0, & se \quad x < \beta \end{cases}$$

$\alpha > 1$ e $\beta > 0$.

$$\int_{-\infty}^{+\infty} f(x)dx = \int_{\beta}^{+\infty} \frac{\alpha\beta^\alpha}{x^{\alpha+1}} dx$$

Pode-se, então, calcular a probabilidade de um valor distribuido conforme Pareto, através da integral apresentada acima.

3.1.5 Distribuição Gama

A função $\Gamma = \Gamma(x)$ dá origem à distribuição Gama, que é utilizada nos fenômenos limitados, como por exemplo, os intervalos de tempo de espera numa fila de banco ou para analisar o tempo de parmenência de pacientes num hospital (VILCHES, 2011).

A função de densidade de probabilidade Gama, de parâmetros $\lambda > 0$ e $v > 0$, ambos reais, é definida por (VILCHES, 2011):

$$f(x) = \begin{cases} \dfrac{\lambda^{\nu}}{\Gamma(\nu)} e^{-\lambda x} x^{\nu-1}, & se \ \ x > 0 \\ 0, & outro \ \ caso. \end{cases}$$

É possível notar que, se $v = 1$, temos a densidade de probabilidade exponencial. Utilizando a definição da função gama, obtem-se que:

$$\int_0^{+\infty} e^{-\lambda x} x^{\nu-1} dx = \frac{\Gamma(\nu)}{\lambda^{\nu}}.$$

Na presente seção apresentou-se as definições com entradas reais para cada densidade de probabilidade das variáveis aleatórias. Agora, o objetivo da próxima seção é apresentar as definições intervalares já existentes para Uniforme, Exponencial e Normal, além das definições realizadas no presente trabalho para as funções com distribuição Gama e Pareto (VARJÃO, 2011).

3.2 Forma Intervalar

A presente seção apresenta as definições intervalares para cada distribuição. Salienta-se que para Uniforme, Exponencial e Normal, as definições intervalares foram realizadas por Santos (SANTOS, 2010). A fim de completar as definições intervalares de densidade de probabilidade das variáveis aleatórias contínuas, o presente trabalho propõe definições intervalares para Pareto e Gama.

3.2.1 Distribuição Uniforme

A distribuição Uniforme possui densidade com primitiva na forma analítica, como apresentado na Seção 2.2.3.1, podendo ser resolvida pela aplicação do Teorema Fundamental do Cálculo Integral.

Analisando-se os possíveis casos de soluções para esta distribuição, Santos (SANTOS, 2010) definiu a função $UNIF$, a qual é uma proposta de como calcular um intervalo encapsulador para probabilidades da Uniforme no intervalo [a, b]. Enfatiza-se que $UNIF$ é uma extensão intervalar monotônica para a inclusão (MOORE, 1966) e fornece o intervalo de menor comprimento para encapsular probabilidades para esta distribuição.

$$UNIF(c,d) = \left[\underline{\frac{w([c,d] \cap [a,b])}{b-a}}, \overline{\frac{w([c,d] \cap [a,b])}{b-a}} \right], [c,d] \cap [a,b] \neq \emptyset.$$

onde $[c,d]$ e $[a,b]$ são intervalos $\subset \mathbb{R}$.

Se $[c,d] \cap [a,b] = \emptyset$, então $UNIF(c,d) = 0_\varepsilon$, onde 0_ε é o menor intervalo de máquina que contém o zero (SANTOS, 2010).

$UNIF$, definida com suporte na matemática intervalar (MOORE, 1966)(MOORE, 1979)(MOORE; KEARFOTT; CLOUD, 2009) e na aritmética de exatidão máxima (KULISCH, 2008)(KULISCH; MIRANKER, 1981), fornece o intervalo de menor amplitude para encapsular probabilidades para a distribuição Uniforme (SANTOS, 2010).

3.2.2 Distribuição Exponencial

Segundo Santos (SANTOS, 2010), como a probabilidade da variável aleatória X, apresentada na Seção 2.2.3.2, assumir valores negativos é 0 pode-se restringir, na aplicação do método de Simpson intervalar, o domínio de f e de $f^{(4)}$ ao conjunto $\mathbb{R}_+$.

A extensão intervalar definida por Santos (SANTOS, 2010) para esta função densidade de probabilidade é calculada através das seguintes funções:

$$F_E(X) = \alpha[e^{-\alpha\bar{x}}, e^{-\alpha\underline{x}}], \quad \underline{x} > 0 \ \ e \ \ G_E(X) = \alpha^5[e^{-\alpha\bar{x}}, e^{-\alpha\underline{x}}],$$

Santos (SANTOS, 2010) verifica que

$$F_E(X) = f|\mathbb{R}_+(X) = \tilde{f}|\mathbb{R}_+(X) = \bar{f}|\mathbb{R}_+(X)$$

e

$$G_E(X) = f^{(4)}|\mathbb{R}_+(X) = \tilde{f}^{(4)}|\mathbb{R}_+(X) = \bar{f}^{(4)}|\mathbb{R}_+(X),$$

onde $\tilde{f}$ é a cobertura intervalar da função *f* (MOORE, 1966)(MOORE, 1979)(MOORE; KEARFOTT; CLOUD, 2009) e $\bar{f}$ é extensão unida da função *f* (MOORE, 1966)(MOORE, 1979)(MOORE; KEARFOTT; CLOUD, 2009).

Teorema 1. *G_E é uma extensão intervalar monotônica linear para $f^{(4)}|\mathbb{R}_+$.*

Prova 1. *(SANTOS, 2010)*

3.2.3 Distribuição Normal

A partir da definição com entradas reais, Santos (SANTOS, 2010) definiu a extensão intervalar para a função densidade de probabilidade da variável aleatória com distribuição Normal da seguinte maneira:

$$F_N(X) = \frac{1}{\sqrt{2\pi}} \begin{cases} \left[e^{\frac{-\bar{x}^2}{2}}, e^{\frac{-\underline{x}^2}{2}}\right], & se \ \underline{x} > 0, \\ \left[e^{\frac{-\underline{x}^2}{2}}, e^{\frac{-\bar{x}^2}{2}}\right], & se \ \bar{x} < 0, \\ \left[e^{-max\{\frac{\underline{x}^2}{2}, \frac{\bar{x}^2}{2}\}}, 1\right], & se \ 0 \in X, \end{cases}$$

a qual é uma extensão intervalar inclusão monotônica para f (SANTOS, 2010).

3.2.4 Distribuição de Pareto

Seja $X \sim P[a,b]$, $a < b$, uma variável aleatória com distribuição de Pareto com parâmetros α e c. Para computar o intervalo encapsulador da probabilidade intervalar para esta variável aleatória utiliza-se a seguinte fórmula (FINGER et al., 2012):

$$F_P(X) = \alpha \frac{c^\alpha}{X^{\alpha+1}} = \alpha \left(\left[\frac{c^\alpha}{\bar{x}^{\alpha+1}}, \frac{c^\alpha}{\underline{x}^{\alpha+1}} \right] \right), \ para \ 0 \notin X = [\underline{x}, \bar{x}],$$

onde α e $c \in \mathbb{R}$.

Define-se a função densidade aplicando a extensão intervalar, ou seja, para cada operação de $F_P(X)$, se é conhecido o intervalo X, computa-se a imagem através da aritmética intervalar definida por Moore (MOORE, 1966) (MOORE, 1979).

3.2.5 Distribuição Gama

Seja X uma Gama com parâmetros λ e $v \in \mathbb{R}$. Para computar o intervalo encapsulador da densidade de probabilidade intervalar para esta variável aleatória, define-se a função densidade como extensão intervalar:

$$F_G(X) = \frac{\Gamma(v)}{\lambda^v} . (e^{[-\lambda \underline{x}, -\lambda \bar{x}]} . [\underline{x}^{v-1}, \bar{x}^{v-1}])$$

O presente capítulo teve como objetivo apresentar as definições já desenvolvidas por Santos (SANTOS, 2010) para as funções de densidade de probabilidade das variáveis aleatórias com distribições Uniforme, Exponencial e Normal. Além disso, um dos objetivos específicos da dissertação apresenta-se neste capítulo, onde são definidas funções de densidade de probabilidade com distribuições Gama e Pareto através da aritmética intervalar. Nos capítulos seguintes, são utilizadas as definições aqui apresentadas para computar os intervalos encapsuladores, onde cada função é implementada e analisada conforme critérios de qualidade e complexidade.

4 ANÁLISE DA QUALIDADE DOS INTERVALOS ENCAPSULADORES

O objetivo do presente capítulo é analisar os intervalos encapsuladores obtidos para cada variável definida de forma intervalar. Após apresentar as definições das variáveis Uniforme, Exponencial e Normal, realizadas por Santos (SANTOS, 2010) e em seguida as definições realizadas nessa dissertação, das variáveis Pareto e Gama, o próximo passo é implementar cada uma das distribuições. Para isso, analisa-se separadamente as distribuições, a fim de verificar as soluções que podem ser utilizadas no desenvolvimento dos algoritmos. As seções deste capítulo estão separadas por distribuição, apresentando em cada uma delas suas soluções de implementação tanto na forma com entradas reais quanto na forma intervalar, com o objetivo de comparar os resultados obtidos em cada solução proposta e justificar qual retorna um intervalo solução com melhor qualidade.

A linguagem escolhida para implementar os métodos foi Python (PYTHON, 2014), para o qual se utiliza o pacote intervalar IntPy (INTPY, 2014). A escolha se deu através de comparações anteriores entre os ambientes intervalares mais utilizados na literatura. Comparando-se os resultados obtidos no IntPy, IntLab, Maple Intervalar e C-XSC, o pacote desenvolvido na linguagem Python foi o que demonstrou os melhores resultados, retornando intervalos com qualidade e em tempos de processamento menores (BALBONI et al., 2014).

Segundo Ratschek e Rokne (RATSCHEK; ROKNE, 1988), os computadores utilizam aritmética de ponto flutuante. Nesta aritmética números reais são aproximados por um subconjunto de números reais chamados representação numérica da máquina. Devido a esta representação são gerados dois tipos de erros: o primeiro ocorre quando uma entrada de valor real é aproximada por um número de máquina e o segundo é causado por resultados intermediários aproximados pelos números de máquina. A aritmética intervalar fornece uma ferramenta para estimar e controlar esses erros automaticamente. No lugar de aproximar um valor real x por um número de máquina, o valor real x, usualmente desconhecido, é aproximado por um intervalo $\mathbf{x}$ tendo número de máquina nos extremos inferior e superior. O intervalo $\mathbf{x}$ contém o valor x. O comprimento (ou diâmetro) deste intervalo, definido como $w(\mathbf{x}) = \bar{x} - \underline{x}$, pode ser usado como medida para qualidade da aproximação. Os cálculos são executados usando intervalos ao invés de números reais e, consequentemente, a aritmética real é substituída pela aritmética intervalar. A computação com utilização de intervalos fornece as seguintes estimativas para o erro:

- *Erro Absoluto*: $|x - m(\mathbf{x})| < \frac{w(\mathbf{x})}{2}$,

  onde $m(\mathbf{x}) = (\frac{\underline{x}+\bar{x}}{2})$ é o ponto médio do intervalo $\mathbf{x}$;

- *Erro Relativo*: $\left|\frac{x-m(\mathbf{x})}{x}\right| \leq \frac{w(\mathbf{x})}{2min|\mathbf{x}|}$ se $0 \notin \mathbf{x}$.

Observa-se que nas medidas de erros, utiliza-se o ponto médio $m(\mathbf{x})$ do intervalo $\mathbf{x}$ para medir a distância do valor real em relação ao valor pontual (ponto médio) do intervalo. Segundo Sunaga (SUNAGA, 1958) a interpretação usualmente aceita para um intervalo no contexto da aritmética intervalar é a de envoltória intervalar de um número real. Esta semântica, de envoltória intervalar, sugere a representação dos intervalos na forma $m(\mathbf{x}) \pm \frac{w(\mathbf{x})}{2}$, aludindo à idéia que o ponto médio $m(\mathbf{x})$ seria o número real "medido" e o raio $\frac{w(\mathbf{x})}{2}$ indicaria a incerteza gerada pelas restrições de precisão e ambientais existentes. Dessa forma o valor exato estaria limitado pelo intervalo apresentado.

Aplicam-se estas medidas de erros nos intervalos encapsuladores obtidos para as funções densidade de probabilidade com o objetivo de verificar a qualidade do intervalo solução, obtido após o processamento de operações aritméticas intervalares.

As funções onde a primitiva da função era simples de se obter, utilizam-se duas soluções, onde uma calcula o intervalo encapsulador através da primitiva da função, tanto na forma real quanto na intervalar, e a outra realiza o cálculo através do método 1/3 de Simpson na forma real e Simpson Intervalar definido por Caprani *et al* (CAPRANI; MADSEN; NIELSEN, 2002) para a forma intervalar. Já aquelas funções em que a primitiva não é explícita, calculou-se os intervalos encapsuladores através dos métodos de Simpson 1/3 e Simpson Intervalar. Abaixo, é listada cada função com suas respectivas formas de resolução, primeiro para entradas reais e depois para as intervalares.

- Distribuição Uniforme:

  – Real

- Primitiva da função:

$$\int_a^b f(x)dx = \frac{1}{b-a}$$

- Intervalar
  - Primitiva da função:

$$\int_a^b f(X)dx = \bar{x} \times \frac{1}{(b-a)} - \underline{x} \times \frac{1}{(b-a)}$$

- Distribuição Exponencial:
  - Real
    - Primitiva da função:

$$\alpha \int_a^b e^{-\alpha x}dx = e^{-\alpha \times x} = e^{-\alpha \times a} - e^{-\alpha \times b}$$

    - Simpson
  - Intervalar
    - Primitiva da função:

$$\alpha \int_a^b e^{-\alpha X}dx = e^{-\alpha \times X} = e^{-\alpha \times \bar{x}} - e^{-\alpha \times \underline{x}}$$

    - Simpson Intervalar
- Distribuição Normal:
  - Real
    - Simpson
  - Intervalar
    - Simpson Intervalar
- Distribuição Pareto:
  - Real
    - Primitiva da função:

$$\int_\beta^{+\infty} \frac{\alpha \beta^\alpha}{x^{\alpha+1}}dx = \beta^\alpha \times (x^{-\alpha}) = \beta^\alpha \times (-b^{-\alpha}) - \beta^\alpha \times (-a^{-\alpha})$$

    - Simpson

- Intervalar
  - Primitiva da função:

$$\int_{\beta}^{+\infty} \frac{\alpha\beta^{\alpha}}{X^{\alpha+1}} dx = \beta^{\alpha} \times (X^{-\alpha}) = \beta^{\alpha} \times (-\bar{x}^{-\alpha}) - \beta^{\alpha} \times (-\underline{x}^{-\alpha})$$

  - Simpson Intervalar

- Distribuição Gama:
  - Real
    - Simpson
  - Intervalar
    - Simpson Intervalar

As distribuições Normal e Gama não possuem uma primitiva explícita da integral, logo é preciso utilizar o método 1/3 de Simpson e Simpson Intervalar para resolução.

A seguir, apresentam-se os resultados obtidos através das implementações de cada função de densidade de probabilidade. Os exemplos utilizados nos cálculos foram retirados do trabalho desenvolvido por Varjão (VARJÃO, 2011), onde encontram-se os resultados das funções com distribuição Uniforme, Exponencial e Pareto implementadas no IntPy. Além disso, Varjão (VARJÃO, 2011) também apresenta os valores obtidos por Santos (SANTOS, 2010) em cada exemplo. O objetivo das próximas seções é apresentar os resultados obtidos pelas impelementações desenvolvidas na presente dissertação e comparar os resultados obtidos com os resultados descritos em Varjão (VARJÃO, 2011) através de uma análise da qualidade dos intervalos encapsuladores. Salienta-se que como as funções Gama e Pareto foram definidas no presente trabalho, não há comparação para estas distribuições.

Para todos os resultados foi utilizado o mesmo computador com as seguintes configurações: processador Intel Core i5 CPU 760 @ 2.80GHz Quad-Core, L1 Cache 64Kb, L2 Cache 512Kb, L3 Cache 8Mb, Memória RAM de 16GB DDR3 1333MHz, armazenamento HD Sata 500GB modelo ATA Samsung HD502HJ, sistema operacional Linux Ubuntu 13.10.

Todos os resultados intervalares e reais que serão apresentados, utilizaram o sistema de ponto flutuante F(10, 14, -10, 10).

4.1 Distribuição Uniforme

Para a Uniforme, o intervalo encapsulador foi definido a partir da função densidade, pois a derivada de quarta ordem necessária no método de Simpson Intervalar é nula (SANTOS, 2010).

Tanto para a forma real como para a forma intervalar utilizou-se apenas a forma da densidade de probabilidade desta distribuição para realizar seus cálculos.

Abaixo, a Tabela 1 apresenta os resultados obtidos com a implementação, no IntPy, da forma real e intervalar da distribuição Uniforme. Para fins de comparação, utilizou-se os mesmos exemplos e valores utilizados por Varjão (VARJÃO, 2011). Além dos intervalos encapsuladores, também é descrito o tempo, em segundos, de cada exemplo aplicado.

- Exemplo 1: Intervalo A = [0.0, 3.0], Intervalo B = [1.0, 2.0]
- Exemplo 2: Intervalo A = [0.0, 3.0], Intervalo B = [0.5, 4.0/7.0]

Tabela 1: Aplicação com a primitiva da função nas formas real e intervalar

	Valor Real	**Valor Intervalar**
Exemplo 1	0.33333333333333	[0.33333333333333, 0.33333333333334]
Tempo (s)	0.00008	0.00042
Exemplo 2	0.02380952380952	[0.02380952380952, 0.02380952380953]
Tempo (s)	0.00008	0.00039

Analisando a Tabela 1, é possível verificar que o valor real está contido no intervalo solução nos dois exemplos.

A Tabela 2 apresenta os intervalos encapsuladores calculados por Varjão (VARJÃO, 2011) e Santos (SANTOS, 2010) para a Uniforme.

Tabela 2: Resultados obtidos por Varjão e Santos

	Valor Intervalar (Varjão)	**Valor Intervalar (Santos)**
Exemplo 1	[0.33333333333331, 0.33333333333337]	[0.33333333333333, 0.33333333333334]
Tempo (s)	0.0005	0.0460
Exemplo 2	[0.02380952380952, 0.02380952380954]	[0.02380952380952, 0.02380952380953]
Tempo (s)	0.0003	0.0630

Analisando-se a Tabela 2, a qual apresenta os resultados obtidos por Varjão (VARJÃO, 2011) e por Santos (SANTOS, 2010) utilizando-se os mesmos parâmetros, verifica-se que os intervalos gerados na presente dissertação são melhores, ou seja, a solução aqui proposta retorna intervalos com menor diâmetro. Tal afirmação é comprovada na Tabela abaixo, a qual apresenta a análise da qualidade dos intervalos resultantes das soluções desenvolvidas por

Santos, Varjão e as apresentadas na presente dissertação. Quanto ao tempo que cada solução requer para gerar o resultado, a solução aqui apresentada retorna o intervalo encapsulador em menor tempo.

Os erros gerados dos intervalos encapsuladores calculados no presente trabalho encontram-se destacados na Tabela 3, bem como nas análises das qualidades dos intervalos das demais distribuições.

Tabela 3: Qualidade dos intervalos encapsuladores para a variável com distribuição Uniforme

	Diâmetro	Erro Absoluto	Erro Relativo
Finger Exemplo 1	0.00000000000001	$4.99600361081\times10^{-15} < 4.99600361081\times10^{-15}$	$1.49880108324\times10^{-14} \leq 1.49880108324\times10^{-14}$
Varjão Exemplo 1	0.00000000000006	$1.00475183729\times10^{-14} < 3.00037772405\times10^{-14}$	$3.01425551186\times10^{-14} \leq 9.00113317215\times10^{-14}$
Santos Exemplo 1	0.00000000000001	$4.99600361081\times10^{-15} < 4.99600361081e\times10^{-15}$	$1.49880108324\times10^{-14} \leq 1.49880108324\times10^{-14}$
Finger Exemplo 2	0.00000000000001	$5.00294250472\times10^{-15} < 5.00120778124\times10^{-15}$	$2.10123585198\times10^{-13} \leq 2.10050726812\times10^{-13}$
Varjão Exemplo 2	0.00000000000002	$1.00024155625\times10^{-14} < 1.0000680839\times10^{-14}$	$4.20101453624\times10^{-14} \leq 4.20028595238\times10^{-14}$
Santos Exemplo 2	0.00000000000001	$5.00294250472\times10^{-15} < 5.00120778124\times10^{-15}$	$2.10123585198\times10^{-13} \leq 2.10050726812\times10^{-13}$

Como na solução proposta na dissertação o diâmetro é nulo até a 14ª casa decimal, verifica-se que o intervalo encapsulador possui qualidade de aproximação do valor real.

No erro absoluto, verifica-se que a diferença entre o valor exato e o ponto médio do intervalo encapsulador ocorre somente na 14ª casa decimal, enquanto que os demais resultados, obtidos por Varjão e Santos, ocorrem na 13ª e 14ª casa decimal, respectivamente. Além disso, constata-se que em todos os resultados a condição $|x - m(\mathbf{x})|$ é menor que a metade do diâmetro (ou o raio) do intervalo encapsulador, satisfazendo o critério de verificação da qualidade desta estimativa para o erro.

Para o erro relativo, os resultados da solução aqui proposta apresentam diferença apenas na 15ª casa decimal, já os desenvolvidos por Varjão e Santos apresentam diferença em casas decimais anteriores a 15ª. Porém, é possível verificar que os resultados obtidos por Santos para o erro relativo são melhores, uma vez que os dois valores são bem próximos, garantindo um intervalo encapsulador próximo do valor real.

A partir da Tabela 3, é possível comprovar que o o intervalo encapsulador implementado na presente dissertação apresenta resultados melhores. Analisando-se, principalmente os valores para o diâmetro, a resolução intervalar da distribuição através da primitiva da função mostra-se com melhor qualidade que as soluções presentes em Varjão (VARJÃO, 2011).

4.2 Distribuição Exponencial

A densidade de probabilidade intervalar da distribuição Exponencial, definida por Santos (SANTOS, 2010), foi implementada no IntLab utilizando o método de Simpson Intervalar. Porém, analisando-se a integral existente na fórmula foi possível perceber que pode-se implementar a primitiva da função para obter o intervalo solução.

Dessa forma, para a Exponencial foram propostas duas implementações para a forma real e intervalar. A primeira utilizando a resolução pelo método de Simpson e Simpson Intervalar, e a segunda a partir da primitiva da função.

Apresentam-se na Tabela 4 os valores obtidos em cada uma das soluções utilizadas para a forma real e para a forma intervalar, bem como o tempo em segundos que cada solução levou para computar o cálculo. Chama-se de solução sem Simpson aquela que utiliza a primitiva da função.

- Exemplo: Intervalo A = [20, 50], α = 0.01, subintervalos = 50

Tabela 4: Aplicação com a primitiva da função e com o método de Simpson nas formas real e intervalar

	Resultado	**Tempo (s)**
Primitiva Real	0.21220009336535	0.00003
Primitiva Intervalar	[0.21220009336535, 0.21220009336536]	0.00036
Simpson Real	0.21220009336642	0.00006
Simpson Intervalar	[0.21220009336542, 0.21220009336543]	0.00238

Pelos valores apresentados na Tabela 4, nota-se que a resolução sem o método de Simpson apresentou um resultado melhor se analisarmos o tempo de processamento e o intervalo encapsulador.

A fim de comparar as soluções propostas, são apresentados na Tabela 5 os intervalos obtidos por Santos e Varjão. Já a Tabela 6 traz os resultados da análise da qualidade dos intervalos.

Tabela 5: Resultados obtidos por Varjão e Santos

	Valor Intervalar (Varjão)	**Valor Intervalar (Santos)**
Resultado	[0.21220009336534, 0.21220009336535]	[0.21220009336534, 0.21220009336535]
Tempo (s)	0.0052	1.1410

É possível verificar que a implementação realizada na dissertação retorna intervalos encapsuladores com limites confiáveis e em um tempo menor do que os obtidos por Varjão e Santos.

Tabela 6: Qualidade dos intervalos encapsuladores para a variável com distribuição Exponencial

	Diâmetro	Erro Absoluto	Erro Relativo
Finger Primitiva	0.00000000000001	$4.99600361081 \times 10^{-15} < 4.99600361081 \times 10^{-15}$	$2.35438332358 \times 10^{-14} \leq 2.35438332358 \times 10^{-14}$
Finger Simpson	0.00000000000001	$4.99600361081 \times 10^{-15} < 4.99600361081 \times 10^{-15}$	$2.35438332358 \times 10^{-14} \leq 2.35438332358 \times 10^{-14}$
Varjão	0.00000000000001	$3.45001804902 \times 10^{-13} < 5.00988139862 \times 10^{-15}$	$1.62583248401 \times 10^{-14} \leq 2.36092327726 \times 10^{-14}$
Santos	0.00000000000001	$3.45001804902 \times 10^{-13} < 5.00988139862 \times 10^{-15}$	$1.62583248401 \times 10^{-12} \leq 2.36092327726 \times 10^{-14}$

Além de retornar resultados em menor tempo, a implementação realizada no presente trabalho também retorna intervalos encapsuladores com melhor qualidade, como apresenta a Tabela 6, quando comparados aos resultados obtidos em Varjão (VARJÃO, 2011).

No cálculo do erro absoluto, verifica-se que a diferença entre o valor exato e o ponto médio do intervalo encapsulador com a solução pela primitiva da função ocorre na 14ª casa decimal, enquanto que os demais resultados, obtidos pelo método de Simpson Intervalar e por Varjão e Santos, ocorre variação na 12ª casa decimal. Já o resultado da divisão do diâmetro do intervalo pela metade, a solução aqui proposta utilizando a primitiva da função apresenta variação na 14ª casa decimal, assim como os demais resultados.

Para o erro relativo, o resultado da solução com a primitiva da função apresenta diferença apenas na 13ª casa decimal, enquanto que o método utilizando Simpson Intervalar varia já na 11ª casa. Assim, a qualidade do intervalo é garantida através das estimativas de erros calculados.

4.3 Distribuição Normal

Assim como a Exponencial, a distribuição Normal na sua forma intervalar foi implementada com o método de Simpson Intervalar por Santos (SANTOS, 2010). Como na fórmula da distribuição Normal sua primitiva da função não é explícita, não foi viável a implementação com a primitiva da função. Assim, apresentam-se nas Tabelas 7, 8, 9 e 10 os resultados das formas real e intervalar utilizando Simpson. Os exemplos utilizados nos cálculos são os mesmos aplicados por Santos (SANTOS, 2010).

- Exemplo 1: Intervalo = [0.35, 1.00], subintervalos = 50
- Exemplo 2: Intervalo = [-1.00, 1.00], subintervalos = 50
- Exemplo 3: Intervalo = [0.5555, 0.5555], subintervalos = 50
- Exemplo 4: Intervalo = [-0.5, 0.25], subintervalos = 50

Tabela 7: Aplicação com o método de Simpson nas formas real e intervalar do exemplo 1

	Resultado 1	**Tempo (s)**
Valor Real com Simpson	0.20451409489291	0.01388
Valor Intervalar com Simpson	[0.20451409489278, 0.20451409489295]	0.44832

Tabela 8: Aplicação com o método de Simpson nas formas real e intervalar do exemplo 2

	Resultado 2	**Tempo (s)**
Valor Real com Simpson	0.68268949213709	0.00167
Valor Intervalar com Simpson	[0.68268949213700, 0.68268949213712]	0.09139

Tabela 9: Aplicação com o método de Simpson nas formas real e intervalar do exemplo 3

	Resultado 3	**Tempo (s)**
Valor Real com Simpson	0.00000000000000	0.00168
Valor Intervalar com Simpson	[0.00000000000000, 0.00000000000000]	0.10106

Tabela 10: Aplicação com o método de Simpson nas formas real e intervalar do exemplo 4

	Resultado 4	**Tempo (s)**
Valor Real com Simpson	0.29016878695694	0.00183
Valor Intervalar com Simpson	[0.29016878695690, 0.29016878695695]	0.09651

As Tabelas apresentadas anteriormente comprovam que, para todos os exemplos, o resultado real obtido através do método de Simpson está contido no intervalo gerado através da aplicação do método de Simpson Intervalar.

A tabela abaixo apresenta os resultados obtidos por Santos (SANTOS, 2010).

Tabela 11: Resultados obtidos por Santos

	Valor Intervalar	**Tempo (s)**
Resultado 1	[0.20451409489283, 0.20451409489302]	16.485
Resultado 2	[0.68268949207779, 0.68268949219732]	16.969
Resultado 3	[0.00000000000000, 0.00000000000000]	16.2350
Resultado 4	[0.29016878695693, 0.29016878695695]	16.8430

Os resultados apresentados por Santos apresentam-se melhores no Exemplo 4, utilizando a mesma quantidade de subintervalos. Porém, é importante comentar que quando esse valor de subintervalos utilizado no método de Simpson é aumentado, os intervalos convergem para uma resposta melhor. Além de intervalos encapsuladores melhores nos demais exemplos, o tempo de execução também permaneceu menor.

Com a finalidade de comparar os resultados, a Tabela 12 apresenta a análise da qualidade dos intervalos encapsuladores obtidos por Santos e os obtidos na presente dissertação (em destaque).

Tabela 12: Qualidade dos intervalos encapsuladores para a variável com distribuição Normal

	Diâmetro	Erro Absoluto	Erro Relativo
Intervalar Simpson 1	0.00000000000017	$6.85562717706 \times 10^{-15} < 1.76109127281 \times 10^{-14}$	$3.35215388487 \times 10^{-14} \leq 8.61109975688 \times 10^{-14}$
Santos 1	0.00000000000019	$5.02375918643 \times 10^{-15} < 9.49934575445 \times 10^{-14}$	$2.45643665248 \times 10^{-14} \leq 4.6448367089 \times 10^{-13}$
Intervalar Simpson 2	0.00000000000012	$3.38618022511 \times 10^{-14} < 5.72875080707 \times 10^{-14}$	$4.96005909584 \times 10^{-14} \leq 8.39144424083 \times 10^{-14}$
Santos 2	0.00000000011953	$4.65072425015 \times 10^{-13} < 5.97650262613 \times 10^{-11}$	$6.81235657458 \times 10^{-13} \leq 8.75434981127 \times 10^{-11}$
Intervalar Simpson 3	0.00000000000000	$0.0 < 0.0$	$0.0 \leq 0.0$
Santos 3	0.00000000000000	$0.0 < 0.0$	$0.0 \leq 0.0$
Intervalar Simpson 4	0.00000000000005	$2.6645352591 \times 10^{-15} < 2.32869279415 \times 10^{-14}$	$9.18270806121 \times 10^{-15} \leq 8.02530423266 \times 10^{-14}$
Santos 4	0.00000000000002	$0.0 < 9.99200722163 \times 10^{-15}$	$0.0 \leq 3.44351552295 \times 10^{-14}$

De acordo com os dados apresentados na Tabela 12, é possível dizer que os resultados obtidos pela solução proposta na presente dissertação apresentam diâmetro menor do que os resultados encontrados no trabalho de Santos (SANTOS, 2010), onde no Exemplo 4 há um resultado com diâmetro maior.

Para o erro absoluto, verifica-se que a diferença entre o valor exato e o ponto médio do intervalo encapsulador, nos 4 exemplos, apresenta resultados melhores que o de Santos (SANTOS, 2010), onde a diferença nas casas decimais começa a surgir antes. O resultado da divisão do diâmetro do intervalo pela metade também mantêm a mesma característica, certificando que a solução proposta na presente dissertação apresenta qualidade boa quanto a estimativa de erro absoluto.

Quanto ao erro relativo, pode-se dizer que os resultados apresentados mantêm pouca diferença na variação das casas decimais. Chama-se atenção para o Exemplo 4, onde o diâmetro para a solução proposta no presente trabalho é maior do que o diâmetro da solução realizada pro Santos, bem como o erro relativo também apresenta melhores resultados na solução de Santos. Resultados que podem ser melhorados se aumentarmos o número de subintervalos existentes no cálculo do método de Simpson Intervalar.

4.4 Distribuição de Pareto

A implementação da distribuição de Pareto foi realizada utilizando-se a primitiva da função e o método de Simpson. Assim, apresentam-se duas maneiras para computar o valor com entradas reais e duas para computar o intervalo solução para esta distribuição com exemplos aplicados em Varjão (VARJÃO, 2011).

- Exemplo 1: Intervalo = [1.0, 2.0], α = 0.25, c = 1.0

- Exemplo 2: Intervalo = [1.0, 2.0], α = 0.50, c = 1.0
- Exemplo 3: Intervalo = [1.0, 2.0], α = 0.75, c = 1.0

Tabela 13: Aplicação com a primitiva da função e com o método de Simpson nas formas real e intervalar do exemplo 1

	Resultado 1	Tempo (s)
Primitiva Real	0.15910358474629	0.00006
Primitiva Intervalar	[0.15910358474629, 0.15910358474630]	0.00039
Simpson Real	0.15910358666866	0.00007
Simpson Intervalar	[0.15910357486653, 0.15910359486654]	0.08407

Tabela 14: Aplicação com a primitiva da função e com o método de Simpson nas formas real e intervalar do exemplo 2

	Resultado 2	Tempo (s)
Valor Real	0.29289321881345	0.00019
Valor Intervalar	[0.29289321881345, 0.29289321881346]	0.00036
Valor Real com Simpson	0.29289322438220	0.00007
Valor Intervalar com Simpson	[0.2928932188119327, 0.2928932188133538]	0.27304

Tabela 15: Aplicação com a primitiva da função e com o método de Simpson nas formas real e intervalar do exemplo 3

	Resultado 3	Tempo (s)
Valor Real	0.40539644249864	0.00020
Valor Intervalar	[0.40539644249864, 0.40539644249865]	0.00031
Valor Real com Simpson	0.40539644249707	0.02409
Valor Intervalar com Simpson	[0.40539644249858, 0.40539644249865]	0.08585

As Tabelas 13, 14 e 15 mostram que o valor real está contido no intervalo solução gerado pelas duas implementações. Além disso, é posível verificar que o tempo de processamento para computar o intervalo encapsulador utilizando o método de Simpson é maior.

Abaixo, apresentam-se os dados obtidos por Varjão e Santos para essa distribuição.

Tabela 16: Resultados obtidos por Varjão e Santos

	Valor Intervalar (Varjão)	Valor Intervalar (Santos)
Resultado 1	[0.15910358474628, 0.15910358474629]	[0.15910358474628, 0.15910358474629]
Tempo 1 (s)	0.0001	0.0310
Resultado 2	[0.29289321881345, 0.29289321881346]	[0.29289321881345, 0.29289321881346]
Tempo 2 (s)	0.001	0.0310
Resultado 3	[0.40539644249863, 0.40539644249864]	[0.40539644249863, 0.40539644249864]
Tempo 3 (s)	0.001	0.0780

Analisando os resultados presentes nas Tabelas 13, 14, 15 e 16, percebe-se que os valores obtidos com a primitiva da função são melhores se comparados com os obtidos através do método de Simpson. Ainda, é possível afirmar que os resultados aqui descritos são intervalos encapsuladores com limites melhores que os apresentados por Varjão (VARJÃO, 2011).

A partir desses dados, apresenta-se então a Tabela 17, a qual é uma análise da qualidade dos intervalos obtidos na presente dissertação e os obtidos em Varjão (VARJÃO, 2011). Salienta-se que tal análise não foi encontrada neste trabalho.

Tabela 17: Qualidade dos intervalos encapsuladores para a variável com distribuição Pareto

	Diâmetro	Erro Absoluto	Erro Relativo
Finger Primitiva 1	0.00000000000001	$4.99600361081 \times 10^{-15} < 4.99600361081 \times 10^{-15}$	$3.14009493801 \times 10^{-14} \leq 3.14009493801 \times 10^{-14}$
Finger Simpson 1	0.00000002000000	$1.80212500478 \times 10^{-9} < 1.00000049907 \times 10^{-8}$	$1.13267402861 \times 10^{-8} \leq 6.28521703496 \times 10^{-8}$
Varjão 1	0.00000000000001	$4.99600361081 \times 10^{-15} < 4.99600361081 \times 10^{-15}$	$3.14009493801 \times 10^{-14} \leq 3.14009493801 \times 10^{-14}$
Santos 1	0.00000000000001	$4.99600361081 \times 10^{-15} < 4.99600361081 \times 10^{-15}$	$3.14009493801 \times 10^{-14} \leq 3.14009493801 \times 10^{-14}$
Finger Primitiva 2	0.00000000000001	$4.99600361081 \times 10^{-15} < 4.99600361081 \times 10^{-15}$	$1.70574232857 \times 10^{-14} \leq 1.70574232857 \times 10^{-14}$
Finger Simpson 2	0.00000000014211	$1.85407245112 \times 10^{-14} < 2.8449465006 \times 10^{-14}$	$6.3301993082 \times 10^{-14} \leq 9.71325492658 \times 10^{-14}$
Varjão 2	0.00000000000001	$4.99600361081 \times 10^{-15} < 4.99600361081 \times 10^{-15}$	$1.70574232857 \times 10^{-14} \leq 1.70574232857 \times 10^{-14}$
Santos 2	0.00000000000001	$4.99600361081 \times 10^{-15} < 4.99600361081 \times 10^{-15}$	$1.70574232857 \times 10^{-14} \leq 1.70574232857 \times 10^{-14}$
Finger Primitiva 3	0.00000000000001	$4.99600361081 \times 10^{-15} < 4.99600361081 \times 10^{-15}$	$1.2323748033 \times 10^{-14} \leq 1.2323748033 \times 10^{-14}$
Finger Simpson 3	0.00000000000007	$2.38697950294 \times 10^{-14} < 3.90243393156 \times 10^{-14}$	$5.88801294908 \times 10^{-14} \leq 9.62621651908 \times 10^{-14}$
Varjão 3	0.00000000000001	$4.99600361081 \times 10^{-15} < 4.99600361081 \times 10^{-15}$	$1.2323748033 \times 10^{-14} \leq 1.2323748033 \times 10^{-14}$
Santos 3	0.00000000000001	$4.99600361081 \times 10^{-15} < 4.99600361081 \times 10^{-15}$	$1.2323748033 \times 10^{-14} \leq 1.2323748033 \times 10^{-14}$

Verifica-se, através do cálculo do diâmetro, que o método de resolução utilizando a primitiva da função resulta em intervalos com menor comprimento, em todos os exemplos apresentados.

Para o erro absoluto, verifica-se na solução com a primitiva da função que, nos exemplos 2 e 3, a diferença entre o valor exato e o ponto médio do intervalo encapsulador é nula, bem como no resultado da divisão do diâmetro do intervalo pela metade. E no exemplo 1, a estimativa do erro absoluto é bem menor que nas soluções desenvolvidas por Santos, Varjão e a solução que utiliza o método de Simpson.

Quanto ao erro relativo, pode-se dizer que a mesma característica é mantida, comprovando que os resultados dos intervalos encapsuladores utilizando a primitiva da função como solução apresentam uma qualiddade melhor quanto a estimativa de erros.

4.5 Distribuição Gama

Para implementar a distribuição Gama Intervalar, definida na presente dissertação, utilizou-se o método de Simpson Intervalar. A forma com entradas reais foi implementada também pelo método de Simpson.

- Exemplo 1: Intervalo = [20, 40], λ = 0.25, v = 5
- Exemplo 2: Intervalo = [0, 10], λ = 0.81, v = 7.81

Tabela 18: Aplicação com o método de Simpson nas formas real e intervalar do exemplo 1

	Resultado 1	**Tempo (s)**
Valor Real com Simpson	0.41124059698899	0.02758
Valor Intervalar com Simpson	[0.41124059698587, 0.41124059698825]	2.24485

Tabela 19: Aplicação com o método de Simpson nas formas real e intervalar do exemplo 2

	Resultado 2	**Tempo (s)**
Valor Real com Simpson	0.58775519331371	0.04377
Valor Intervalar com Simpson	[0.58775519331208, 0.58775519331416]	3.48640

Analisando-se as Tabelas 18 e 19, afirma-se que o valor real está contido no intervalo solução de cada exemplo. Salienta-se que o método de Simpson retorna resultados não tão bons devido ao número de subdivisões que deve ser feita. Quanto maior for esse número, melhor ele converge para uma solução melhor. Porém, o tempo de processamento acaba aumentando significativamente.

A Tabela 20 apresenta os diâmetros e erros Absoluto e Relativo dos exemplos 1 e 2.

Tabela 20: Qualidade dos intervalos encapsuladores para a variável com distribuição Gama

Exemplo	**Diâmetro**	**Erro Absoluto**	**Erro Relativo**
1	0.00000000000238	$5.06650277288 \times 10^{-13} < 1.9004242624 \times 10^{-12}$	$1.23200452727 \times 10^{-12} \leq 4.62119809263 \times 10^{-12}$
2	0.00000000000208	$9.72999458781 \times 10^{-13} < 1.04055652983 \times 10^{-12}$	$1.65545021099 \times 10^{-12} \leq 1.77039104319 \times 10^{-12}$

Importante lembrar que a extensão intervalar para a função de densidade de probabilidade da variável aleatória com distribuição Gama não foi desenvolvida na literatura anteriormente, por isso não existe uma comparação a ser feita. Analisando a Tabela 20, pode-se afirmar que os resultados apresentados são bons, onde os limites diferem no 12^o dígito.

Para o erro absoluto, verifica-se que, a diferença entre o valor exato e o ponto médio do intervalo encapsulador, bem como o resultado da divisão do diâmetro do intervalo pela metade, apresentam variação pela 11^a e 12^a casa decimal.

Para os resultados do erro relativo, a variação ocorre na 11^a casa decimal, o que garante um intervalo encapsulador com qualidade e próximo do valor real.

O presente capítulo apresentou a análise da qualidade dos intervalos encapsuladores para as distribuições. Considerou-se as estimativas de erros Absoluto e Relativo, bem como o diâmetro do intervalo. Além de realizar a análise dos intervalos encapsuladores obtidos pelas implementações propostas na presente dissertação, ainda analisou-se a qualidade dos intervalos presentes no trabalho de Varjão (VARJÃO, 2011), a fim de comparar e justificar, quanto a qualidade do intervalo, qual a melhor solução intervalar para cada distribuição. Assim, foi possível concluir que as implementações utilizando a primitiva da função sempre se mostraram de melhor qualidade em todas as estimativas. Para aquelas distribuições que não possuem solução pela primitiva da função, o método de Simpson Intervalar retornou resultados bons, se comparados com os de Varjão (VARJÃO, 2011) e Santos (SANTOS, 2010).

Com a finalidade de completar a justificativa anterior, o próximo capítulo apresenta a análise de complexidade para cada implementação apresentada na presente dissertação. Além disso, também realiza-se análise dos algoritmos com entradas reais, já que a mesma não foi encontrada na literatura.

5 ESFORÇO COMPUTACIONAL DAS FUNÇÕES DE DISTRIBUIÇÃO DE PROBABILIDADE

O termo complexidade, no contexto de algoritmos, refere-se aos requerimentos de recursos necessários para que um algoritmo possa resolver um problema sob o ponto de vista computacional, ou seja, à quantidade de trabalho despendido pelo algoritmo (TOSCANI; VELOSO, 2001). Quando o recurso é o tempo, são escolhidas uma ou mais operações fundamentais e então são contados os números de execuções desta operação fundamental na execução do algoritmo. Segundo Toscani (TOSCANI; VELOSO, 2001) a escolha de uma operação como operação fundamental é aceitável se o número de operações executadas pelo algoritmo é proporcional ao número de execuções da operação fundamental.

A complexidade também pode ser vista como uma propriedade do problema, o que significa dar uma medida independente do tratamento dado ao problema, independente do caminho percorrido na busca da solução, portanto independente de algoritmos (TOSCANI; VELOSO, 2001).

Questões relativas à complexidade de um algoritmo em termos do tempo de computação e espaço de memória são determinantes para o julgamento da eficiência do mesmo (TOSCANI; VELOSO, 2001). Um algoritmo, para ser razoável ou não, vai depender de quantos passos computacionais ele necessita para chegar a solução de um problema. Um algoritmo é considerado razoável quando obtém a solução de um problema em tempo polinomial (KREINOVICH et al., 1998).

Sob um ponto de vista computacional, Garey e Johnson (GAREY; JOHNSON, 1979) descrevem informalmente um problema como uma questão genérica a ser respondida, geralmente possuindo vários parâmetros, ou variáveis livres.

Um problema é chamado de problema computável se existir um procedimento efetivo que o resolva em um número finito de passos, ou seja, se existe um algoritmo que leve à sua solução. Observa-se, contudo, que um problema considerado ”em princípio”computável pode não ser tratável na prática, devido às limitações dos recursos computacionais para executar o algoritmo implementado (TOSCANI; VELOSO, 2001).

Se existe um algoritmo de tempo polinomial que resolve todas as instâncias de um problema, este problema é tratável, caso contrário diz-se que é intratável (TOSCANI; VELOSO, 2001).

Nas seções a seguir, apresentam-se as ordens de complexidade encontradas para cada uma das funções de densidade de probabilidade utilizadas na presente dissertação. Primeiramente, realiza-se a análise para os algoritmos com entradas reais, uma vez que não se encontrou na literatura esse tipo de análise. Em seguida, apresenta-se a análise dos algoritmos intervalares que utilizam a primitiva da função, e, posteriormente, a análise da complexidade de cada distribuição que utiliza o método de Simpson Intervalar.

5.1 Esforço Computacional dos Algoritmos com Entradas Reais

Nesta seção apresenta-se a análise feita para os algoritmos com entradas reais utilizando a primitiva da função e o método de Simpson para resolução da integral. Os algoritmos analisados podem ser encontrados no Anexo A.

5.1.1 Primitiva da função

- **Uniforme:** Como dados de entrada é preciso fornecer ao algoritmo os intervalos necessários para o cáculo da distribuição de probabilidade. A resolução da distribuição Uniforme requer apenas operações aritméticas, como subtração e divisão. A complexidade computacional de resolver tais operações é de ordem constante. Deve ser resolvida uma integral que possui primitiva na forma analítica e é simples de obter seu resultado. Como no desenvolvimento do algoritmo é necessário a implementação de operações aritméticas, a ordem de complexidade é constante, ou seja, $O(1)$.

- **Exponencial:** A distribuição Exponencial utiliza em seu algoritmo, como dados de entrada, um parâmetro α e o intervalo em que a probabilidade do valor a ser calculado deve estar contido. Após passar esses dados, o algoritmo realiza operações aritméticas para resolução da integral. Assim, a complexidade computacional do algoritmo é de ordem $O(1)$.

- **Pareto:** Para realizar o cálculo de uma distribuição de probabilidade de Pareto, é preciso fornecer como dados de entrada os parâmetros α e β, bem como o intervalo em que se quer a probabilidade. Após, é realizado o cálculo de uma integral, na qual são resolvidas operações aritméticas como soma, multiplicação, divisão e exponenciação. Assim, pode-se concluir que a complexidade computacional do algoritmo é de ordem constante, $O(1)$.

Importante lembrar que Gama e Normal não aparecem nesta subseção pois suas primitivas da integral não são explícitas, logo não são implementadas com essa solução.

5.1.2 Método 1/3 de Simpson

- **Exponencial:** A distribuição Exponencial utiliza em seu algoritmo, como dados de entrada, um parâmetro α e o intervalo em que a probabilidade do valor a ser calculado deve estar contido, além do número de subdivisões n. Após passar esses dados, o algoritmo realiza operações aritméticas e operações em um laço de repetição que deve ser executado n vezes. Assim, a complexidade computacional do algoritmo é de ordem $O(n)$.

- **Pareto:** No cálculo de uma distribuição de probabilidade de Pareto utilizando o método de Simpson, é preciso fornecer como dados de entrada os parâmetros α e β, bem como o número de subdivisões a ser utilizado no método. Após, são resolvidas operações aritméticas como soma, multiplicação, divisão e exponenciação, seguida de um laço, o qual é repetido em função do número de subdivisões. Assim, pode-se concluir que a complexidade computacional do algoritmo é de ordem linear, $O(n)$.

- **Normal:** A implementação da distribuição Normal na sua forma real utilizando o método de Simpson realiza operações de entrada de dados além do valor das subdivisões que serão utilizadas no cálculo do método. Utiliza as mesmas operações aritméticas, porém o método de Simpson para ser calculado utiliza um laço que é repetido o número de subdiviões definidos na entrada do algoritmo. Assim, sua complexidade computacional depende do número de subdivisões n, obtendo uma ordem de complexidade linear, $O(n)$.

- **Gama:** O algoritmo para calcular o valor da distribuição Gama na forma real com o método de Simpson utiliza como parâmetros de entrada os mesmos valores de λ e v, além de uma variável n a qual armaze o número de subdivisões a ser utilizada no cálculo do método. São realizadas operações aritméticas de subtração, multiplicação e exponenciação e, em seguida, existe um laço, o qual é repetido n vezes (quantas subdivisões foram definidas para n) e é executado em ordem linear. Portanto, a ordem de complexidade do algoritmo é de ordem linear, $O(n)$.

A complexidade computacional quando se utiliza o método 1/3 de Simpson não depende da entrada, pois é sempre a mesma, mas depende das subdivisões do método. A eficiência do algoritmo está relacionada com o tamanho dessas subdivisões que deve ser passada como parâmetro na execução do método, portanto a complexidade é de ordem $O(2^n)$.

Importante ressaltar que na literatura não foram encontrados resultados de complexidade computacional, ou seja, a verificação do esforço computacional requerido para computar essas funções densidade de probabilidade não foram analisados até o presente trabalho.

5.2 Esforço Computacional dos Algoritmos com Entradas Intervalares

A seguir, são apresentadas as análises de complexidade dos algoritmos desenvolvidos para as funções com entradas intervalares. Os mesmos podem ser encontrados no Anexo A do presente trabalho.

5.2.1 Primitiva da função

- **Uniforme:** O algoritmo intervalar para resolução da distribuição Uniforme utiliza as mesmas entradas do algoritmo real. Todas as operações aritméticas realizadas na forma real foram substituídas pelas operações intervalares. A complexidade de resolver tais operações é constante, assim como as operações reais. Dessa maneira, a complexidade computacional para calcular a distribuição de probabilidade da Uniforme com entradas intervalares permanece constante, $O(1)$.

- **Exponencial:** Na resolução de forma intervalar da distribuição de probabilidade da Exponencial é utilizado o parâmetro α e as entradas reais são substituídas por intervalares. Todas as operações aritméticas reais foram substituídas pelas operações aritméticas intervalares. Analisando-se a complexidade computacional, é possível notar que é a mesma encontrada para a forma real dessa distribuição. Portanto, a complexidade computacional é constante, de ordem $O(1)$.

- **Pareto:** Para realizar o cálculo da forma intervalar de uma distribuição de probabilidade de Pareto, é preciso fornecer como dados de entrada os parâmetros α e β, bem como o intervalo em que se quer a probabilidade.

Após, todas as operações aritméticas foram substituídas pelas operações da aritmética intervalar. Assim, é possível concluir que a complexidade computacional do algoritmo é de ordem constante, $O(1)$.

5.2.2 Método de Simpson Intervalar

- **Exponencial:** A forma intervalar com Simpson realiza operações aritméticas intervalares e parâmetros de entrada que são utilizados também pela forma intervalar sem Simpson. A diferença em utilizar o método é que ele precisa criar uma lista com todas as n subdivisões, o que acarreta em um laço o qual é executado n vezes. Por fim, o algoritmo apresenta mais um laço, no qual são realizados os cálculos do método. Assim, é possível afirmar que a complexidade computacional do algoritmo é de ordem linear, $O(n)$.

- **Normal:** O algoritmo inicia da mesma maneira do algoritmo sem o método de Simpson, com os parâmetros utilizados na distribuição, além do número de subdivisões as quais serão utilizadas no método. A partir daí, é criada uma lista com as n distribuições, após é realizado um laço para calcular o resultado com o método de Simpson. Portanto, a complexidade computacional do algoritmo intervalar com o método de Simpson é $O(n)$.

- **Pareto:** A realização do cálculo intervalar através do método de Simpson de uma distribuição de probabilidade de Pareto é feito com os mesmos dados de entrada sem o método, além do número de subdivisões que serão realizadas. Assim, também existem dois laços, um para criar a lista de subdivisões e o outro para realizar o cálculo do método. Afirma-se que a complexidade do algoritmo é de ordem linear, $O(n)$.

- **Gama:** O algoritmo inicia com os mesmos parâmetros de entrada da forma intervalar sem o método de Simpson. O próximo passo é um laço de repetição, o qual cria uma lista de acordo com o número de subdivisões selecionadas. A finalização do algoritmo se dá com um laço de repetição que realiza os cálculos do método de acordo com o número n de subdivisões. Portando, diz-se que a complexidade computacional do algoritmo é $O(n)$.

Assim como na complexidade do método 1/3 de Simpson, a complexidade computacional quando se utiliza o método de Simpson Intervalar também não depende mais da entrada, e sim do número de subdivisões para o método.

Portanto, a eficiência do algoritmo está relacionada com o tamanho dessas subdivisões, sendo classificado como $O(2^n)$.

A análise de complexidade de um algoritmo representa a quantidade de trabalho gasto por ele para resolver determinado problema. Na dissertação, realiza-se esse tipo de análise, uma vez que para algumas funções foi possível chegar no resultado através de duas maneiras, pela primitiva da função ou pelo método de Simpson. A Tabela 21 apresenta um quadro resumido das complexidades analisadas para todas as distribuições exploradas no presente trabalho.

Tabela 21: Esforço computacional das Distribuições

Distribuição	**Complexidade Real**		**Complexidade Intervalar**	
	Primitiva da Função	Simpson	Primitiva da Função	Simpson Intervalar
Uniforme	$O(1)$	-	$O(1)$	-
Exponencial	$O(1)$	$O(2^n)$	$O(1)$	$O(2^n)$
Normal	-	$O(2^n)$	-	$O(2^n)$
Gama	-	$O(2^n)$	-	$O(2^n)$
Pareto	$O(1)$	$O(2^n)$	$O(1)$	$O(2^n)$

Na Tabela 21 é possível verificar de forma mais clara a ordem de complexidade encontrada para cada distribuição.

A partir da análise da complexidade computacional dos algoritmos propostos para computar as funções com entradas reais, é possível afirmar que utilizando a primitiva da função como solução, os algoritmos são executados em uma complexidade menor, ou seja, menos trabalho para computar o resultado. Com as funções na forma intervalar, a mesma observação é feita, o método de Simpson Intervalar gera resultados utilizando um maior esforço computacional do que os obtidos a partir da primitiva da função. Assim, conclui-se que em ambas as formas, real e intervalar, as melhores soluções são obtidas através da aplicação da primitiva da função.

6 CONCLUSÃO

No estudo das variáveis aleatórias sobre o conjunto dos números reais, $\mathbb{R}$, um dos problemas é o cálculo de probabilidades, visto que é necessário resolver uma integral definida da função densidade que, na maioria das vezes, não possui primitiva explícita ou cuja primitiva não é simples de se obter. Embora integrais de funções densidade de probabilidade como a Uniforme, a Exponencial e a de Pareto, sejam resolvidas analiticamente, seu valor numérico no computador é dado por aproximação, e portanto afetado por erros de arredondamento ou truncamento. Outras funções densidade como a Normal ou Gama, por exemplo, não possuem primitivas na forma analítica, sendo necessário o uso de integração numérica onde erros de arredondamentos e truncamentos são propagados devido às operações aritméticas realizadas no computador.

Quando se trabalha com computação numérica, um dos fatores de maior importância é a exatidão da resposta desses cálculos. O que sempre se procura são resultados cada vez mais exatos e com um menor erro possível contido neles. A matemática intervalar surge com o objetivo principal de realizar um controle automático de erros dos cálculos, retornando respostas com a maior exatidão possível.

No presente trabalho definiu-se de forma intervalar as funções densidade de probabilidade das variáveis aleatórias com distribuições Gama e Pareto, desenvolveu-se e implementou-se resoluções das funções com distribuições Uniforme, Exponencial, Normal, Gama e Pareto. Para a variável Uniforme o intervalo encapsulador foi definido a partir da primitiva da função da função. Para as distribuições Exponencial e Pareto, utilizou-se duas implementações, uma com o método de Simpson e a outra pela primitiva da função. Já para Normal e Gama, o método utilizado foi o de Simpson.

A implementação das funções com entradas reais foi realizada na linguagem Python. Para a forma intervalar, também utilizou-se a mesma linguagem, a qual possui um pacote para programação com intervalos, chamado IntPy (INTPY, 2014). A escolha da linguagem se deu a partir de comparações realizadas entre alguns ambientes de programação intervalar (BALBONI et al., 2014).

Com o sistema de ponto flutuante F(10, 14, -10, 10) (ou com quatorze casas decimais) verificou-se, através das medidas de erros (absoluto e relativo) e diâmetro do intervalo, que todos os resultados para as probabilidades reais estão contidos nos intervalos encapsuladores. Além disso, mostrou-se resultados intervalares melhores do que os apresentados na literatura.

Verificou-se ainda se, ao utilizar intervalos para calcular a função densidade de probabilidade das variáveis aleatórias contínuas, a quantidade de trabalho dispendido pelo algoritmo aumenta em relação a forma real. Após a análise, constata-se que o esforço computacional é o mesmo, tanto na forma real quanto na intervalar. Resultado importante, o qual justifica o uso da matemática intervalar na resolução das funções. A aplicação de intervalos proporciona o controle de erros e exatidão dos resultados para estas variáveis.

O objetivo deste trabalho foi mostrar a importância e justificativa de se utilizar a matemática intervalar no cálculo de intervalos encapsuladores para as funções densidade de probabilidade das variáveis aleatórias contínuas com distribuições Uniforme, Exponencial, Normal, Gama e Pareto. A partir do desenvolvimento dos métodos e algoritmos para encontrar intervalos, baseados na matemática intervalar e na aritmética de exatidão máxima que encapsulassem probabilidades reais para as variáveis aleatórias realizou-se uma análise de complexidade de cada algoritmo para justificar seu uso.

6.1 Trabalhos Futuros

Como trabalhos futuros, pretende-se estudar algumas definições, encontradas recentemente na literatura, de integral intervalar. Acredita-se que nas funções de distribuição, como a Normal e a Gama, as quais não possuem primitiva da função na forma analítica, pode-se alcançar melhores intervalos encapsuladores e em um menor tempo de processamento aplicando-se a regra da integral com suas definições intervalares. Lembrando que o método de Simpson retorna bons resultados, porém em um tempo maior de processamento.

Muitas são as aplicações que utilizam as funções densidade de probabilidade das variáveis aleatórias com ditribuições. Sendo assim, sugere-se como complemento dessa dissertação estudar aplicações que utilizam as distribuições na forma com entradas reais e verificar os resultados obtidos aplicando-se a forma intervalar das mesmas.

As soluções propostas para o cálculo das funções intervalares devem ser inseridas no pacote estatístico intervalar que está em fase de desenvolvimento como parte do projeto de pesquisa sobre Teoria Estatística Intervalar.

REFERÊNCIAS

BALBONI, M. D. C.; TORTELLI, L. M.; LORINI, M.; FURLAN, V. S.; FINGER, A. F.; LORETO, A. B. Critérios para Análise e Escolha de Ambientes Intervalares. **Revista Jr de Iniciação Científica em Ciências Exatas e Engenharia**, Rio Grande - RS, n.7, 2014.

CAMPOS, M. A. **Uma Extensão Intervalar para a Probabilidade Real**. 1997. Tese (Doutorado em Ciência da Computação) — Universidade Federal de Pernambuco, Recife.

CAPRANI, O.; MADSEN, K.; NIELSEN, H. B. Introduction to interval analysis. **IMM - Informatics and Mathematical Modelling**, [S.l.], 2002.

DUARTE, G. C. P. **Introdução à matemática e à administração financeira intervalar**. Natal: [s.n.], 2007.

FELLER, W. **An Introduction to Probability and Its Applications**. 3th.ed. [S.l.]: John Wiley & Sons, 1968.

FERSON, S.; GINZBURG, L.; KREINOVICH, V. Absolute bounds on the mean of sum, product, etc.: A probabilistic extension of interval arithmetic. In: SIAM WORKSHOP ON VALIDATED COMPUTING, 2002, Toronto. **Anais...** [S.l.: s.n.], 2002.

FINGER, A. F.; LORETO, A. B.; CAMPOS, M. A.; VARJÃO, F.; SANTOS, M. G. A Complexidade Computacional dos Problemas de Computar Intervalos Encapsuladores para as Variáveis Aleatórias Uniforme, Exponencial e Pareto. **CLEI**, [S.l.], 2012.

GAREY, M. R.; JOHNSON, D. S. **Computers and intractability**: a guide to the theory of NP-completeness. San Francisco: Freeman, 1979.

INTPY. http://pypi.python.org/pypi/IntPy/0.1.3.

KEARFOTT, R. B. Interval computations: Introduction, uses, and resources. **Euromath Bulletin**, [S.l.], v.2, p.95–112, 1996.

KREINOVICH, V.; LAKEYEV, A.; ROHN, J.; KAHL, P. **Computational Complexity and Feasibility of Data Processing and Interval Computations**. [S.l.]: Dordrecht: Kluwer, 1998.

KULISCH, U.; MIRANKER, L. **Computer Arithmetic in Theory and Practice**. 1th.ed. [S.l.]: Academic Press, 1981.

KULISCH, U. W. Complete Interval Arithmetic and Its Implementation on the Computer. In: NUMERICAL VALIDATION IN CURRENT HARDWARE ARCHITECTURES, 2008. **Anais...** Springer, 2008. v.5492.

LAPPONI, J. C. **Estatística Usando Excel**. São Paulo: Treinamento Editora Ltda, 1995.

LORETO, A. B. **Análise da Complexidade Computacional de Problemas de Estatística Descritiva com Entradas Intervalares**. 2006. Tese (Doutorado em Ciência da Computação) — Universidade Federal do Rio Grande do Sul, Porto Alegre.

MATLAB. http://www.mathworks.com.

MEYER, P. L. **Probabilidade Aplicações à Estatística**. 2th.ed. [S.l.]: LTC, 1983.

MOORE, R. E. **Interval Analysis**. [S.l.]: Englewood Cliffs: Prentice Hall, 1966.

MOORE, R. E. **On computing the range of a rational function of variables over a bounded region**. [S.l.]: Computing, 1976. 1-15p. v.16.

MOORE, R. E. **Methods and Applications of Interval Analysis**. 2th.ed. [S.l.]: SIAM, 1979.

MOORE, R.; KEARFOTT, M.; CLOUD, J. **Introduction to Interval Analysis**. Philadelphia: SIAM, 2009.

NAGHETTINI, M.; PINTO, E. **Hidrologia Estatística**. [S.l.]: CPRM Serviço Geológico do Brasil, 2007.

OLIVEIRA, P.; DIVERIO, T.; CLAUDIO, D. **Fundamentos de Matemática Intervalar**. [S.l.]: Sagra-Luzzato, 1997.

PYTHON. http://www.python.org.

RATSCHEK, H.; ROKNE, R. **New Computer Methods for Global Optimization**. [S.l.]: Ellis Horwood, 1988.

RUGGIERO, M. A. G.; LOPES, V. L. R. **Cálculo numérico**: aspectos teóricos e computacionais. 2.ed. São Paulo: Pearson Makron Books, 1996.

SANTIAGO, R. H. N.; BEDREGAL, B. R. C.; ACIÓLY, B. M. Formal Aspects of Correctness and Optimality of Interval Computations. **Formal Aspects of Computing**, [S.l.], v.18, p.231–243, 2006.

SANTOS, M. G. **Probabilidades Autovalidáveis para as Variáveis Aleatórias Exponencial, Normal e Uniforme**. 2010. Tese (Doutorado em Ciência da Computação) — Universidade Federal de Pernambuco, Recife.

SPIEGEL, M. R. **Probabilidade e Estatística**. São Paulo: McGraw-Hill do Brasil, 1978.

STEIN, C.; DRYSDALE, R. L.; BOGART, K. **Matemática Discreta para Ciência da Computação**. [S.l.]: Pearson, 2013.

SUNAGA, T. **Theory of an Interval Algebra and its Applications to Numerical Analysis**. [S.l.]: RAAG Memoirs, 1958.

TOSCANI, L.; VELOSO, P. **Complexidade de Algoritmos**: análise, projetos e métodos. [S.l.]: Sagra-Luzzato, 2001.

VARJÃO, F. R. G. **IntPy**: Computação Científica Auto Validável em Python. 2011. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal de Pernambuco, Recife.

VILCHES, M. A. **Cálculo para Economia e Administração. Acessado em**: 15 de maio de 2013. Disponível em: http://www.ime.urj.br/$\sim calculo$.

ANEXO A IMPLEMENTAÇÕES DAS FUNÇÕES DENSIDADE DE PROBABILIDADE DAS VARIÁVEIS ALEATÓRIAS COM DISTRIBUIÇÕES UNIFORME, EXPONENCIAL, NORMAL, GAMA E PARETO NO INTPY

Todas as implementações descritas a seguir foram realizadas pelo Maurício Balboni, bolsista de iniciação científica no projeto de pesquisa TEIa (Teoria Estatística Intervalar), o qual encontra-se em andamento.

A.1 Distribuição Uniforme

```
from intpy import *
from intpy.support.stdfunc import sqrt
import math
from decimal import *
def UniformeR(a,b,c,d):
        return d/(b-a)-c/(b-a)
def UniformeI(a,b,c,d):
        a=IReal(a)
        b=IReal(b)
        c=IReal(c)
        d=IReal(d)
        return d/(b-a)-c/(b-a)
```

A.2 Distribuição Exponencial

```
from scipy import integrate
from scipy import inf
from math import exp, expm1
from intpy import *
from intpy.support.stdfunc import sqrt
import math
from decimal import *
import time
def powI(dado, exp):
    result = IReal(dado.inf**exp.inf, dado.sup**exp.sup)
    return result

def ExponencialR(alpha,a,b):
   e=2.71828182845904523536028747135266249775724709369995
   return ((e**(-alpha*a))-e**(-alpha*b))
def ExponencialI(alpha,a,b):
        e=IReal(2.71828182845904523536028747135266249775724709369995)
        alpha=IReal(alpha)
        a=IReal(a)
        b=IReal(b)
        y=(powI(e,-alpha*a))
        z=(powI(e,-alpha*b))
        return y-z
def ExponencialRS(alp,a,b):
        total=0
        n=1000
        tamanho=(a-b)/float(n)
        total=total+(alp*(exp(-a*alp)))
        total=total+(alp*(exp(-b*alp)))
        aux=b
        for i in range(n-1):
                if i%2 == 0:
                        aux=aux+tamanho
                        total=total+4*(alp*(exp(-aux*alp)))
                else:
                        aux=aux+tamanho
                        total=total+2*(alp*(exp(-aux*alp)))
                total=(tamanho/3.0)*total
         return -total
def ExponencialIS(alp,a,b):
        listaInt = []
        e=2.71828182846
        totalI=IReal(0)
        total=0
        n=10000
        x = IReal(a,b)
        h = (x.sup-x.inf)/float(n)
        aux=x.inf
        for i in range(n):
                listaInt += [ IReal(aux, aux+h) ]
                aux=aux+h
        for i in range(n):
                totalaux=IReal(0)
                totalaux=totalaux+(alp*e**(-listaInt[i].inf*alp))
                totalaux=totalaux+(alp*e**(-listaInt[i].sup*alp))
                aux=(listaInt[i].inf+listaInt[i].sup)/2
                totalaux=totalaux+(4*alp*e**(-aux*alp))
                totalaux=totalaux*((listaInt[i].sup-listaInt[i].inf)/6)
                total=totalaux+total
                aux=total
        return aux
```

A.3 Distribuição Normal

```
from scipy import integrate
from scipy import inf
from math import exp, expm1
import math
from decimal import *
from math import sqrt
from intpy import *
def NormalR(alpha,mi,a,b):
    pi=3.14159265358979323846264338327950288419716939937510
    eu=2.71828182845904523536028747135266249775724709369995
    f=lambda x,a: (1/(float(alpha)*sqrt(2*pi)))*eu**(-(x-float(mi))**2/(2*float(alpha)**2))
    y, err = integrate.quad(f, a, b, args=(1,))
    return y
def NormalI(alpha,mi,a,b):
        pi=3.14159265358979323846264338327950288419716939937510
        eu=2.71828182845904523536028747135266249775724709369995
        f=lambda x,a: eu**(-(x-mi)**2/(2*alpha**2)) #coloca a expressao numa variavel
        y, err = integrate.quad(f, a, b, args=(1,)) #realiza a integral
        y=IReal(y) #VAriavel intervalar
        alpha=IReal(alpha) #Variavel intervalar
        aux=IReal(sqrt(2*pi)) #Variavel intervalar
        aux2=IReal(1.0) # Variavel Intervalar
        aux=y*(aux2/(alpha*aux)) # Resultado final
        return aux
def NormalRS(alpha,mi,a,b):
        n=10000
        pi=3.14159265358979323846264338327950288419716939937510
        eu=2.71828182845904523536028747135266249775724709369995
        total=0
        tamanho=(a-b)/float(n)
        total=total+((1/(alpha*sqrt(2*pi)))*eu**(-(a-mi)**2/(2*alpha**2)))
        total=total+((1/(alpha*sqrt(2*pi)))*eu**(-(b-mi)**2/(2*alpha**2)))
        aux=b
        for i in range(n-1):
                if i%2 == 0:
                   aux=aux+tamanho
                   total=total+4*((1/(alpha*sqrt(2*pi)))*eu**(-(aux-mi)**2/(2*alpha**2)))
                else:
                    aux=aux+tamanho
                    total=total+(2*((1/(alpha*sqrt(2*pi)))*eu**(-(aux-mi)**2/(2*alpha**2))))
        total=(tamanho/3.0)*-total
        return total
def NormalIS(alpha,mi,a,b):
        listaInt = []
        n=100000
        e=2.71828182846
        pi=3.14159265358979323846264338327950288419716939937510
        total=0
        x = IReal(a,b)
        h = (x.sup-x.inf)/float(n)
        aux=x.inf
        for i in range(n):
                listaInt += [ IReal(aux, aux+h) ]
                aux=aux+h
        for i in range(n):
                totalaux=IReal(0)
                totalaux=totalaux+((1/(alpha*sqrt(2*pi)))*e**(-((listaInt[i].inf-mi)**2)/(2*alpha**2)))
                totalaux=totalaux+((1/(alpha*sqrt(2*pi)))*e**(-(listaInt[i].sup-mi)**2/(2*alpha**2)))
                aux=(listaInt[i].inf+listaInt[i].sup)/2
                totalaux=totalaux+(4*((1/(alpha*sqrt(2*pi)))*e**(-(aux-mi)**2/(2*alpha**2))))
                totalaux=totalaux*((listaInt[i].sup-listaInt[i].inf)/6)
                total=totalaux+total
        return total
```

A.4 Distribuição Gama

```
from scipy import integrate
from scipy import inf
from math import exp, expm1
from intpy import *
from intpy.support.stdfunc import sqrt
import math
from decimal import *
def GamaR(lamb,v,a,b):
        f=lambda x,a: exp(-lamb*x)*x**(v-1)
        y, err = integrate.quad(f, 0, inf, args=(1,))
        y=y**-1
        z, err = integrate.quad(f, a, b, args=(1,))
        return z*y
def GamaI(lamb,v,a,b):
        f=lambda x,a: exp(-lamb*x)*x**(v-1)
        y, err = integrate.quad(f, 0, inf, args=(1,))
        y=y**-1
        z, err = integrate.quad(f, a, b, args=(1,))
        y=IReal(y)
        z=IReal(z)
        return z*y
def GamaRS(L,v,a,b):
        n=50000
        total=0
        tamanho=(a-b)/float(n)
        f=lambda x,a: exp(-L*x)*x**(v-1)
        y, err = integrate.quad(f, 0, inf, args=(1,))
        y=y**-1
        total=total+(exp(-L*a)*a**(v-1))
        total=total+((exp(-L*b)*b**(v-1)))
        aux=b
        for i in range(n-1):
                if i%2 == 0:
                        aux=aux+tamanho
                        total=total+4*((exp(-L*aux)*aux**(v-1)))
                else:
                        aux=aux+tamanho
        total=total+2*((exp(-L*aux)*aux**(v-1)))
        total=(tamanho/3.0)*total
        return -(total*y)
def GamaIS(L,v,a,b):
       listaInt = []
       n=50000
       eI=IReal(2.71828182846)
       e=2.71828182846
       total=0
       x = IReal(a,b)
       h = (x.sup-x.inf)/float(n)
       aux=x.inf
       f=lambda x,a: exp(-L*x)*x**(v-1)
       y, err = integrate.quad(f, 0, inf, args=(1,))
       y=y**-1
       for i in range(n):
             listaInt += [ IReal(aux, aux+h) ]
             aux=aux+h
       for i in range(n):
             totalaux=IReal(0)
             totalaux=totalaux+(e**(-L*listaInt[i].inf)*listaInt[i].inf**(v-1))
             totalaux=totalaux+(e**(-L*listaInt[i].sup)*listaInt[i].sup**(v-1))
             aux=(listaInt[i].inf+listaInt[i].sup)/2
             totalaux=totalaux+4*(e**(-L*aux)*aux**(v-1))
             totalaux=totalaux*((listaInt[i].sup-listaInt[i].inf)/6)
             total=totalaux+total
        return total*y
```

A.5 Distribuição de Pareto

```
from scipy import integrate
from scipy import inf
from math import exp, expm1
from intpy import *
from intpy.support.stdfunc import sqrt
import math
from decimal import *
from PowI import *
def ParetoR(alpha,beta,a,b):
        return beta**alpha*(-b**-alpha)-beta**alpha*(-a**-alpha)
def ParetoI(alpha,beta,a,b):
        alpha=IReal(alpha)
        beta=IReal(beta)
        a=IReal(a)
        b=IReal(b)
        y1=powI(beta,alpha)
        y2=-(powI(b,-alpha))
        y3=-(powI(a,-alpha))
        return y1*y2-y1*y3
def ParetoRS(alpha,beta,a,b):
        n=10000
        total=0
        tamanho=(a-b)/float(n)
        total=total+((alpha*beta**alpha)/(a**(alpha+1)))
        totalaux=total+((alpha*beta**alpha)/(a**(alpha+1)))
        total=total+((alpha*beta**alpha)/(b**(alpha+1)))
        aux=b
        for i in range(n-1):
        if i%2 == 0:
                   aux=aux+tamanho
                   total=total+4*((alpha*beta**alpha)/(aux**(alpha+1)))
        else:
                   aux=aux+tamanho
                   total=total+2*((alpha*beta**alpha)/(aux**(alpha+1)))
                   total=(tamanho/3.0)*total
                   return -total
def ParetoIS(alpha,beta,a,b):
        listaInt = []
        n=50000
        e=2.71828182846
        totalI=IReal(0)
        total=0
        x = IReal(a,b)
        h = (x.sup-x.inf)/float(n)
        aux=x.inf
        for i in range(n):
                listaInt += [ IReal(aux, aux+h) ]
                aux=aux+h
        for i in range(n):
                totalaux=IReal(0)
                totalaux=totalaux+((alpha*beta**alpha)/(listaInt[i].inf**(alpha+1)))
                totalaux=totalaux+((alpha*beta**alpha)/(listaInt[i].sup**(alpha+1)))
                aux=(listaInt[i].inf+listaInt[i].sup)/2
                totalaux=totalaux+4*((alpha*beta**alpha)/(aux**(alpha+1)))
                totalaux=totalaux*((listaInt[i].sup-listaInt[i].inf)/6)
                total=totalaux+total
         return total
```