

TESE DE DOUTORADO

**Unscented Transform Framework
for Quantization Modeling
in Data Conversion Systems**

José Edil Guimarães de Medeiros

Brasília, Novembro de 2017

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

UNSCENTED TRANSFORM FRAMEWORK FOR QUANTIZATION
MODELING IN DATA CONVERSION SYSTEMS

JOSÉ EDIL GUIMARÃES DE MEDEIROS

TESE DE DOUTORADO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA
FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS
REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR.


APROVADA POR:




SANDRO AUGUSTO PAVLIK HADDAD, Dr., ENE/UNB
(ORIENTADOR)



CRISTIANO JACQUES MIOSSO, Dr., FGA/UNB
(EXAMINADOR EXTERNO)



JOSÉ CAMARGO DA COSTA, Dr., ENE/UNB
(EXAMINADOR INTERNO)



RAIMUNDO CARLOS SILVÉRIO FREIRE, Dr., UFCG
(EXAMINADOR EXTERNO)

Brasília, 10 de novembro de 2017.

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

TESE DE DOUTORADO

**Unscented Transform Framework
for Quantization Modeling
in Data Conversion Systems**

**Modelagem do Processo de Quantização
em Conversores de Dados Baseada
na Transformada da Incerteza**

José Edil Guimarães de Medeiros

*Tese de Doutorado submetida ao Departamento de Engenharia Elétrica
da Faculdade de Tecnologia da Universidade de Brasília como parte dos
requisitos necessários para a obtenção do grau de Doutor.*

Aprovada por

Prof. Sandro A. P. Haddad, ENE/UnB
Orientador

Prof. José Camargo da Costa, ENE/UnB
Examinador Interno

Prof. Raimundo Carlos Silvério Freire, UFCG
Examinador Externo

Prof. Cristiano J. M. R. Mendes, FGA/UnB
Examinador Externo

FICHA CATALOGRÁFICA

Medeiros, José Edil Guimarães de

Unscented Transform Framework for Quantization Modeling in Data Conversion Systems [Distrito Federal] 2017.

viii, 108p., 210 x 297 mm (ENE/FT/UnB, Doutor, Tese de Doutorado – Universidade de Brasília. Faculdade de Tecnologia.

Departamento de Engenharia Elétrica

1. Unscented Transform

2. Data Converters

3. Quantization

4. Design

I. ENE/FT/UnB

II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

De Medeiros, J. E. G. (2017). Unscented Transform Framework for Quantization Modeling in Data Conversion Systems. Tese de Doutorado em Engenharia Elétrica, Publicação PGEA 124/2017, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 108p.

CESSÃO DE DIREITOS

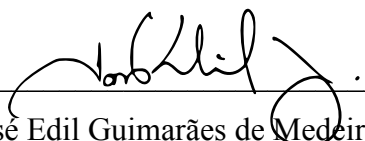
AUTOR: José Edil Guimarães de Medeiros

TÍTULO: Unscented Transform Framework for Quantization Modeling in Data Conversion Systems.

GRAU: Doutor

ANO: 2017

É concedida à Universidade de Brasília permissão para reproduzir cópias desta tese de doutorado e para emprestar ou vender tais cópias somente para propósitos acadêmicos e científicos. O autor reserva outros direitos de publicação e nenhuma parte dessa tese de doutorado pode ser reproduzida sem autorização por escrito do autor.



José Edil Guimarães de Medeiros
SHTN Trecho 1 Conj. 2 Bloco F Apart. 405
70800-200 Brasília – DF – Brasil

“I’m convinced that the only thing that kept me going was that I loved what I did.”

Steve Jobs

Agradecimentos

Impossível não citar minha família, em especial meus pais José Edil Benedito e Maria Celeste Guimarães, por suportarem com seu trabalho árduo meus estudos até o nível superior.

Agradeço ao meu orientador, Prof. Sandro A. P. Haddad, por ter me proporcionado ampla liberdade para pesquisar sobre qualquer tema que eu julgava pertinente, pelas inúmeras discussões exaltadas sem hora para acabar e por assumir o risco de orientar um doutorando que era também seu colega de trabalho.

Sou grato ao Prof. Leonardo R. A. X de Menezes por insistir para que eu assistisse a seu curso sobre a Transformada da Incerteza mesmo após eu ter concluído formalmente os requisitos de créditos para o prosseguimento do Doutorado. Na época não poderíamos prever que aquela seria uma semente tão próspera.

Obrigado aos colegas que trabalharam comigo na Suécia durante o desenvolvimento deste trabalho, em especial ao Prof. Ingo Sander que aceitou receber um estrangeiro vindo do Brasil em seu grupo de pesquisa e dedicar a este parte do seu tempo, algo de extrema valia perante tantas responsabilidades, ao amigo Prof. Gilmar S. Beserra que me apresentou o grupo de pesquisa da KTH e trouxe a possibilidade de realizarmos um projeto conjunto e ao amigo George Ungureanu que naquele país conheci e que juntos estabelecemos uma próspera (e espero que duradoura) cooperação de pesquisa.

A Fernanda P. M. F. Reis que pelos últimos dois anos tem sido um porto seguro.

Muitas outras pessoas contribuíram de forma direta ou indireta para o meu trabalho de Doutorado que culmina com a publicação desta tese, seria impossível agradecer nominalmente a cada uma dessas pessoas.

José Edil Guimarães de Medeiros

ABSTRACT

This thesis presents a framework for the design of signal specific quantizers based on the Unscented Transform — UT — for the design of data converters. We formally define the UT in terms of the interpolatory quadrature and we choose the Gaussian quadrature as the optimal scheme for maximizing the order of the transformation. We present an efficient method for computing the UT. The UT is presented as an alternative to Monte Carlo methods in which we introduce an Extended UT for the probability function estimation problem. We show how to abstract a time signal into a probability function and use the UT to design signal specific quantizers.

RESUMO

Esta tese apresenta uma abordagem para o projeto de quantizadores para sinais específicos baseada na Transformada da Incerteza — Unscented Transform (UT) — visando o projeto de conversores de dados. É apresentada uma definição formal da UT em termos da quadratura interpolatória, é demonstrada que a quadratura Gaussiana representa a escolha ótima para maximizar a ordem da transformada e é apresentado um algoritmo para o cálculo eficiente da UT. A UT é apresentada como uma alternativa a métodos de Monte Carlo e é introduzida a Transformada da Incerteza Extendida no contexto do problema de estimação de funções de probabilidade. É apresentado um método para abstrair sinais definidos no tempo em funções de probabilidade e como utilizar a UT para o projeto de quantizadores para sinais específicos.

INDEX

1	INTRODUCTION	1
1.1	THE SIGNAL PROCESSING CHAIN	1
1.2	THE UNSCENTED TRANSFORM	3
1.3	RESEARCH OBJECTIVES AND SCOPE	5
1.4	OUTLINE OF THE THESIS	5
2	THE UNSCENTED TRANSFORM	7
2.1	PROBABILITY THEORY DEFINITIONS	7
2.1.1	RANDOM VARIABLES AND PROBABILITY FUNCTIONS	10
2.1.2	STATISTICAL MOMENTS	11
2.2	THE UNSCENTED TRANSFORM	12
2.2.1	DIRECT METHOD FOR COMPUTING THE UT	14
2.3	THE UT AS A MECHANICAL QUADRATURE PROBLEM	17
2.3.1	THE INTERPOLATORY QUADRATURE	19
2.3.2	GAUSSIAN QUADRATURE	21
2.3.3	EXAMPLE	26
2.4	CHAPTER FINAL REMARKS	28
3	PROBABILITY FUNCTIONS ESTIMATION	29
3.1	THE UT AS AN ALTERNATIVE TO MONTE CARLO	29
3.2	YIELD ESTIMATION CASE STUDY	30
3.3	THE EXTENDED UNSCENTED TRANSFORM	36
3.4	CHAPTER FINAL REMARKS	39
4	QUANTIZER DESIGN WITH THE UT	40
4.1	SIGNAL MODELING	40
4.2	QUANTIZER DESIGN	42
4.3	THE ANALYSIS OF THE ARCSINE QUANTIZER	43
4.4	CIRCUIT IMPLEMENTATION PROPOSAL	48
4.5	CHAPTER FINAL REMARKS	51
5	RELATED WORK	52
5.1	FILTERING	52
5.2	PROBABILITY FUNCTIONS ESTIMATION	53

5.3	QUANTIZATION	54
6	CONCLUSION	55
6.1	FUTURE RESEARCH	56
6.1.1	PUBLICATIONS	57
	REFERENCES	57
	ANEXOS	64
I	RESUMO ESTENDIDO	65
I.1	INTRODUÇÃO	65
I.1.1	OBJETIVOS E ESCOPO	66
I.2	A TRANSFORMADA DA INCERTEZA	66
I.2.1	THE UT AS A MECHANICAL QUADRATURE PROBLEM	66
I.2.2	A QUADRATURA INTERPOLATÓRIA	67
I.2.3	A QUADRATURA GAUSSIANA	68
I.3	ESTIMAÇÃO DE FUNÇÕES DE PROBABILIDADE	70
I.3.1	A UT ESTENDIDA	71
I.4	PROJETO DE QUANTIZADORES BASEADO NA UT	75
I.4.1	PROJETO DO QUANTIZADOR	76
I.5	CONCLUSÕES	79
II	UNSCENTED TRANSFORM IMPLEMENTATION	81
II.1	RECURRENCE RELATIONS	81
II.2	GAUSSIAN QUADRATURE	83
II.3	UNSCENTED TRANSFORM	84
II.4	THE EXTENDED UT	86
II.5	ARCSINE QUANTIZER	99

FIGURES INDEX

1.1	Typical signal processing chain.....	2
1.2	Signal Processing domains and definitions.....	2
1.3	Pictorial view of the Unscented Transform.....	4
1.4	This thesis relates the numerical quadrature, the interpolatory quadrature and the Unscented Transform as a single discipline. It also provides a step towards expanding the quantization discipline by relating it to the UT theory.....	4
2.1	The UT maps a continuous probability functions from the set on the left into a discrete probability function on the set on the right. Figure 2.2 shows three possible mappings from $p_{x1}(x)$, a Gaussian probability function, into three different sets of sigma-points.....	14
2.2	Different 2-th order UT sigma points for $\mu = 0$ and $\sigma = 2$ Gaussian distribution.....	15
2.3	Interpolation showing the $\ell_i(x)$ polynomial basis and the Lagrange interpolation polynomial.....	21
3.1	Ring oscillator.....	30
3.2	Monte Carlo and UT yield estimation application flowcharts.....	32
3.3	Monte Carlo experiments with increasing number of simulations to show the convergence of the method.....	33
3.4	The UT estimates of the density function with 3 sigma points per random variable...	34
3.5	Distribution function estimates with analytical, Monte Carlo and UT approaches.....	34
3.6	Relative error for the moments estimates given by the UT.....	35
3.7	Relative error for a typical Monte Carlo run with increasing number of points.....	35
3.8	Absolute error for the probability estimation using the UT.....	36
3.9	Different UT formulations.....	36
3.10	A pictorial representation of the Extended UT process.....	38
3.11	ExUT probability estimation.....	39
3.12	Absolute error for the probability estimation using the ExUT.....	39
4.1	Unscented Transform and the moment preserving quantizer.....	41
4.2	Characteristic curve for a 4-level arcsine distribution quantizer.....	42
4.3	Characteristic curves for the linear and arcsine quantizers.....	45
4.4	Transient analysis for a single tone input signal.....	45
4.5	Spectral analysis for a single tone input.....	47

4.6	Quantization error spectral analysis for a single tone input.	48
4.7	Spectral analysis for a single tone input on 10-bit quantizers.	49
4.8	SNR and SINAD for increasing quantizers resolution.	49
4.9	THD and SFDR for increasing quantizers resolution.	50
4.10	ADC and DAC block diagrams showing the influence of the thresholds th_n and output points S_n given by Eq (4.8).	50
4.11	Schematic ADC and DAC circuit topologies based on parallel architectures for the implementation of the moment preserving quantizers.	51
5.1	The BTC coding scheme is based on moment preserving quantizers.	54
I.1	Typical signal processing chain.	65
I.2	Oscilador em anel utilizado no estudo de caso. Os atrasos d_n de cada inversor são modelados por uma variável aleatória Gaussiana com média $d_0 = 1\mu s$ e variância $\Delta d = 0.02\mu s$	70
I.3	Fluxogramas comparando o método de estimação de funções de probabilidade com Monte Carlo e com a Transformada da Incerteza.	72
I.4	Erro relativo da estimação dos momentos pelo método da UT.	73
I.5	Erro relativo para uma execução típica do método de Monte Carlo com aumento no número de execuções comparado com o resultado do método da UT.	73
I.6	Different UT formulations.	74
I.7	Representação gráfica da Transformada da Incerteza Estendida.	75
I.8	A Transformada da Incerteza e o quantizador que preserva momentos.	75
I.9	Curva característica de um quantizador de 4 bits para a distribuição arco-seno.	76
I.10	Curvas características dos quantizadores linear e da distribuição arco-seno.	78
I.11	Análise transiente para um sinal de entrada senoidal.	78
I.12	SNR e SINAD para quantizadores de diferentes resoluções.	79
I.13	THD e SFDR para quantizadores de diferentes resoluções.	79

TABLES INDEX

2.1	Orthogonal polynomials associated with classical distributions.....	23
-----	---	----

SYMBOLS INDEX

\mathbb{R}	The set of all Real numbers
\mathbb{Z}	The set of all Integer numbers
p_i	Probability of an event
$P_x(x)$	Probability distribution function
$p_x(x)$	Probability density function
m_n	n -th order moment
$E\{.\}$	Expected value operator
s_i	UT sigma-points
w_i	UT weights
S_n	Interval partition with n points
$Q_n(f)$	Numerical quadrature of the function f
x_i	Numerical quadrature abscissas
λ_i	Numerical quadrature weights
π_n	Set of all polynomials of degree at most n
$\ell_i(x)$	Lagrange polynomial
$L(x)$	Lagrange interpolant
$p_n(x)$	n -th order orthogonal polynomial
th_n	Quantizer threshold levels

Acronyms

ADC	Analog-to-Digital Converter
DAC	Digital-to-Analog Converter
UT	Unscented Transform
ExUT	Extended Unscented Transform
EKF	Extended Kalman Filter
UKF	Unscented Kalman Filter
MC	Monte Carlo
AGC	Automatic Gain Control
PGA	Programable Gain Amplifier
PUF	Physical Unclonable Function
SRAM	Static Random Access Memory
RV	Random Variable
THD	Total Harmonic Distortion
SNR	Signal-to-Noise Ratio
SINAD	Signal-to-Noise and Distortion Ratio
ENOB	Effective Number of Bits
SFDR	Spurious Free Dynamic Range
FFT	Fast Fourier Transform
RUL	Remaining Useful Life
BTC	Block Truncation Coding
MPQ	Moment Preserving Quantizer
CMOS	Complementary Metal-Oxide-Semiconductor

Chapter 1

Introduction

This chapter presents the context in which this thesis is developed. It introduces the problem of preserving the statistical moments of an input signal under quantization. Our approach involves the use of the Unscented Transform to solve such a problem. It follows with the research objectives and scope of the thesis and the contributions of the thesis.

1.1 The Signal Processing Chain

Data conversion is everywhere around us. In all forms of electronic equipment, analog-to-digital and digital-to-analog conversion link our physical world to digital computing machines. Pure analog electronics can be used to implement classical signal processing needs, such as audio amplification, filtering and radio frequency interfaces, in a cheap and well-established way. In more complex situations, digital signal processing offers advantageous extensions of this functionality: improved signal-to-noise ratio which leads to better storage of digitized signals, the option to carry complex calculations that are not obvious on the analog domain and the easiness to adapt the algorithms to changing circumstances [1]. To allow an application to benefit from these advantages, analog signals must be converted to a digital format, usually in an early stage of the processing chain. At the end of the digital processing, it is desirable for some applications to carry out the conversion in the reverse direction. Figure 1.1 shows a typical architecture for a digital signal processing application in which an input analog signal is conditioned by means of analog amplifiers and filters, gets converted to digital format by an analog-to-digital converter — ADC — and is processed in the digital domain by some kind of digital signal processor — DSP. After the processing phase, the signal is converted back to the analog domain by a digital-to-analog converter — DAC — and this output signal is reconditioned.

In engineering, we usually model a signal by means of a mathematical function mapping numbers from one domain to another. Figure 1.2 show the four possible processing domains for time varying signals. In this work, we define an analog signal to be a function f_{analog} mapping from a continuous domain into a continuous range, $f_{\text{analog}} : \mathbb{R} \rightarrow \mathbb{R}$. Throughout this work we will use the notation $y = f(t)$ to denote a continuous signal y by means of the function f applied on a

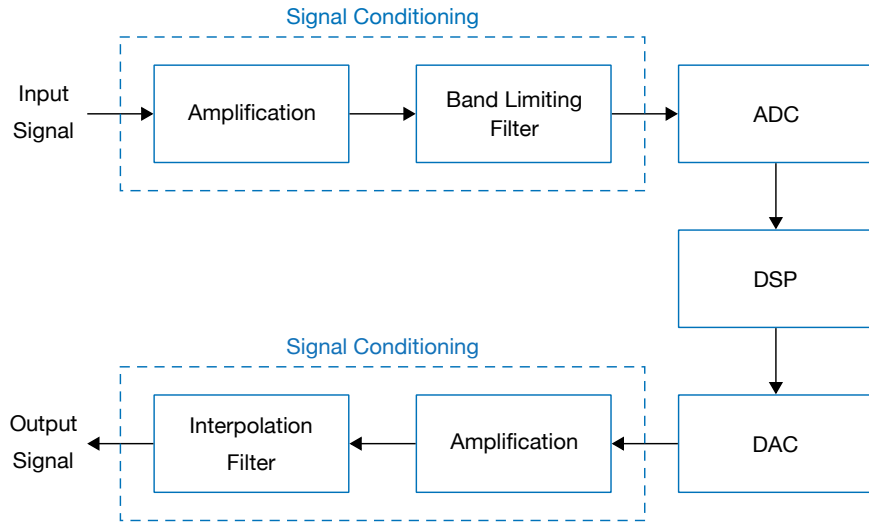


Figure 1.1: Typical signal processing chain.

continuous valued variable t . A digital signal, on the other hand, will be modeled by a function $f_{digital}$ mapping a discrete domain into a discrete range, $f_{digital} : \mathbb{Z} \rightarrow \mathbb{Z}$. We will use the notation $y[n] = f_{digital}(n \cdot T_s)$, where T_s denotes a sampling period.

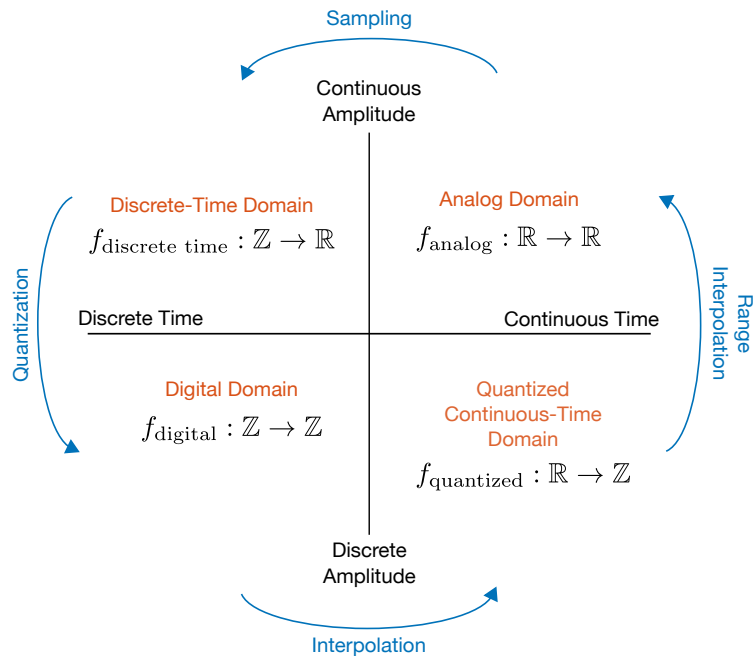


Figure 1.2: Signal Processing domains and definitions.

Starting in the analog domain, we call sampling the process of discretization of the signal domain to get a discrete-time domain signal, $f_{discrete\ time} : \mathbb{Z} \rightarrow \mathbb{R}$. The inverse process will be called interpolation. The discrete-time domain is the model we use to represent sampled systems such as switched-capacitor circuits. We call quantization the process of discretization of the signal range to get a quantized continuous-time signal, $f_{quantized} : \mathbb{R} \rightarrow \mathbb{Z}$. The inverse process will be called range interpolation. Quantized discrete-time signals, although not so common, are used to model continuous-time digital signal processing applications [2]. Sampling and quantization can be

seen as independent processes and both should be implemented by an analog-to-digital converter.

Analog-to-Digital conversion involves sampling and quantization. Since the works by Nyquist [3] and Shannon [4], designers have established a sampling theory to understand and implement data converters causing little to no distortion. Quantization, on the other hand, is an error-inducing process and is the major cause of distortion, limiting the theoretical performance of data converters.

Several criteria for quantizers design have been proposed, each trying to optimize performance by using different design goals. Max proposes minimizing distortion which he defines as the expected value of some function of the error between the input and the output of the quantizer [5]. This is an interesting approach as it considers the statistical behavior of the input signal on the design of the quantizer [6]. Lloyd extends Max's idea and proposes optimizing the quantizer for the mean squared error [7]. In general, such quantizers yield nonuniform quantizers, i.e., quantizers in which quantization level thresholds are not equally spaced.

Statistical-based quantization are standard techniques for signal coding in the digital domain. However the majority of analog-to-digital and digital-to-analog converters implement uniform quantization [8]. In some specific applications the use of non-linear quantization yields better performance [9]. Additionally, uniform quantization alters the statistical structure of the input signal [10, 11, 12]. This thesis proposes to optimize quantizers for data converter applications to preserve the statistical moments of an input signal. We approach the problem by means of the Unscented Transform. Such a class of quantizers would benefit applications sensitive to higher-order statistics such as matched filters for signal detection [13], modulation identification [14], identification of non-linear systems [15] and non-Gaussian signal processing [16].

1.2 The Unscented Transform

The Unscented Transform – UT –, as we will show in the next chapters, is a mathematical framework that models a continuous probability function into a discrete one proposed by Uhlmann to overcome the linearization issues that arrive in the implementation of the Extended Kalman Filters [17]. In his doctoral dissertation [18], Uhlmann explains:

Consider the following intuition: With a fixed number of parameters it should be easier to approximate a given distribution than it is to approximate an arbitrary nonlinear function/transformation. Following this intuition, the goal is to find a parameterization that captures the mean and covariance information while at the same time permitting the direct propagation of the information through an arbitrary set of nonlinear equations. This can be accomplished by generating a discrete distribution having the same first and second (and possibly higher) moments, where each point in the discrete approximation can be directly transformed. The mean and covariance of the transformed ensemble can then be computed as the estimate of the nonlinear transformation of the original distribution. More generally, the application of a given nonlinear transfor-

mation to a discrete distribution of points, computed so as to capture a set of known statistics of an unknown distribution, is referred to as an unscented transformation.

Figure 1.3 illustrates the basic idea behind the UT that will be formalized and detailed in Chapter 2. The UT is a mapping between a continuous random variable — RV — input characterized by its probability density function and a discrete random variable output characterized by its probability mass function. This mapping is chosen such that the α first statistical moments of the output RV are equal to those of the input distribution. On the discrete output, the location where the probabilities concentrate are called the sigma-points and their probabilities will be called weights.

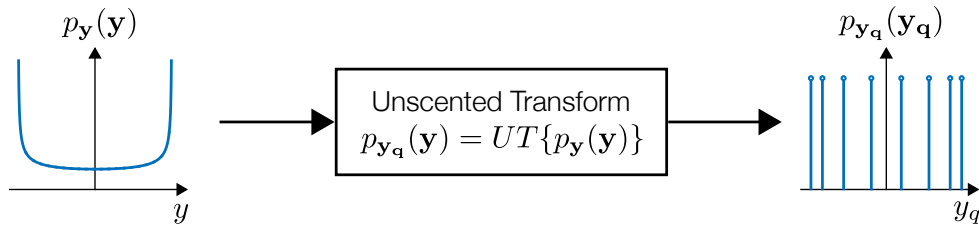


Figure 1.3: Pictorial view of the Unscented Transform.

The Unscented Transform has been successfully used in applications where nonlinear transformations are the main concern for the system analysis or design. In Chapter 5 we present a brief survey showing different applications for the Unscented Transform. As we show, the two main applications of the UT are on the Unscented Kalman Filter and for the estimation of probability functions. In this work we propose a novel application, the use of the UT in the design of moment preserving quantizers.

Our goal is to first present a formal definition for the UT based on the interpolatory quadrature theory, a particular numerical quadrature technique. Later, we present our quantizer design methodology to link the numerical quadrature, the unscented transform and the quantization disciplines as shown in Figure 1.4.

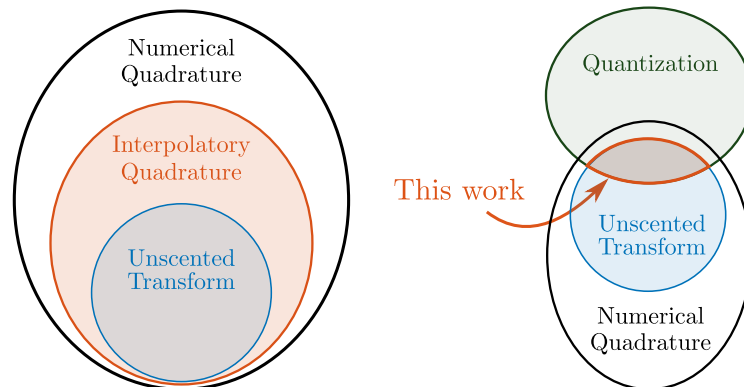


Figure 1.4: This thesis relates the numerical quadrature, the interpolatory quadrature and the Unscented Transform as a single discipline. It also provides a step towards expanding the quantization discipline by relating it to the UT theory.

1.3 Research Objectives and Scope

The main objective of this work is to propose the use of the Unscented Transform as a framework for the design of quantizers for data conversion applications. In this way, we are setting the optimization goal of quantizer design to be the preservation of the statistical moments of the input signal and thus we are trying to carry structural information from the analog signal into the digital domain.

To achieve that goal, we propose the following specific objectives:

- propose a formal definition for the Unscented Transform based on the numerical quadrature discipline;
- compare the use of the UT with Monte Carlo methods for probability functions estimation;
- propose an Extended Unscented Transform formulation to improve the number of sigma-points in some of the UT applications;
- propose a method to model deterministic time signals as random variables suitable for the analysis with the UT;
- propose a design flow for the design of quantizers based on the UT deriving its design equations;
- analyze a case study comparing the figures of merit of the linear quantizer and the of the arcsine distribution quantizer.

1.4 Outline of the thesis

To achieve the research objectives, this thesis provides a formal definition for the Unscented Transform and a formal argument to use Gaussian Quadrature methods to compute the required sigma-points and weights. It follows with the idea of understanding the UT as a Gaussian quadrature problem. The definitions and theorems presented in Chapter 2 bond together the numerical quadrature, the interpolatory quadrature and the Unscented Transform disciplines. It provides the relevant theorems to support our MATLAB implementation. Proofs are provided when they are based on constructive approaches.

It follows by presenting how the UT can be used to substitute Monte Carlo techniques and present an extended formulation for the sigma-points computation. Chapter 3 details the most immediate use of the Unscented Transform, to serve as an alternative to Monte Carlo methods. It starts by presenting a case study and follows with the development of the Extended Unscented Transform, a method to improve the probability functions estimates.

Chapter 4 presents the main contribution of this thesis, the derivation of a design flow for signal specific nonlinear quantizers that preserve statistical moments from an analog input signal when converted to the digital domain. We show how to design a class of moment preserving

quantizers and present the analysis of the arcsine quantizer as a case study in which we get distortion improvements for the quantizer.

The thesis follows with an overview of related works to help the reader position the current work compared to the state-of-the-art, presented in Chapter 5. To this date, we could not find any published paper with explicit mention to the use of the Unscented Transform on the design of data converters or quantizers.

Chapter 6 presents the conclusions and some open research problems we could identify during the development of this thesis. Since this thesis is intended to serve as a reference to future research efforts, we include in Appendix I a translation to Portuguese of the main relevant material included in the thesis and in Appendix II we included all Matlab/Octave source code necessary to reproduce the results reported on this thesis.

Chapter 2

The Unscented Transform

In this chapter we establish the formal grounds that will be extensively used in this thesis. We first introduce the basic probability theory concepts that we refer throughout this thesis. We follow by providing our definition for the Unscented Transform in a way that will lead to straightforward definitions for the nonlinear quantizers studied in Chapter 4. We follow by reviewing the classical theory of numerical quadrature by means of orthogonal polynomials and how they relate to the UT.

2.1 Probability Theory Definitions

We start by defining the terminology that will be used in the probability theory discussion.

Definition 1 A *set* is a collection of objects called elements. A subset \mathcal{B} of a set \mathcal{A} is another set whose elements are also elements of \mathcal{A} . All sets under consideration will be subsets of a set \mathcal{S} which we shall call *space*. The *empty* set contains no elements.

In general, the elements of a set are arbitrary objects. In this work we will mostly deal with sets of numbers. The elements of a set will be identified by the Greek letter ζ . Thus

$$\mathcal{A} = \{\zeta_1, \zeta_2, \dots, \zeta_n\}$$

will mean that the set \mathcal{A} is composed by the elements $\zeta_1, \zeta_2, \dots, \zeta_n$. The notation

$$\zeta_i \in \mathcal{A}, \quad \zeta_i \notin \mathcal{A}$$

will mean that ζ_i is or is not an element of \mathcal{A} , respectively. We denote the empty set by \emptyset .

The following set operations will be useful for the definitions that follow. For more details, refer to [19].

Definition 2 The *union* or *sum* of two sets \mathcal{A} and \mathcal{B} is a set whose elements are all elements of \mathcal{A} or of \mathcal{B} or of both. This set will be referred as $\mathcal{A} \cup \mathcal{B}$.

Definition 3 The *intersection* or *product* of two sets \mathcal{A} and \mathcal{B} is a set whose elements are all elements that are common to both the sets \mathcal{A} and \mathcal{B} . This set will be referred as $\mathcal{A} \cap \mathcal{B}$.

The following definitions establish some useful concepts.

Definition 4 The space \mathcal{S} is a set called *certain event*, its elements *experimental outcomes*, and its subsets *events*. The empty set \emptyset is the *impossible event*, and the event $\{\zeta_i\}$ consisting of a single element ζ_i is an elementary event.

Definition 5 A single performance of an experiment will be called a *trial*. At each trial we observe a single outcome ζ_i . We say that an event \mathcal{A} occurs during this trial if it contains the element ζ_i . The certain event occurs at every trial and the impossible event never occurs.

Finally, the following definition establishes the probability axioms. These axioms are chosen in a way that the resulting theory gives a satisfactory representation of the physical world.

Definition 6 The *probability of the events* \mathcal{A} and \mathcal{B} will be numbers $P(\mathcal{A})$ and $P(\mathcal{B})$ respectively assigned to them. Those numbers are arbitrarily chosen to satisfy the following *probability axioms*.

$$P(\mathcal{A}) \geq 0 \quad \text{and} \quad P(\mathcal{B}) \geq 0 \tag{2.1a}$$

$$P(\mathcal{S}) = 1 \tag{2.1b}$$

$$\text{If } \mathcal{A} \cap \mathcal{B} = \emptyset \quad \text{then} \quad P(\mathcal{A} \cup \mathcal{B}) = P(\mathcal{A}) + P(\mathcal{B}). \tag{2.1c}$$

Events are subsets of \mathcal{S} to which we assign probabilities. It is convenient, however, to restrict the subsets of \mathcal{S} that we consider to be events. The main reason is the difficulty that arises when considering cases with infinitely many outcomes, in which the assignment of probabilities satisfying the axioms (2.1a)–(2.1c) to all subsets may become impossible [19]. This leads us to the following definitions.

Definition 7 A *field* \mathcal{F} is a nonempty class of sets such that

$$\text{If } \mathcal{A} \in \mathcal{F} \quad \text{then} \quad \overline{\mathcal{A}} \in \mathcal{F}. \tag{2.2}$$

$$\text{If } \mathcal{A} \in \mathcal{F} \quad \text{and} \quad \mathcal{B} \in \mathcal{F} \quad \text{then} \quad \mathcal{A} + \mathcal{B} \in \mathcal{F}. \tag{2.3}$$

Definition 8 Suppose that $\mathcal{A}_1, \dots, \mathcal{A}_n, \dots$ is an infinite sequence of sets in \mathcal{F} . If the union and intersection of these sets also belong to \mathcal{F} , the \mathcal{F} is called a *Borel field* [20].

We define events to which we assign probabilities as certain subsets of \mathcal{S} forming a Borel field. This permits us to assign probabilities not only to finite unions and intersections, but also to their

limits [20]. Finally, we extend axiom (2.1c) to include infinitely many sets: if $\mathcal{A}_1 \cap \mathcal{A}_2 \cap \dots = \emptyset$ then

$$P(\mathcal{A}_1 \cup \mathcal{A}_2 \cup \dots) = P(\mathcal{A}_1) + P(\mathcal{A}_2) + \dots, \quad (2.4)$$

This additional condition is known as the *axiom of infinite addition*.

Finally, we can define an experiment in terms of the set \mathcal{S} of all experimental outcomes, the Borel field of all events of \mathcal{S} and the probabilities of these events. For our discussion, we only care with experiments defined over countable spaces and over the real line.

Definition 9 If the space \mathcal{S} consists of N outcomes and N is a finite number, then the probabilities of all events can be expressed in terms of the probabilities

$$P\{\zeta_i\} = p_i \quad (2.5)$$

of the elementary events $\{\zeta_i\}$ [19].

From the axioms it follows that

$$p_i \geq 0. \quad (2.6)$$

$$p_1 + \dots + p_N = 1. \quad (2.7)$$

If \mathcal{A} is an event consisting of the m elements ζ_{k_m} , it follows from (2.1c) that

$$P(\mathcal{A}) = P\{\zeta_{k_1}\} + \dots + P\{\zeta_{k_m}\} = p_{k_1} + \dots + p_{k_m}. \quad (2.8)$$

The above holds true, by application of axiom (2.4), even if \mathcal{S} consists of an infinite but countable number of elements [20].

Definition 10 Suppose that \mathcal{S} is the set of all real numbers. To construct a probability space on the real line, we shall consider as events all intervals $x_1 \leq x \leq x_2$ and their countable unions and intersections. These events form the smallest Borel field that includes all half-lines $x \leq x_i$ where x_i is any number. This field contains all open and closed intervals, all points, and every set of points on the real line that is of interest in most applications.

The numbers not covered by this definition form a negligible set in the sense they can be covered by intervals the total sum of whose lengths can be made arbitrarily small and do not require the assignment of probabilities [20]. To complete the specification of \mathcal{S} , it suffices to assign probabilities to the events $\{x \leq x_i\}$.

Definition 11 Suppose that $w(x)$ is a function such that

$$\int_{-\infty}^{\infty} w(x) dx = 1 \quad \text{and} \quad w(x) \geq 0. \quad (2.9)$$

We define the probability of the event $\{x \leq x_i\}$ by the integral

$$P\{x \leq x_i\} = \int_{-\infty}^{x_i} w(x) dx. \quad (2.10)$$

2.1.1 Random Variables and Probability Functions

We follow by defining a random variable.

Definition 12 A *random variable* \mathbf{x} — RV — is an arbitrary function that assigns a number $\mathbf{x}(\zeta)$ to every outcome ζ of an experiment satisfying two conditions:

$$\text{The set } \{\mathbf{x} < x\} \text{ is an event for every } x. \quad (2.11a)$$

$$\text{The probabilities of the events } \{\mathbf{x} = -\infty\} \text{ and } \{\mathbf{x} = +\infty\} \text{ equal } 0. \quad (2.11b)$$

In the following discussions we will note the random variables in boldface letters to avoid any confusion with other notation.

A function $f(t)$ is a rule of correspondence between objects of two sets. The objects of the independent variable t form a set \mathcal{S}_t called the *domain* of the function and the objects of the dependent variable x form a set \mathcal{S}_x called the *range* of the function. The rule of correspondence between these two sets can be a table, a curve, or a formula. The objects in \mathcal{S}_t and \mathcal{S}_x can be anything. Random variables are a class of functions that guarantees that we get a numeric range from arbitrary domain events.

Definition 13 The *distribution function* (or *cumulative distribution function* or even *probability function*) of the RV \mathbf{x} is the function

$$F_{\mathbf{x}}(x) = P\{\mathbf{x} \leq x\} \quad (2.12)$$

defined for every x in the range $(-\infty, +\infty)$.

Based on Definition 13 we define the terms continuous distribution, discrete distribution and density function.

Definition 14 We shall say that an RV \mathbf{x} is *continuous* if it's distribution function is continuous, that is, $F_{\mathbf{x}}(x^-) = F_{\mathbf{x}}(x^+)$ for all x in $(-\infty, +\infty)$ and hence

$$P\{\mathbf{x} = x\} = 0. \quad (2.13)$$

Definition 15 We shall say that an RV \mathbf{x} is *discrete* if there exists at least one point x_i of discontinuity such that $F_{\mathbf{x}}(x_i^-) \neq F_{\mathbf{x}}(x_i^+)$. $F_{\mathbf{x}}(x)$ will then present a staircase pattern as the one shown in Figure 3.9c and it follows that

$$F_{\mathbf{x}}(x_i^+) - F_{\mathbf{x}}(x_i^-) = P\{\mathbf{x} = x_i\} = p_i. \quad (2.14)$$

and the statistics of \mathbf{x} are determined in terms of x_i and p_i [19].

Definition 16 The *density function* (or the *frequency function*) of the RV \mathbf{x} is given by the derivative

$$f_{\mathbf{x}(x)} = \frac{dF_{\mathbf{x}}(x)}{dx} \quad (2.15)$$

of $F_{\mathbf{x}}(x)$.

In other sections of this work we also use the notation $P_{\mathbf{x}}(x) = F_{\mathbf{x}}(x)$ for the distribution function and $p_{\mathbf{x}}(x) = f_{\mathbf{x}}(x)$ for the density function where it may cause ambiguity with other notations.

The following *existence theorem* is useful for dealing with RVs without worrying about any specific physical experiment and to derive a model for deterministic signals in Chapter 4.

Theorem 2.1.1 Let $f(x)$ be a nonnegative function with unity area in the interval $(-\infty, \infty)$. Let its integral

$$F(x) = \int_{-\infty}^x f(\tau) d\tau$$

be continuous from the right and monotonically increasing from 0 to 1 as x increases from $-\infty$ to ∞ . Then, given $f(x)$ or $F(x)$ we can construct an RV \mathbf{x} with distribution function $F(x)$ and density function $f(x)$.

Proof First, consider the space \mathcal{S} as the set of all real numbers, and as its events all intervals on the real line and their unions and intersections. We define the probability of the event $\{x \leq x_1\}$ by

$$P\{x \leq x_1\} = F(x_1) \quad (2.16)$$

where $F(x)$ is the given function. According to Eq. (2.10) and the associated definition, this completely specifies the experiment.

The outcomes of the experiment are the real numbers so we can simply define the RV \mathbf{x} such that x is the outcome of the experiment and the corresponding value of the random variable, that is

$$\mathbf{x}(x) = x. \quad (2.17)$$

In this way, the event $\{x \leq x_1\}$ consists of all outcomes x such that $\mathbf{x}(x) \leq x_1$. Hence

$$P\{\mathbf{x} \leq x_1\} = P\{x \leq x_1\} = F(x_1). \quad (2.18)$$

Since this is true for every x_1 , the theorem is proved [19]. \square

2.1.2 Statistical Moments

We follow by defining the expected value operator.

Definition 17 The *expected value* $E\{x\}$ of a continuous RV \mathbf{x} is defined by

$$E\{x\} = \int_{-\infty}^{\infty} x f_{\mathbf{x}}(x) dx \quad (2.19)$$

where $f_{\mathbf{x}}(x)$ is the probability density function of \mathbf{x} .

Definition 18 The *expected value* $E\{x\}$ of a discrete RV \mathbf{x} is defined by

$$E\{x\} = \sum_{i=1}^k x_i p_i \quad (2.20)$$

where p_i is the probability of the event x_i and k denotes the number of discontinuity points in Definition 15..

Corollary 2.1.2 *The expected value operator is a linear operator, i.e.,*

$$E\{ax + by\} = aE\{x\} + bE\{y\} \quad (2.21)$$

in which a, b are arbitrary constants and x, y are arbitrary random variables.

We end this section by defining the moments of an RV.

Definition 19 We call the quantities

$$m_n = E\{x^n\} \quad (2.22)$$

for $n = 1, 2, \dots$ the *n -th order moments* of the RV \mathbf{x} .

For a continuous RV \mathbf{x} , the moments become

$$m_n = \int_{-\infty}^{\infty} x^n f_{\mathbf{x}}(x) dx \quad (2.23)$$

and for the discrete case

$$m_n = \sum_{i=1}^k x_i^n p_i \quad (2.24)$$

2.2 The Unscented Transform

Based on the principle that *it is easier to approximate a probability function than it is to approximate an arbitrary nonlinear function or transformation*, Julier and Uhlmann [17] propose the Unscented Transform – UT – to be a discrete set of sigma points, $\{S_i, w_i\}$, that can be used to capture the influences of nonlinear mappings on the statistical properties of a random variable.

We consider here the same approach of Menezes et al. [21] to justify the UT. For this, let \mathbf{x} be a continuous random variable with probability density function $p_{\mathbf{x}}(x)$ and $\mathbf{y} = g(\mathbf{x})$ the RV

built by a smooth nonlinear function $g(x)$ applied over \mathbf{x} . Taking the Maclaurin series of \mathbf{y} with arbitrary number of terms k we can write \mathbf{y} in the polynomial form

$$y = \sum_{n=0}^k \frac{g^{(n)}(0)}{n!} x^n = \sum_{n=0}^k a_n x^n = g(x), \quad (2.25)$$

where $g^{(n)}(0)$ denotes the n -th derivative of $g(x)$ evaluated at $x = 0$ and $a_n = \frac{g^{(n)}(0)}{n!}$. By applying the expected value operator over Eq. (2.25) we get:

$$E\{y\} = \sum_{n=0}^k a_n E\{x^n\} = E\{g(x)\}. \quad (2.26)$$

Analyzing Eq. (2.26), Menezes et al. [21] show that the knowledge of the moments of the input continuous distribution \mathbf{x} is sufficient to capture the nonlinear mapping behavior of $\mathbf{y} = g(\mathbf{x})$. The more moments we know, the better the estimate. This also holds true for higher order moments of \mathbf{y} . In this way we define the Unscented Transform.

Definition 20 The α -th order *Unscented Transform* of a continuous RV \mathbf{x} with probability density function $p_x(\mathbf{x})$ is a set of n sigma-points and weights pairs, $\{s_i, w_i\}$, characterizing a discrete probability distribution $F_x(\mathbf{X})$ in the sense of Definition 15 such that

$$E\{\mathbf{X}^k\} = \sum_{i=1}^n s_i^k w_i = \int_{-\infty}^{+\infty} x^k p_x(x) dx = E\{\mathbf{x}^k\}, \quad (2.27)$$

for $k = 1, 2, \dots, \alpha$, that is, both the discrete and the continuous distributions have the same moments up to the α -th order.

From a specific set of sigma points one can define the Inverse Unscented Transform problem.

Definition 21 The α -order *Inverse Unscented Transform* of a discrete RV \mathbf{X} defined by a set of n sigma-points and weights pairs, $\{s_i, w_i\}$, is a continuous RV \mathbf{x} with probability density function $p_x(\mathbf{x})$ such that Eq. (2.27) holds for $k = 1, 2, \dots, \alpha$.

Figure 2.1 shows that the UT and the inverse UT are not unique, i.e., from a given continuous RV one can find many sets of sigma-points $\{s_i, w_i\}$ which satisfy Definition 20 and from a given set of sigma-points one can find many continuous probability distributions which satisfy Definition 21. This is because we defined the UT to agree with a continuous RV for all moments up to m_α which causes some structural information about the probability distribution present in the higher order moments to be lost in the transformation. Properties and methods for the inverse UT problem, also known in the literature as the *problem of moments* [22, 23], are out of the scope of this work.

Fig 2.2 show three different sets of 2-th order UT sigma points for $\mu = 0$ and $\sigma = 2$ Gaussian distribution, i.e., three different discrete probability distributions in which $m_1 = 0$ and $m_2 = 2$. These sets are built in an ad-hoc way and they pose interesting questions about the UT that will be addressed on the following sections:

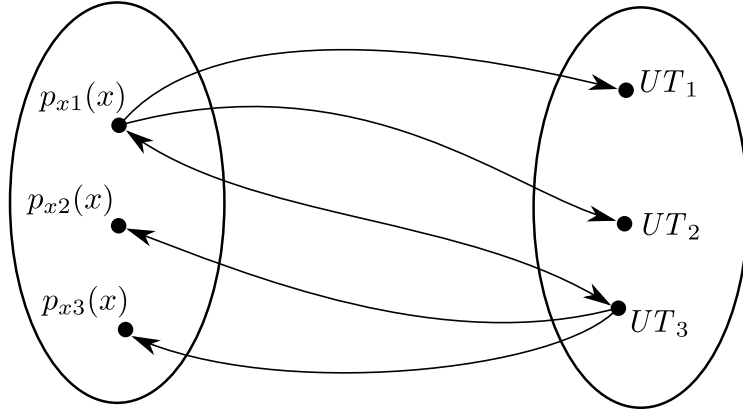


Figure 2.1: The UT maps a continuous probability functions from the set on the left into a discrete probability function on the set on the right. Figure 2.2 shows three possible mappings from $p_{x1}(x)$, a Gaussian probability function, into three different sets of sigma-points.

- what is the minimum number of sigma-points required to preserve the α first moments of a given distribution?
- what are systematic methods to compute minimum sets of sigma-points s_i ?
- how to compute the associated weights w_i ?

On next section we introduce one possible algorithm to systematically compute the UT sigma points.

2.2.1 Direct Method for Computing the UT

Based on the work of Tabatabai et al. [24] and of Da Costa Junior [25] who independently showed the following algorithm, the solutions for Eq. (2.27) can be found by following a three step approach if the moments of $p_x(x)$ are known to exist.

Algorithm 2.2.1 *Direct method for the Unscented Transform calculation.*

First, solve the system described by Eq. (2.28) to find a set of k coefficients α_i :

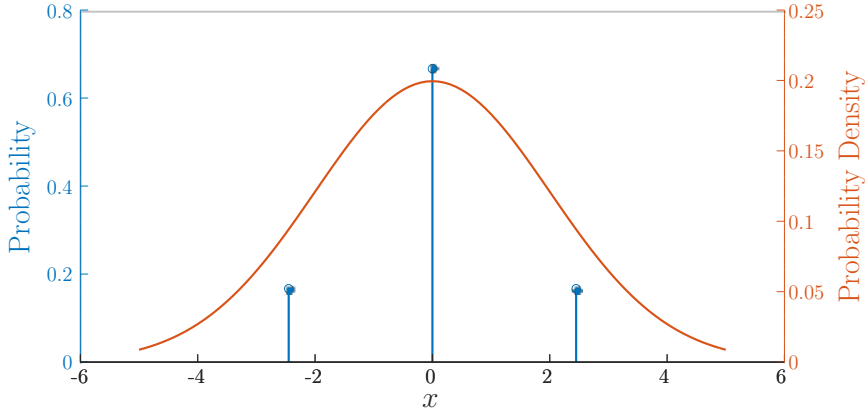
$$\sum_{i=0}^k \alpha_i m_{N+k-1} = 0 \quad (2.28)$$

for $N = 0, 1, \dots, k-1$ and $\alpha_0 = 1$ where m_r is the r -th order moment of $p(\mathbf{x})$.

Second, define a k -order polynomial π_k :

$$\pi_k(x) = \sum_{i=0}^k \alpha_i x^{k-i}. \quad (2.29)$$

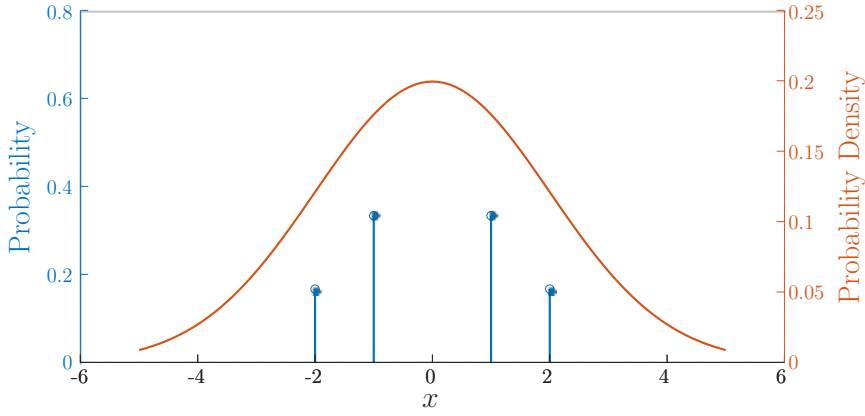
The sigma points s_i will be the roots of this polynomial, i.e., $\pi_k(s_i) = 0$.



(a) Set 1.

Set 1	
s_i	w_i
-2.4495	0.16667
0.0	0.66667
2.4495	0.16667

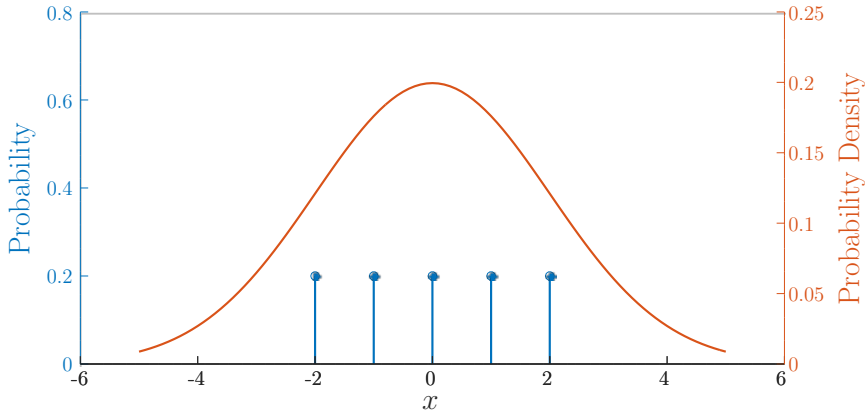
(b) Set 1 numerical values.



(c) Set 2.

Set 2	
s_i	w_i
-2.0	1/6
-1.0	2/6
1.0	2/6
2.0	1/6

(d) Set 2 numerical values.



(e) Set 3.

Set 3	
s_i	w_i
-2.0	0.2
-1.0	0.2
0.0	0.2
1.0	0.2
2.0	0.2

(f) Set 3 numerical values.

Figure 2.2: Different 2-th order UT sigma points for $\mu = 0$ and $\sigma = 2$ Gaussian distribution.

Third, find the probabilities w_i for each sigma point by solving the linear system of equations:

$$\sum_{i=1}^k w_i (s_i)^r = m_r \quad (2.30)$$

for all $r = 0, 1, \dots, k - 1$.

To clarify this process, let's detail the calculation of the UT with 3 sigma points for a given density function $p_x(x)$ with known moments m_0 up to m_5 . We want to find the set $\{(s_1, w_1), (s_2, w_2), (s_3, w_3)\}$ with 6 unknown variables. For this, we first expand Eq. (2.27) to the necessary set of equations:

$$w_1 + w_2 + w_3 = m_0 \quad (2.31a)$$

$$s_1 w_1 + s_2 w_2 + s_3 w_3 = m_1 \quad (2.31b)$$

$$s_1^2 w_1 + s_2^2 w_2 + s_3^2 w_3 = m_2 \quad (2.31c)$$

$$s_1^3 w_1 + s_2^3 w_2 + s_3^3 w_3 = m_3 \quad (2.31d)$$

$$s_1^4 w_1 + s_2^4 w_2 + s_3^4 w_3 = m_4 \quad (2.31e)$$

$$s_1^5 w_1 + s_2^5 w_2 + s_3^5 w_3 = m_5 \quad (2.31f)$$

Next, we construct the polynomial $\pi_3(x)$ by selecting its zeros to be the desired sigma points.

$$\pi_3(x) = (x - s_1)(x - s_2)(x - s_3) = x^3 + \alpha_1 x^2 + \alpha_2 x + \alpha_3. \quad (2.32)$$

We now multiply Eq. (2.31a) by α_3 , Eq. (2.31b) by α_2 , Eq. (2.31c) by α_1 to get the following set of equations.

$$\alpha_3 w_1 + \alpha_3 w_2 + \alpha_3 w_3 = \alpha_3 m_0$$

$$\alpha_2 s_1 w_1 + \alpha_2 s_2 w_2 + \alpha_2 s_3 w_3 = \alpha_2 m_1$$

$$\alpha_1 s_1^2 w_1 + \alpha_1 s_2^2 w_2 + \alpha_1 s_3^2 w_3 = \alpha_1 m_2$$

$$s_1^3 w_1 + s_2^3 w_2 + s_3^3 w_3 = m_3$$

By summing up the four previous equations we get

$$\begin{aligned} (s_1^3 + \alpha_1 s_1^2 + \alpha_2 s_1 + \alpha_3) w_1 + (s_2^3 + \alpha_1 s_2^2 + \alpha_2 s_2 + \alpha_3) w_2 + (s_3^3 + \alpha_1 s_3^2 + \alpha_2 s_3 + \alpha_3) w_3 = \\ \pi_3(s_1) w_1 + \pi_3(s_2) w_2 + \pi_3(s_3) w_3 = \\ m_3 + \alpha_1 m_2 + \alpha_2 m_1 + \alpha_3 m_0 = 0 \end{aligned}$$

since s_1 , s_2 and s_3 are by construction the zeros of the polynomial $\pi_3(x)$. We repeat this process with the set of equations (2.31b)–(2.31e) and then with equations (2.31c)–(2.31f) to get the following system of equations described by Eq. (2.28)

$$m_3 + \alpha_1 m_2 + \alpha_2 m_1 + \alpha_3 m_0 = 0$$

$$m_4 + \alpha_1 m_3 + \alpha_2 m_2 + \alpha_3 m_1 = 0$$

$$m_5 + \alpha_1 m_4 + \alpha_2 m_3 + \alpha_3 m_2 = 0.$$

In matrix form:

$$\begin{bmatrix} m_2 & m_1 & m_0 \\ m_3 & m_2 & m_1 \\ m_4 & m_3 & m_2 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \begin{bmatrix} -m_3 \\ -m_4 \\ -m_5 \end{bmatrix} \quad (2.33)$$

We solve this system of equations to get the coefficients α_i for the polynomial $\pi_3(x)$. With this information, we can find the roots of this polynomial and get the s_i points.

The last step is to get the the necessary set of equations from (2.31a)–(2.31f) and solve for w_i

$$\begin{aligned} w_1 + w_2 + w_3 &= m_0 \\ s_1 w_1 + s_2 w_2 + s_3 w_3 &= m_1 \\ s_1^2 w_1 + s_2^2 w_2 + s_3^2 w_3 &= m_2 \end{aligned}$$

or in matrix form

$$\begin{bmatrix} 1 & 1 & 1 \\ s_1 & s_2 & s_3 \\ s_1^2 & s_2^2 & s_3^2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} m_0 \\ m_1 \\ m_2 \end{bmatrix} \quad (2.34)$$

The direct method to calculate the UT presents the advantage of being applicable even if we don't know *a priori* the probability function $p_x(x)$ we are interested to model. We rely only on the knowledge of its moments and thus is useful in engineering applications where we can estimate the statistical moments from empirical data. However, although sufficient for our discussion so far and for the application of the UT in several applications, this method presents some numerical instability issues due to the nature of the first matrix in Eq. (2.33). In fact, Gautschi shows that solving Eq. (2.33) is as ill-conditioned as the inversion of Hilbert matrices, a problem known to be numerically hard to solve [26]. We observe this fact empirically when trying to employ Algorithm 2.2.1 to solve for a large number of sigma points and it would later limit our ability to design high resolution quantizers. It is necessary to find a better way to find the UT sigma-points and weights.

2.3 The UT as a Mechanical Quadrature Problem

We begin with the classical definition of the mechanical quadrature by Szego [27].

Definition 22 Let $[a, b]$ be a finite or infinite interval, and let

$$S_n : x_1 < x_2 < \cdots < x_n, \quad (2.35)$$

for $a \leq x_1$ and $x_n \leq b$, denote a set of n distinct points in $[a, b]$. We call S_n a *partition* of $[a, b]$.

Furthermore, let

$$\Lambda_n : \lambda_1, \lambda_2, \dots, \lambda_n \quad (2.36)$$

be a set of real numbers. Let $f(x)$ be an arbitrary function defined in the interval $[a, b]$. Then, we call

$$Q_n(f) = \sum_{i=1}^n \lambda_i f(x_i) \quad (2.37)$$

the *mechanical quadrature* of f , the numbers x_i the *abscissas* of the quadrature and the numbers λ_i the *Cotes numbers* associated with the quadrature.

The mechanical quadrature is an arbitrary process that maps sums of weighted samples of a function $f(x)$ into numbers. Of special interest for this work is the special case for the mechanical quadrature in which the Cotes numbers λ_n are defined such that

$$Q_n(f) = \sum_{i=1}^n \lambda_i f(x_i) = \int_a^b f(x) du(x) \quad (2.38)$$

holds if $f(x)$ is an arbitrary polynomial π_{n-1} of order up to $n - 1$ and $u(x)$ is a non-decreasing function. In this case $Q_n(f)$ is called a quadrature of the interpolatory type for reasons we explicit in the following sections. Thus, the UT as per Definition 20 is a special case of this interpolatory quadrature in which we recognize that $f(x) = x^k$ are monomials, the sigma points $s_i = x_i$ are the abscissas of the mechanical quadrature, the weights $w_i = \lambda_i$ are the Cotes numbers and $du(x) = p_x(x)dx$ relates to the probability density function that characterizes the RV of interest. We follow by studying methods to choose the abscissas x_i and weights λ_i .

Here, the integral in Eq. (2.38) is understood in the Riemann-Stieljes sense: let $f(t)$ be a bounded function on the closed interval $[a, b]$ and α an increasing function defined on $[a, b]$. The partition S_n then defines n closed subintervals and we choose arbitrary points $t_k^* \in [t_{k-1}, t_k]$ for $1 \leq k \leq n$. Finally, define the sum

$$\sigma(P, f, \alpha, t^*) = \sum_{k=1}^n f(t_k^*) \Delta \alpha_k$$

in which $\Delta \alpha_k = \alpha(t_k) - \alpha(t_{k-1})$. The Riemann-Stieltjes integral is the number I such that for every $\epsilon > 0$, there corresponds a partition S_ϵ of $[a, b]$ such that for every partition S finer than S_ϵ we have

$$|\sigma(P, f, \alpha, t^*) - I| < \epsilon.$$

This is a more powerful and well-behaved integral definition. It provides a formal definition for the situation in which the variable of integration that controls the integral computation pace presents concentrated quantities apart from continuous contributions that correspond to a discontinuous $u(x)$ distribution with finite jumps. For a more detailed discussion, refer to [28].

2.3.1 The Interpolatory Quadrature

The interpolatory quadrature defined in Eq. (2.38) is obtained by replacing the integrand $f(x)$ with a suitable interpolating polynomial $P(x)$ and considering $\int_a^b P(x)du(x)$ as an approximation for $\int_a^b f(x)du(x)$. Interpolation can be understood as the process of choosing a continuous function that agrees with discrete data. In the past it was used to convey information from tables. Nowadays it is one of the foundations for building numerical methods for integration, differential equation solving and related problems. We start by defining the interpolation problem.

Definition 23 The *interpolation problem* consists of determining a set of parameters a_i so that for $n + 1$ given real or complex pairs (x_i, f_i) , $i = 0, \dots, n$, with $x_i \neq x_k$ for $i \neq k$,

$$\Phi(x_i; a_0, \dots, a_n) = f_i \quad (2.39)$$

holds for $i = 0, \dots, n$ in which $\Phi(x; a_0, \dots, a_n)$ is a family of single variable functions completely defined by the parameters a_i [29]. We call the pairs (x_i, f_i) interpolation *support points*, the locations x_i *support abscissas* and the values f_i *support ordinates*.

In this work we are specially interested in the classical *polynomial interpolation* problem in which

$$\Phi(x; a_0, \dots, a_n) \equiv a_0 + a_1x + a_2x^2 + \dots + a_nx^n. \quad (2.40)$$

We denote π_n the set of all real or complex polynomials whose degrees do not exceed n . Next we reproduce an important theorem first discovered by Edward Waring and made famous by Joseph Louis Lagrange.

Theorem 2.3.1 For $n + 1$ arbitrary support points (x_i, f_i) , with $i = 0, \dots, n$ and $x_i \neq x_k$ for $i \neq k$, there exists a unique polynomial $P \in \pi_n$ with $P(x_i) = f_i$, for $i = 0, 1, \dots, n$.

The proof is interesting for itself as it brings a building method for such polynomials [29].

Proof Uniqueness: Let $P_1, P_2 \in \pi_n$ be two polynomials with $P_1(x_i) = P_2(x_i) = f_i$ for $i = 0, 1, \dots, n$. The polynomial $P = P_1(x) - P_2(x)$ has degree at most n and thus $P \in \pi_n$. We note that the support abscissas x_i are the $n + 1$ zeros of P but by the fundamental theorem of algebra P should have exactly n roots. Therefore, P must vanish identically and $P_1 = P_2$. \square

Proof Existence: We construct the family of *Lagrange polynomials* in the form

$$\begin{aligned} \ell_i(x) &= \frac{(x - x_0) \dots (x - x_{i-1})(x - x_{i+1}) \dots (x - x_n)}{(x_i - x_0) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_n)} \\ &= \frac{w(x)}{(x - x_i)w'(x_i)} \quad \text{with } w(x) = \prod_{i=0}^n (x - x_i). \end{aligned} \quad (2.41)$$

We note that $\ell_i \in \pi_n$ for $i = 0, \dots, n$ and

$$\ell_i(x_k) = \delta_{ik} = \begin{cases} 1 & \text{if } i = k, \\ 0 & \text{if } i \neq k. \end{cases} \quad (2.42)$$

Then, the solution of the interpolation problem is

$$L(x) = \sum_{i=0}^n f_i \ell_i(x) = \sum_{i=0}^n f_i \prod_{\substack{k \neq i \\ k=0}}^n \frac{x - x_k}{x_i - x_k}. \quad (2.43)$$

Eq. 2.43 is called *Lagrange Interpolation formula*. □

To give the reader some intuition about the Lagrange interpolation, Fig. 2.3 shows the process for the support points

$$\begin{aligned} (x_1 = -1.8, y_1 = 1.3) \\ (x_2 = -1.1, y_2 = 1.0) \\ (x_3 = 0.4, y_3 = -1.0) \\ (x_4 = 1.4, y_4 = -0.5) \\ (x_5 = 2.1, y_5 = 1.0). \end{aligned}$$

Fig. 2.3a shows the application of Eq. (2.41) to get the Lagrange polynomial basis. Note that on each abscissa x_i , $\ell_i(x_i) = 1$ and all other polynomials vanish on that abscissa. Fig. 2.3b shows the scaled Lagrange polynomials $y_i \ell_i(x)$ and the Lagrange interpolation polynomial $L(x)$. The Lagrange interpolation is rarely used as in Eq. (2.43) for efficiency reasons but it is an important theoretical tool to prove theorems on approximation theory.

Next we consider an arbitrary partition of the closed interval $[a, b]$ with n abscissas x_n as in Eq. (2.35). Let P_n be the polynomial of degree at most n that interpolates $f(x)$ at the partition abscissas, i.e.,

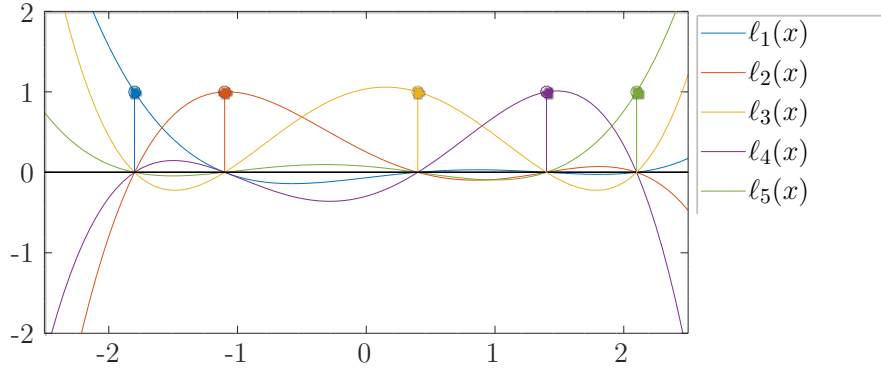
$$P_n(x_i) = f(x_i) = f_i \quad (2.44)$$

for $i = 1 \dots n$. According to Lagrange's interpolation formula

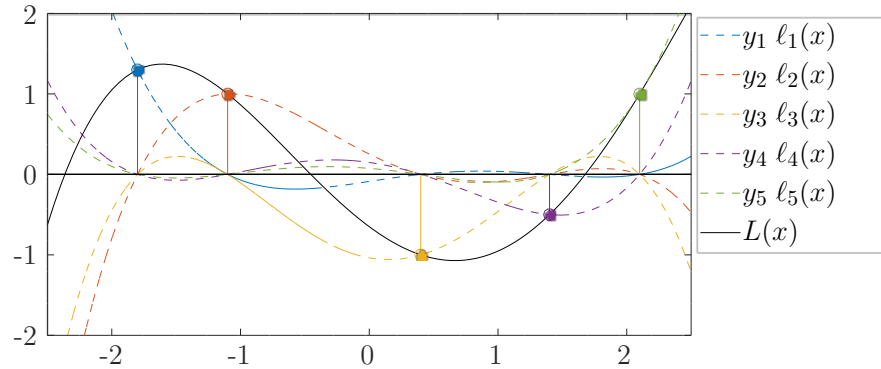
$$P_n(x) = \sum_{i=0}^n f_i \ell_i(x), \quad \ell_i(x) = \prod_{\substack{k \neq i \\ k=0}}^n \frac{x - x_k}{x_i - x_k}. \quad (2.45)$$

Integration yields

$$\begin{aligned} \int_a^b P_n(x) du(x) &= \sum_{i=0}^n f_i \int_a^b \ell_i(x) du(x) \\ &= \sum_{i=0}^n \lambda_i f_i \end{aligned}$$



(a) Lagrange polynomials basis.



(b) Lagrange interpolation polynomial.

Figure 2.3: Interpolation showing the $\ell_i(x)$ polynomial basis and the Lagrange interpolation polynomial.

which establishes the weights in Eq. 2.27 with

$$w_i = \lambda_i = \int_a^b \ell_i(x) du(x). \quad (2.46)$$

for $i = 1, \dots, n$.

We note that Eq. (2.46) holds for an arbitrary partition choice suggesting a method to compute the UT weights for a given set of sigma-points. It however does not indicate how to choose a suitable set of sigma-points which we address next.

2.3.2 Gaussian quadrature

Gaussian quadrature is a classical and well-known numerical integration method based on the idea that it is possible to maximize the order of the integration method by a clever choice of the quadrature abscissas x_i . In this section we revisit the main results related to the theory of the UT.

We start by defining the scalar product of two functions defined in the interval $[a, b]$ with

respect to a weight function $u(x)$

$$(f, g) = \int_a^b f(x)g(x) du(x), \quad (2.47)$$

where we assume that $f(x)$ and $g(x)$ are in the $L_2[a, b]$ space for which the integral

$$(f, f) = \int_a^b f(x)^2 du(x)$$

is well defined and finite and that $u(x)$ is a non-decreasing function in $[a, b]$ which is not constant. For a fixed $u(x)$ the orthogonality with respect to the distribution $du(x)$ is defined by the relation

$$(f, g) = 0 \quad (2.48)$$

in which case we shall use the expression “ $f(x)$ is orthogonal to $g(x)$ ” [27]. If $u(x)$ is absolutely continuous, which is the case treated in the scope of this work, the scalar product reduces to

$$(f, g) = \int_a^b f(x)g(x) p_x(x) dx \quad (2.49)$$

in which $p_x(x)$ is the continuous probability density function of interest in the UT computation.

The following theorem establishes the existence of a sequence of *orthogonal polynomials* associated with the distribution $p_x(x)$. For the proof, refer to [29].

Theorem 2.3.2 *There exist orthogonal polynomials $\rho_j \in \pi_j$, $j = 0, 1, 2, \dots$, such that*

$$(\rho_i, \rho_k) = 0, \text{ for } i \neq k.$$

These polynomials are uniquely determined by a three term recurrence relation given by

$$\rho_0(x) = 1 \quad (2.50)$$

$$\rho_{i+1}(x) = (x - \alpha_{i+1})\rho_i(x) - \beta_{i+1}\rho_{i-1}(x) \quad (2.51)$$

for $i \geq 0$, $\rho_{-1}(x) = 0$ and

$$\alpha_{i+1} = \frac{(x\rho_i(x), \rho_i(x))}{(\rho_i(x), \rho_i(x))} \quad \text{for } i \geq 0,$$

$$\beta_{i+1}^2 = \begin{cases} 1 & \text{for } i = 0, \\ \frac{(\rho_i(x), \rho_i(x))}{(\rho_{i-1}(x), \rho_{i-1}(x))} & \text{for } i \geq 1. \end{cases}$$

The sequence of orthogonal polynomials for a given distribution depends only on the distribution itself. Table 2.1 show the orthogonal polynomials associated with some classical distributions adapted from [30, 31].

Every polynomial $\rho \in \pi_k$ can be represented as a linear combination of the orthogonal polynomials ρ_i , $i \leq k$. It follows that

Table 2.1: Orthogonal polynomials associated with classical distributions

Name	Notation	Interval	Weight function	Three Term Recurrence Relation		
				α_k	β_0	$\beta_k, k \geq 1$
Legendre	P_n	$[-1, 1]$	$u(x) = 1$	0	2	$\frac{1}{4 - k^{-2}}$
Hermite	H_n	$(-\infty, \infty)$	$u(x) = e^{-x^2}$	0	$\sqrt{\pi}$	$k/2$
Laguerre	L_n	$[0, \infty)$	$u(x) = e^{-x}$	$2k + 1$	1	k^2
Jacobi	J_n	$[-1, 1]$	$u(x) = (1 - x)^\alpha(1 + x)^\beta$	α_k^J	β_0^J	β_k^J
			$\beta^2 - \alpha^2$			
			$\alpha_k^J = \frac{(2k + \alpha + \beta)(2k + \alpha + \beta + 2)}{2^{\alpha+\beta+1}\Gamma(\alpha + 1)\Gamma(\beta + 1)}$			
			$\beta_0^J = \frac{\Gamma(\alpha + \beta + 1)}{4k(k + \alpha)(k + \beta)(k + \alpha + \beta)}$			
			$\beta_k^J = \frac{(2k + \alpha + \beta)^2(2k + \alpha + \beta + 1)(2k + \alpha + \beta - 1)}{4k(k + \alpha)(k + \beta)(k + \alpha + \beta)}$			

Corollary 2.3.3 $(\rho, \rho_n) = 0$ for all $\rho \in \pi_{n-1}$.

We need this property to derive an important theorem about the roots of orthogonal polynomials.

Lemma 2.3.4 *The zeros x_i of the orthogonal polynomials $\rho_n(x)$ associated with the distribution $du(x)$ on the interval $[a, b]$ are real and simple, i.e., $x_j \neq x_k$ if $j \neq k$, and are located in the interior of the interval $[a, b]$.*

The proof is based on the orthogonality property [29].

Proof We construct the polynomial

$$q(x) = \prod_{j=1}^l (x - x_j) \in \pi_l$$

such that its zeros are the roots of $\rho_n(x)$ which lie in (a, b) and which are of odd multiplicity. Thus, the polynomial $\rho_n(x)q(x)$ does not change sign in $[a, b]$, that is, is non-negative or non-positive throughout $[a, b]$. It follows that

$$(\rho_n, q) = \int_a^b \rho_n(x)q(x) du(x) \neq 0.$$

Since $(\rho_n, q) = 0$ if $q(x)$ is a polynomial of degree less than n by Corollary 2.3.3 thus $l = n$ shall hold. \square

We finally arrive at the main result for this section, the existence of the numerical quadrature [27].

Theorem 2.3.5 *If $x_1 < x_2 < \dots < x_n$ denote the zeros of $\rho_n(x)$, there exist real numbers*

$\lambda_1, \lambda_2, \dots, \lambda_n$ such that

$$\int_a^b p(x) du(x) = \sum_{i=1}^n \lambda_i p(x_i) \quad (2.52)$$

whenever $p(x)$ is an arbitrary polynomial of degree at most $2n - 1$. The distribution $du(x)$ and the integer n uniquely determine these numbers λ_n .

The relevant proof of Theorem 2.3.5 for the UT theory regards the special cases in which $p(x)$ is a monic polynomial, i.e., $p(x) = x^k$, $k = 0, 1, 2, \dots, 2n - 1$. The following proof is based on results from [27, 29].

Proof We begin by constructing the Lagrange interpolation polynomial $L(x)$ of degree $n - 1$ using x_n as the interpolation abscissas. Since the Lagrange basis polynomials are unique and have the interpolation abscissas as zeros (Theorem 2.3.1) they can be expressed in terms of the orthogonal polynomial ρ_n with zeros at x_i , $i = 1, 2, \dots, n$, associated with the distribution $du(x)$, i.e.,

$$L(x) = \sum_{i=1}^n p(x_i) \ell_i(x) = \sum_{i=1}^n p(x_i) \frac{\rho_n(x)}{(x - x_i) \rho_n'(x_i)}.$$

Now $p(x) - L(x)$ have degree at most $2n - 1$ and is divisible by $\rho_n(x)$ so that

$$p(x) = \rho_n(x)r(x) + L(x)$$

in which r is a polynomial in π_{n-1} that can be expressed as a linear combination of orthogonal polynomials

$$r(x) = \sum_{i=1}^n \beta_i \rho_i(x).$$

Therefore

$$\int_a^b p(x) du(x) = \int_a^b L(x) du(x) + \int_a^b \rho_n(x)r(x) du(x)$$

and since $\int_a^b \rho_n(x)r(x) du(x) = (\rho_n, r) = 0$ because r is a linear combination of $\rho_n(x)$ it follows that

$$\int_a^b p(x) du(x) = \int_a^b L(x) du(x) = \sum_{i=1}^n p(x_i) \int_a^b \ell_i(x) du(x)$$

which establishes Eq. (2.52) with

$$\lambda_i = \int_a^b \ell_i du(x) = \int_a^b \frac{\rho_n(x)}{(x - x_i) \rho_n'(x_i)} du(x), \quad i = 1, 2, \dots, n. \quad (2.53)$$

Conversely, let Eq. (2.52) hold for an arbitrary polynomial $p(x) \in \pi_{2n-1}$. Then choose $p(x) = l(x)r(x)$ with $l(x) = (x - x_1)(x - x_2) \dots (x - x_n)$ and $r(x)$ an arbitrary polynomial in π_{n-1} . From Eq. (2.52)

$$\int_a^b l(x)r(x) du(x) = (l, r) = \sum_{i=1}^n \lambda_i l(x_i)r(x_i) = 0$$

since the numbers x_i are by construction the zeros of $l(x)$ and we conclude that $l(x) = \beta \rho_n(x)$ in

which β is a constant. □

The first part of the proof establishes the existence of the Gaussian quadrature and provides a formula for the weights λ_i , $i = 1, 2, \dots, n$. The second part establishes that if the quadrature exist for a polynomial of order $2n - 1$ the support abscissas shall be the zeros of the orthogonal polynomial $\rho_n(x)$ associated with the distribution $du(x)$. Our definition 20 establishes that the UT can be seen as a special case of the mechanical quadrature in 2.37 with $f(x) = x^k$, $k = 1, 2, \dots, \alpha$, the monic polynomials, we conclude that the zeros of an orthogonal polynomial can constitute a set of sigma-points with the weights $w_i = \lambda_i$. To conclude this chapter, we follow by showing that the UT as a Gaussian quadrature method maximizes the number of moments preserved by the transformation and by showing an efficient method for computing the weights w_i .

Theorem 2.3.6 *It is not possible to find numbers x_i , $i = 1, \dots, n$, such that Eq. (2.52) holds for all polynomials $p(x) \in \pi_{2n}$.*

Proof Assume that x_i and λ_i , $i = 1, \dots, n$, are such that Eq. (2.52) holds for all polynomials $p(x) \in \pi_{2n}$. Define the polynomial $q(x) \in \pi_{2n}$ as

$$q(x) = \prod_{j=1}^n (x - x_j)^2$$

It follows that

$$\int_a^b q(x) du(x) = \sum_{i=1}^n \lambda_i q(x_i) = 0$$

but since $q(x)$ is clearly a nonnegative function and $\int_a^b q(x) du(x) > 0$, we conclude by contradiction that Eq. (2.52) does not hold in general for a polynomial in π_{2n} . □

From Theorem 2.3.6 we establish the minimum number of sigma-points for the α -th order UT to be

$$\alpha = 2n - 1 \tag{2.54}$$

where n is the number of sigma-points for the UT.

Finally, we state two theorems which provide a practical way for computing the sigma-points and weight pairs $\{s_i, w_i\}$ based on the theory of the Gaussian quadrature. For the proof of Theorem 2.3.7 refer to [29]; for Theorem 2.3.8, refer to [27].

Theorem 2.3.7 *The roots x_i , $i = 1, \dots, n$, of the orthogonal polynomial $\rho_n(x)$ are the eigenvalues*

of the tridiagonal matrix

$$J_n = \begin{bmatrix} \alpha_1 & \sqrt{\beta_2} & 0 & 0 & 0 & \dots & 0 \\ \sqrt{\beta_2} & \alpha_2 & \sqrt{\beta_3} & 0 & 0 & \dots & 0 \\ 0 & \sqrt{\beta_3} & \alpha_3 & \sqrt{\beta_4} & 0 & \dots & 0 \\ & & \ddots & \ddots & \ddots & & \\ 0 & 0 & \dots & \sqrt{\beta_{n-2}} & \alpha_{n-2} & \sqrt{\beta_{n-1}} & 0 \\ 0 & 0 & \dots & 0 & \sqrt{\beta_{n-1}} & \alpha_{n-1} & \sqrt{\beta_n} \\ 0 & 0 & \dots & 0 & 0 & \sqrt{\beta_n} & \alpha_n \end{bmatrix} \quad (2.55)$$

where α_i and β_i are the coefficients of the three term recurrence relation defined in Eq. (2.3.2).

Theorem 2.3.8 Let $v^{(i)} = (v_1^{(i)}, \dots, v_n^{(i)})^T$ be an eigenvector of J_n defined in Eq. (2.55) for the eigenvalue X_i , i.e., $J_n v^{(i)} = X_i v^{(i)}$. Suppose $v^{(i)}$ is scaled in such way that

$$v^{(i)T} v^{(i)} = (\rho_0, \rho_0) = \int_a^b du(x).$$

Then the weights w_i are given by

$$w_i = (v_1^{(i)})^2, \quad i = 1, \dots, n. \quad (2.56)$$

2.3.3 Example

We summarize the procedure to compute the UT sigma-points and weights from the UT in the following algorithm.

Algorithm 2.3.1 Gaussian quadrature method for the Unscented Transform.

1. Start with the desired continuous probability distribution $p_x(x)$ and the desired number of sigma-points n . Compute the α_n and β_n , $n = 1, \dots, n$, terms in the three term recurrence relation using Eq. (2.3.2).
2. Build the Jacobi J_n matrix in equation (2.55)
3. Use any known method to compute the eigenvalues of J_n and make them the quadrature abscissas x_i .
4. Use any known method to compute the eigenvectors of J_n and scaled them as in Theorem 2.3.8. Use the first eigenvector components as the λ_i quadrature weights.
5. Scale the quadrature abscissas x_i and weights λ_i according to the desired probability distribution function and use them as the sigma-points s_i and weights w_i of the UT.

To the sake of clarity, we detail the computation of the 5-point UT for the standard Gaussian distribution on the interval. Appendix II shows the Matlab source code for other probability distributions.

The standard Gaussian distribution is described by the probability density function

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (2.57)$$

in which $\mu = 0$ is the mean value (m_1) and $\sigma^2 = 1$ is the variance (m_2) of the distribution. We note that

$$f(x) = \frac{1}{\sqrt{2\pi}} u(t)$$

where $u(x)$ is the weight function associated with the Hermite polynomials and $t = x/4$.

We start by computing the $\alpha_i, \beta_i, i = 1, \dots, 5$, coefficients for the three term recurrence relation in Eq. (2.3.2), to know

$$\begin{aligned} \alpha_1 &= 0, & \beta_1 &= \sqrt{\pi} = 1.77245 \\ \alpha_2 &= 0, & \beta_2 &= 0.5 \\ \alpha_3 &= 0, & \beta_3 &= 1.0 \\ \alpha_4 &= 0, & \beta_4 &= 1.5 \\ \alpha_5 &= 0, & \beta_5 &= 2.0. \end{aligned}$$

Next, we compute the Jacobi matrix J_n

$$J_n = \begin{bmatrix} 0 & \sqrt{0.5} & 0 & 0 & 0 \\ \sqrt{0.5} & 0 & \sqrt{1.0} & 0 & 0 \\ 0 & \sqrt{1.0} & 0 & \sqrt{1.5} & 0 \\ 0 & 0 & \sqrt{1.5} & 0 & \sqrt{2.0} \\ 0 & 0 & 0 & \sqrt{2.0} & 0 \end{bmatrix}$$

with eigenvalues x_i and eigenvectors v_i

$$\begin{aligned} x_1 &= -2.0202 & v_1 &= \begin{bmatrix} 0.10610 & -0.30313 & 0.53735 & -0.63884 & 0.44721 \end{bmatrix} \\ x_2 &= -0.95857 & v_2 &= \begin{bmatrix} -0.47125 & 0.63884 & -0.27915 & -0.30313 & 0.44721 \end{bmatrix} \\ x_3 &= 0.0 & v_3 &= \begin{bmatrix} 0.73030 & 0.0 & -0.51640 & 0.0 & 0.44721 \end{bmatrix} \\ x_4 &= 0.95857 & v_4 &= \begin{bmatrix} -0.47125 & -0.63884 & -0.27915 & 0.30313 & 0.44721 \end{bmatrix} \\ x_5 &= 2.0202 & v_5 &= \begin{bmatrix} 0.10610 & 0.30313 & 0.53735 & 0.63884 & 0.44721 \end{bmatrix}. \end{aligned}$$

We take the first eigenvector and square its components, according to Theorem 2.3.8, to be the weights w_i of the Gaussian quadrature and of the UT

$$w_1 = 0.011257$$

$$w_2 = 0.222076$$

$$w_3 = 0.533333$$

$$w_4 = 0.222076$$

$$w_5 = 0.011257.$$

Finally, we need to properly scale the x_i to have the sigma-points $s_i = \sqrt{2} x_i$

$$s_1 = -2.8570$$

$$s_2 = -1.3556$$

$$s_3 = 0.0$$

$$s_4 = 1.3556$$

$$s_5 = 2.8570.$$

2.4 Chapter Final Remarks

This chapter presented a formal definition for the Unscented Transform. For the best of our knowledge, we are the first to present such a definition in a sufficiently abstract manner to encompass a broad range of approaches reported in the literature. We followed by showing a direct method for computing the UT which was the basis for the developments reported in Chapter 3. Due to numerical instabilities inherent to the direct method as discussed in section 2.2.1, we extended our knowledge about the UT by showing that it is a special form of the Gaussian quadrature. We leave this chapter with an efficient method for computing the optimal number of sigma-points for the α -th order UT without excluding other possible techniques briefly discussed as future work possibilities in Chapter 6.

Chapter 3

Probability Functions Estimation

In this chapter we show how the UT can be used as an alternative to Monte Carlo methods by means of the description of a case study on circuit yield estimation. For an advanced discussion about Monte Carlo formulations and more applications we recommend the works by Kalos [32], Robert [33] and Pereyra [34]. We follow by presenting a contribution of this work, an extension to the initial UT formulation to get more accurate estimates for probability functions [35].

3.1 The UT as an Alternative to Monte Carlo

Suppose we want to calculate the integral

$$I = \int_0^1 g(t)dt. \quad (3.1)$$

Let \mathbf{x} be a RV with uniform distribution in the interval $(0, 1)$ and form a new RV $\mathbf{y} = g(\mathbf{x})$. The first moment of \mathbf{y} will be

$$E\{g(x)\} = \int_0^1 g(x)p_x(x)dx = \int_0^1 g(x)dx. \quad (3.2)$$

That is, we can use the relative frequency interpretation of probabilities, Eq. (??), in order to estimate I . This yields

$$I = E\{g(x)\} \approx \frac{1}{n} \sum_n g(x_i). \quad (3.3)$$

by the strong law of large numbers [33].

The data x_i , no matter how they are obtained, are random numbers. If, therefore, we can numerically generate such numbers, we have a method for determining I . The basic idea behind Monte Carlo methods is to numerically sample N such random numbers, x_i ; map them to the nonlinear transformation, $y_i = g(x_i)$; and use Eq. (3.3) to estimate the result [19].

While the naive Monte Carlo works for simple examples, that is not the case in real life problems. The convergence rate of $O(N^{-1/2})$ of Monte Carlo methods makes a considerable part of the

literature to devote to develop new strategies to improve sampling and error estimates. Variance reduction techniques such as importance sampling and stratification sampling and the use of quasi-random variables are the basis for more advanced methods [36].

We can use the UT to estimate I as the sigma points capture the first moment behavior of the nonlinear mapping $\mathbf{y} = g(\mathbf{x})$ by a polynomial of order $2m - 1$, where m is the number of sigma points. In this way, Eq. (3.3) becomes

$$I = E\{g(x)\} \approx \sum_{i=1}^m w_i g(S_i). \quad (3.4)$$

By following the algorithm described by Eqs. (2.28) to (2.30), any Monte Carlo engine may be modified to use the UT sigma points instead of a random number generator and post-process the results according to Eq. (3.4). As the sigma points do not depend on the nonlinear mapping $\mathbf{y} = g(\mathbf{x})$, they can be precalculated with great accuracy and tabulated to use on specific applications. In the next section we present a case study on how to use the UT as an alternative to Monte Carlo.

3.2 Yield Estimation Case Study

Yield estimation is an important design concern on modern microelectronics industry as it heavily impacts the cost of a functional die and gross profit margin of an electronic product [37]. It is, then, desirable that yield estimation starts at an early stage of the design flow by the use of abstract models. In this work we define yield to be the probability of a device on the wafer to perform properly, i.e. to achieve a performance under a specified range. Monte Carlo has become the *de facto* technique for yield estimation during design phase as it is suitable for arbitrary circuits, complex statistical models and allows arbitrary accuracy [38]. This flexibility is achieved with the cost of computational effort.

Consider a ring oscillator composed by a closed loop of n cascaded inverters as shown in Figure 3.1. For an odd n , the output frequency of this system will be given by

$$f_{\text{out}} = \frac{1}{2 \sum_n d_n} \quad (3.5)$$

where d_n is the delay of the n -th inverter [39]. Due to process variations, the delay for each inverter may vary between different batches of fabrication. It is convenient to model d_n as a random variable. For this case study, we consider $n = 3$, $d_n = \mathcal{N}(d_0, \Delta d)$ to be gaussian distributed with $d_0 = 1\mu s$ mean and $\Delta d = 0.02\mu s$ variance, which should gives us an expected output frequency of 166.7kHz.

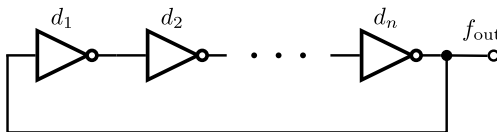


Figure 3.1: Ring oscillator.

In this case, it is possible to find an analytical expression for the density function of f_{out} and we derive next to use it as a benchmark of the method. Often, the random variable transformation is much more complex than this to find an analytical solution.

The probability density function $f_y(y)$ of a random variable $\mathbf{y} = g(\mathbf{x})$ will be given by

$$p_{\mathbf{y}}(y) = \frac{p_{\mathbf{x}}(x)}{\left| \frac{dy}{dx} \right|} \quad (3.6)$$

where the derivative is evaluated at $x = g^{-1}(y)$, considering $g(x)$ to be one-to-one and invertible [19]. If the function is many-to-one, then its inverse will have multiple roots x_i and Eq. (3.6) becomes

$$p_{\mathbf{y}}(y) = \sum_k \left(\frac{p_{\mathbf{x}}(x_i)}{\left| \frac{dy}{dx_i} \right|} \right). \quad (3.7)$$

It follows that, for $y = f_{\text{out}} = \frac{1}{2x}$

$$p_{\mathbf{y}}(y) = \frac{1}{2y^2} p_{\mathbf{x}}\left(\frac{1}{2y}\right)$$

where $x = \sum d_n$ will be normal distributed with $\mu = 3 \times d_0$ mean and $\sigma = 3 \times \Delta d$ variance. Finally, we get

$$p_{\mathbf{y}}(y) = \frac{1}{2y^2} \times \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\left(\frac{1}{2y} - \mu\right)^2 / 2\sigma^2} \quad (3.8)$$

We implement the Monte Carlo method as shown in Figure 3.2a. We start by setting N to be the number of simulations to run. Next, we enter a loop in which in pick random numbers, taken in accordance to the desired probability function, and evaluate the system model. After repeating this process for N times, we take an histogram of $f[n]$ and estimate the density function from it.

For the UT, we modify the algorithm as shown in Figure 3.2b. First we set m to the desired number of sigma points. Note that this also sets the number of moments we can correctly estimate at the end of the experiment to be $2m - 1$. Next, we build the set $\{s_i, w_i\}$ with m sigma points and weights. These will be fed to the model instead of random numbers. We extend the model evaluation phase to also get the correct probability of that specific event to happen according to the UT weights. We evaluate the system model to stress all sigma points combinations. At the end, we inspect $f[n]$ and merge repeated results by summing their probabilities.

This case study shows a three dimensional estimation problem as each one of the three inverters in the system model contribute with one RV to the analysis. Figure 3.3 shows typical runs for the Monte Carlo method with increasing number of points. We plot the analytical results for comparison. We can note the slow $O(N^{-1/2})$ rate of convergence of the Monte Carlo approach. This turns this basic Monte Carlo approach to be prohibitively slow in real world problems if we wants to get fine results.

Next, in Figure 3.4 we show the results we get from the UT algorithm with three sigma points for each RV and the analytically determined density function for comparison. We first note that

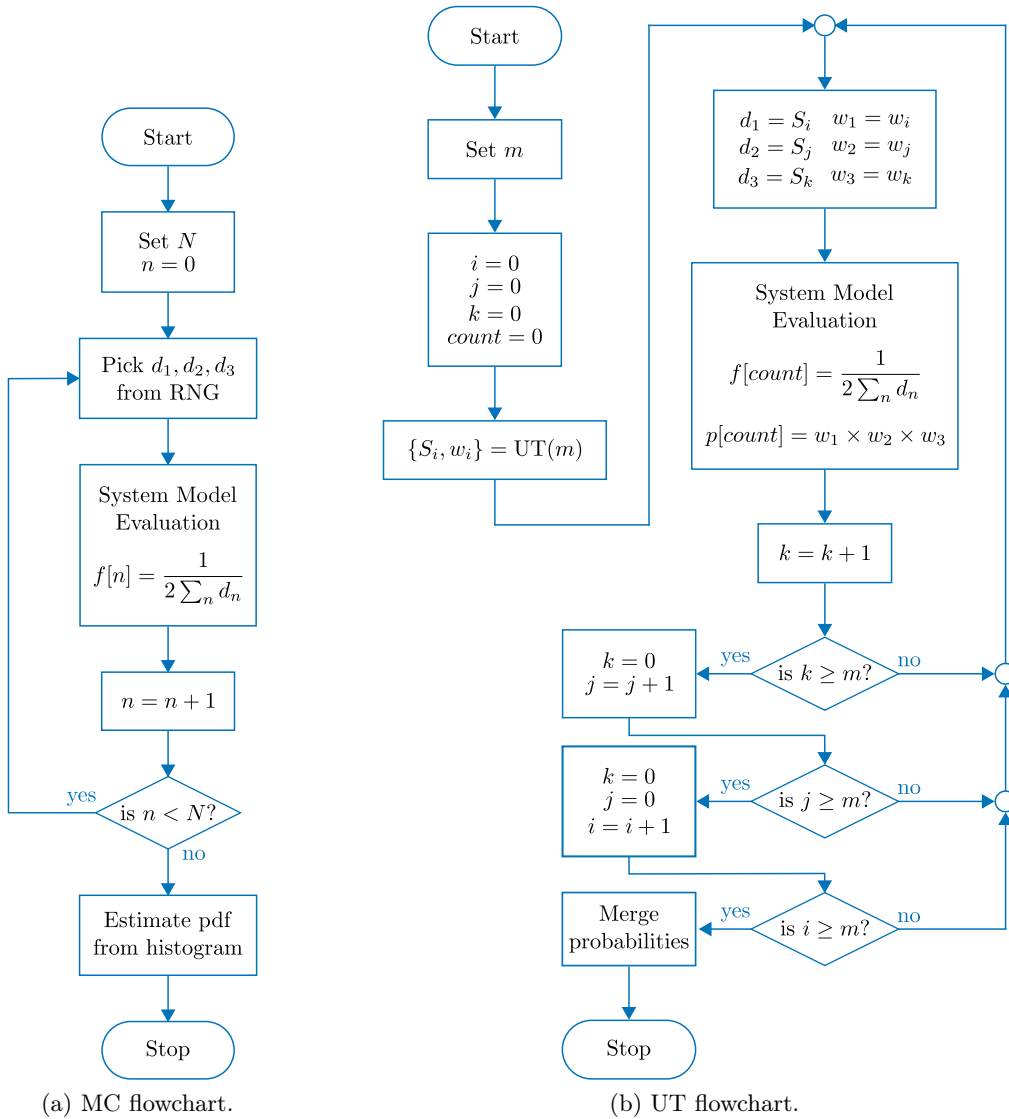


Figure 3.2: Monte Carlo and UT yield estimation application flowcharts.

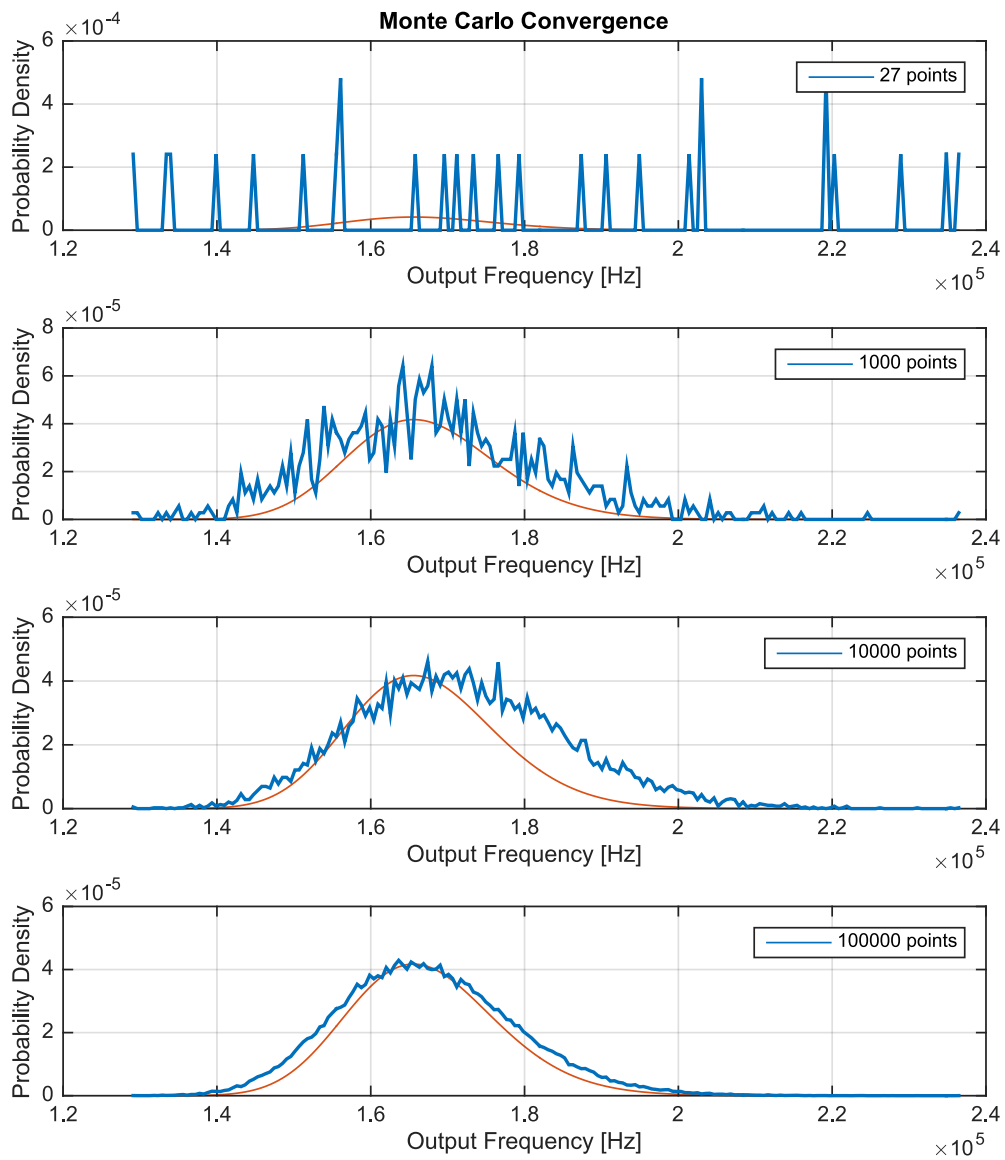


Figure 3.3: Monte Carlo experiments with increasing number of simulations to show the convergence of the method.

the output resembles a sampled and scaled version of the density function. We will explore this later to develop the moment preserving quantizer theory. Also, the UT points are always the same, as the set $\{S_i, w_i\}$ of sigma points and weights are deterministically set and depend only on the density function we use to model the system uncertainty and on the desired number of sigma points.

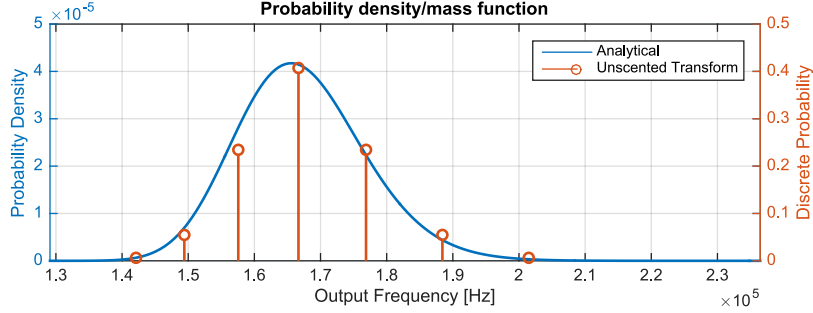


Figure 3.4: The UT estimates of the density function with 3 sigma points per random variable.

Finally, in Figure 3.5 we show the cumulative distribution function for each approach. We compare the typical run of Monte Carlo with only 27 points as this is the same number of sigma points combinations used to excite the system model on the UT algorithm. We note the staircase pattern due to the discrete nature of the Unscented Transform.

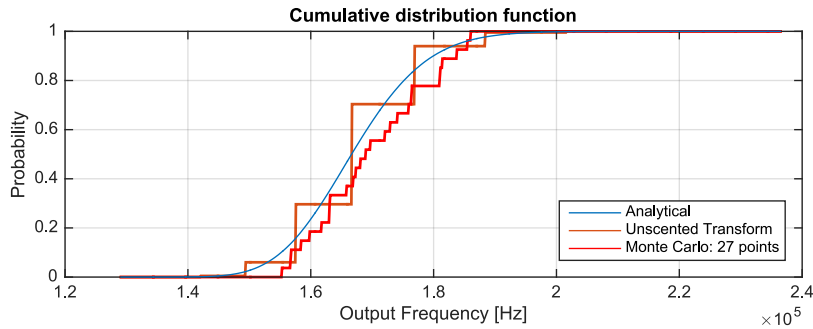


Figure 3.5: Distribution function estimates with analytical, Monte Carlo and UT approaches.

Next, in Figure 3.6 we show the relative errors for the moments estimates using the UT defined by

$$\eta = \left| \frac{m_{exact} - m_{estimate}}{m_{exact}} \right|$$

where m_{exact} are the moments calculated from Eq. (3.8) and $m_{estimate}$ are the moments estimates given from the UT. We note that the higher the number of sigma points, the more moments we correctly estimate. Also, the UT is capable of giving good estimates for the moments even with low number of sigma points.

In Figure 3.7 we present the relative errors for the moments estimates from a typical run of the Monte Carlo approach. We note the convergence of the method for increasing number of points and we note the superior performance of the UT for low number of points.

Finally, in Figure 3.8 we present the absolute error for the probability estimation using the

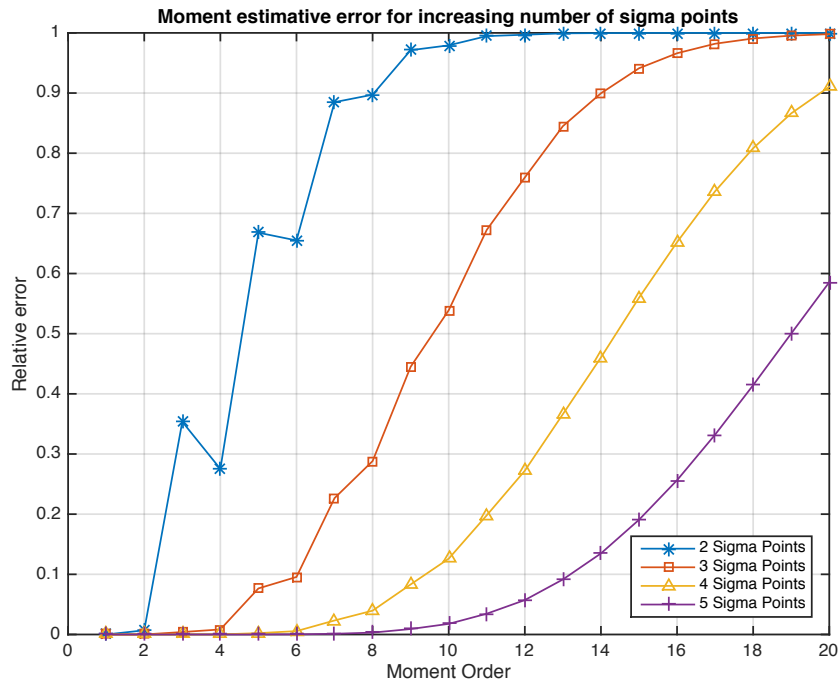


Figure 3.6: Relative error for the moments estimates given by the UT.

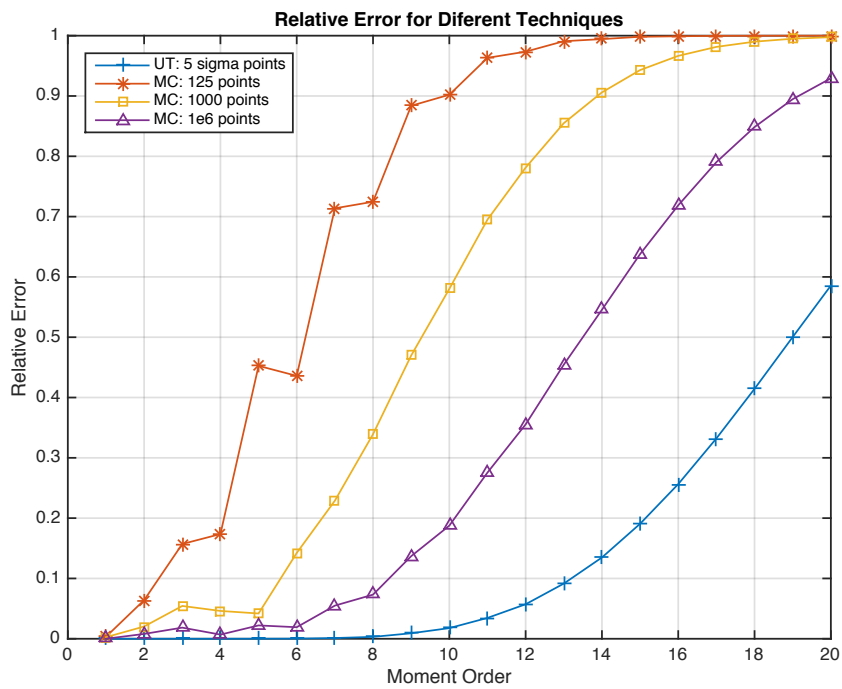


Figure 3.7: Relative error for a typical Monte Carlo run with increasing number of points.

UT. We note that the UT approach is very effective for the moments estimation but presents poor performance for estimating the probability function. In the next section we present an extension to the UT theory to improve this.

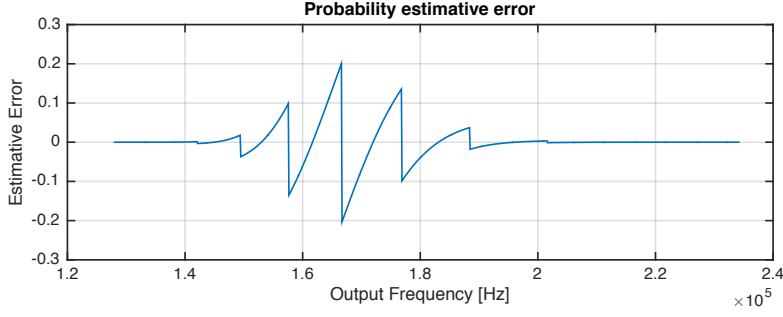


Figure 3.8: Absolute error for the probability estimation using the UT.

3.3 The Extended Unscented Transform

One issue of the UT is that it maps a continuous random variable, \mathbf{x} , into a discrete one, \mathbf{X} . If one is interested in approximating the continuous probability distribution of $\mathbf{y} = g(\mathbf{x})$ after the nonlinear transformation $g(\cdot)$ from the discrete probability function $\mathbf{Y} = g(\mathbf{X})$, one has to interpolate the mapped sigma points back into a continuous distribution. Menezes et al. [40], inspired on the observation that the cumulative distribution function from \mathbf{X} will present a staircase pattern as shown in Figure 3.9c, proposes this kind of interpolation. On the same work, the authors suggest without detailed derivations a smoothing technique based on approximating the cumulative probability function by a piecewise linear function and we develop this approach next.

This strategy implies a new formulation for the UT, based now on a new random variable \mathbf{X}_{bars} with a histogram like probability density function $p_{\mathbf{X}_{\text{bars}}}(x)$, with discrete number of bars,

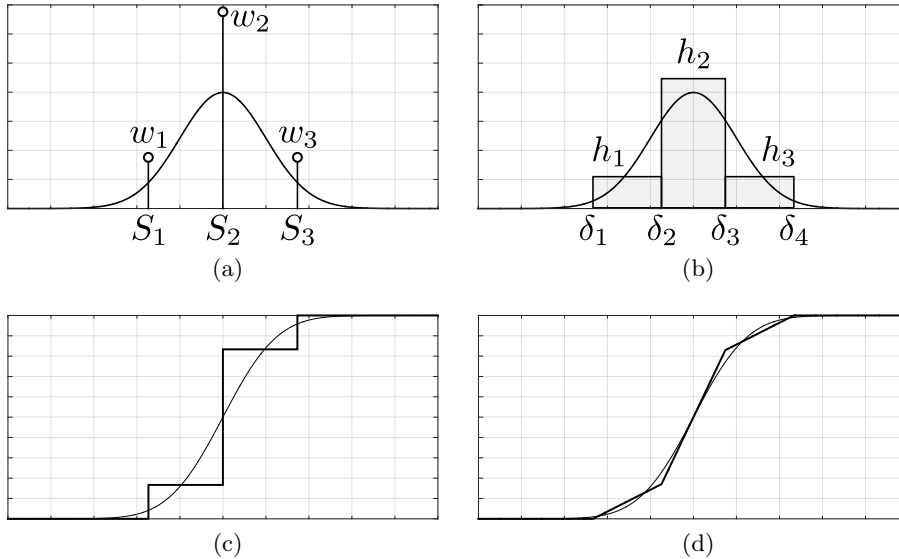


Figure 3.9: Different UT formulations. (a) Continuous probability distribution and the UT points. (b) Continuous probability function and the histogram-like approximation. (c) Cumulative distribution functions for the continuous and the point-wise UT. (d) Cumulative distribution functions for the continuous and the histogram-like approximation. The piece-wise linear behavior observed in (d) is the inspiration for the ExUT development.

as shown in Figure 3.9b, and not on discrete points. With this new approach, Eq. (2.27) becomes

$$\sum_{i=1}^m h_i \int_{\delta_i}^{\delta_{i+1}} x^k dx = \int_{-\infty}^{+\infty} x^k p_{\mathbf{x}}(x) dx \quad (3.9)$$

for all $k = 0, 1, 2, \dots, 2m+1$ where m is now the desired number of bars on \mathbf{X}_{bars} probability density function, $p_{\mathbf{x}}(x)$ is the probability density function of the input RV \mathbf{x} , h_i stands for the height of the i -th bar and δ_i are the transition points between the i -th and the $(i+1)$ -th bar as shown in Figure 3.9b. It is important to note that this formulation is more general than the one presented on [40] in the sense we do not impose that $\delta_i = S_i$.

We can get a solution for Eq. (3.9) by using a modified version of Algorithm 2.2.1.

Algorithm 3.3.1 *Direct method for the Extended Unscented Transform calculation.*

First, solve the system of equations defined by Eq. (3.10) to find a set of m coefficients α_i :

$$\sum_{i=0}^m \alpha_i (N+m) M_{N+m-1} = 0 \quad (3.10)$$

for $N = 0, 1, \dots, m-1$ and $\alpha_0 = 1$ where M_r are the r -th order moment of $p_{\mathbf{x}}(x)$. Second, define a m -order polynomial π_m :

$$\pi_m(x) = \sum_{i=0}^m \alpha_i x^{m-i}. \quad (3.11)$$

The bin transition points δ_i will be the roots of this polynomial, i.e., $\pi_m(\delta_i) = 0$. Third, find the heights h_i for each bar by solving the linear system of equations:

$$\sum_{i=1}^m h_i (\delta_{i+1} - \delta_i)^r = (r+1) M_r \quad (3.12)$$

for all $r = 0, 1, \dots, m-1$.

This formulation poses a new issue for the analysis of nonlinear mappings. The convenience of the original formulation provided by Eq. (2.27) is based on the intuition that it is easier to solve a nonlinear mapping on a discrete set of points than it is on a continuous distribution, which implies the numerical integration of $\mathbf{Y}_{\mathbf{c}} = g(\mathbf{X}_{\mathbf{c}})$. The new RV \mathbf{X}_{bars} is a continuous RV. One way to tackle this is to sample into a new discrete distribution $\mathbf{X}_{\mathbf{s}}$ as shown in Figure 3.10.

By applying the expected value operator as defined in Eq (2.22) for \mathbf{X}_{bars} we get

$$m_k = \int_{-\infty}^{\infty} x^k p_{\mathbf{X}_{\text{bars}}}(x) dx = \sum_{k=1}^m \left(\int_{\delta_k}^{\delta_{k+1}} x^k h_k dx \right) \quad (3.13)$$

where m is the number of bars in \mathbf{X}_{bars} . We can multiply each term of the summation by $(\delta_{k+1} - \delta_k) / (\delta_{k+1} - \delta_k)$ to get

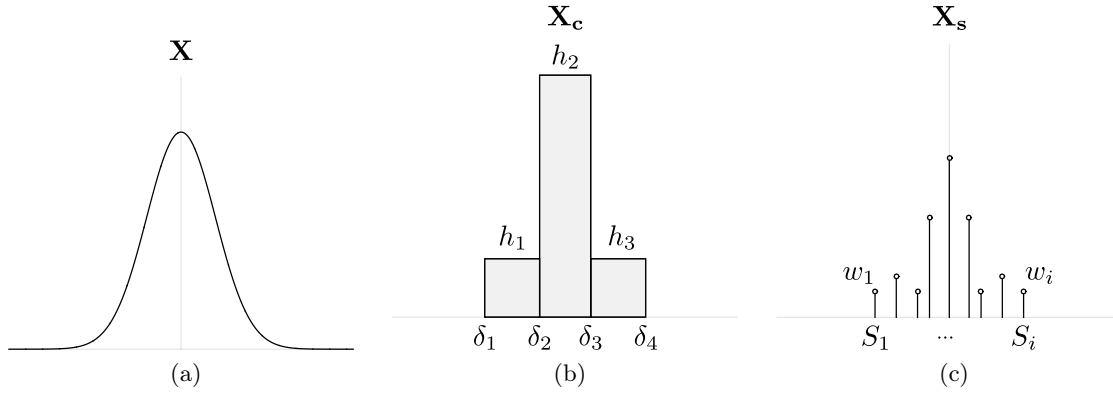


Figure 3.10: A pictorial representation of the Extended UT process. (a) The initial probability distribution, \mathbf{X} . (b) The continuous distribution, \mathbf{X}_c , obtained by solving Eqs. (3.10) to (3.12). (c) The final ExUT discrete distribution, \mathbf{X}_s , obtained by treating each bar in \mathbf{X}_c as a continuous uniform distribution as stated in Eq. (I.19).

$$m_k = \sum_{k=1}^m \left(h_k (\delta_{k+1} - \delta_k) \int_{\delta_k}^{\delta_{k+1}} x^k \frac{1}{\delta_{k+1} - \delta_k} dx \right). \quad (3.14)$$

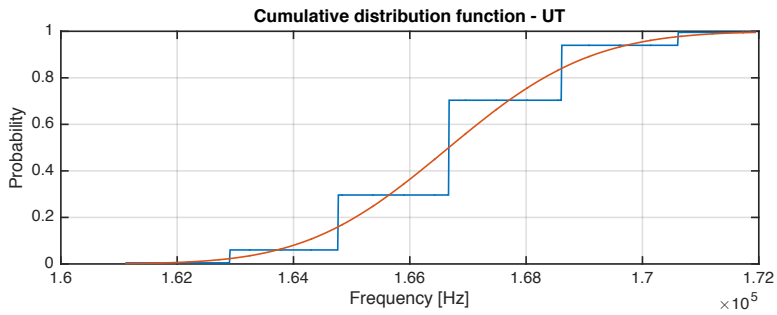
We recognize that the integral in Eq. 3.14 gives the moments of a continuous uniform distribution defined over the interval $[\delta_k, \delta_{k+1}]$, for each $k = 1, \dots, m$, to conclude

$$E\{x^k\} = \sum_{k=1}^m h_k (\delta_{k+1} - \delta_k) E\{\mathbf{U}[\delta_k, \delta_{k+1}]^k\} \quad (3.15)$$

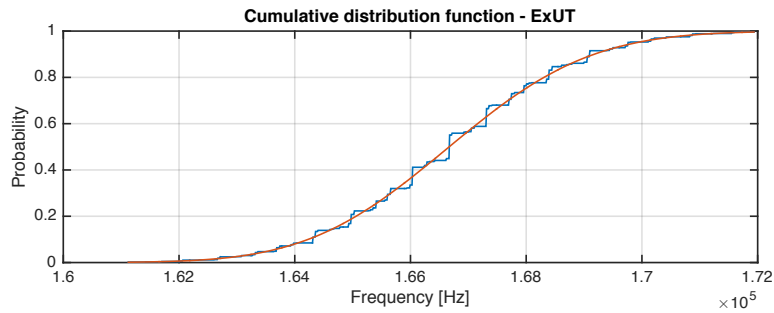
where m is the number of bars on \mathbf{X}_{bars} probability density function and $\mathbf{U}[\mathbf{a}, \mathbf{b}]$ is the continuous uniform distribution with $[a, b]$ support. Eq. (I.19) shows that we can treat each bar on \mathbf{X}_{bars} probability density function as an individual continuous uniform distribution with support $[\delta_i, \delta_{i+1}]$ and the sigma points for \mathbf{X}_s can be taken from the UT of the continuous uniform distribution. This final set of sigma points with associated probabilities will be called Extended Unscented Transform — ExUT. Figure 3.10 show the steps involved in the ExUT calculation.

Next we compare the probability estimation given by applying the ExUT to get the set $\{S_i, w_i\}$ in the algorithm shown in Figure 3.2b. Figure 3.11a shows the resulting estimate using 3 sigma points for each RV, as in Figure 3.5. Figure 3.11b, on the other hand, shows the probability estimate using ExUT set to 3 bars and 3 sigma points for each bar. We note the finer approximation we get with the ExUT.

Finally, we show the absolute error in the probability estimation with the ExUT in Figure 3.12. The ExUT is capable of improving the probability estimate due to the smoother probability function.



(a)



(b)

Figure 3.11: ExUT probability estimation. (a) The probability estimation given by the UT with 3 sigma points. (b) The probability estimation given by the ExUT with 3 bars and 3 sigma points for each bar.

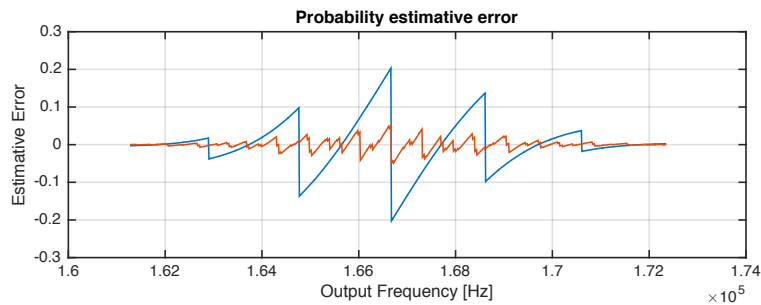


Figure 3.12: Absolute error for the probability estimation using the ExUT.

3.4 Chapter Final Remarks

This chapter presented a case study showing how to use the UT to replace the naive Monte Carlo method. It presents an extended formulation for the UT sigma-points as an approach to improve the number of sigma-points in the UT to end with smoother probability functions estimates. The proposed approach suffers from scalability issues. From Figure 3.2b one can see that the proposed algorithm has running time proportional to $O(n^k)$, where n is the number of sigma-points used to represent the delays probability functions and k is the number of inverters in the oscillator. On the other hand, this approach is completely deterministic, which leads to straightforward parallel implementations [41].

Chapter 4

Quantizer Design with the UT

In this chapter we show the main contribution of this thesis: how the UT can be understood as a framework for the design of quantizers for data conversion. We present a study case for the arcsine distribution showing interesting results about the proposal. We conclude by presenting a circuit topology for the implementation for such analog-to-digital and digital-to-analog converters. Some of the topics discussed in this chapter were published in reference [42].

4.1 Signal Modeling

Quantization is the process of mapping a large set of input values to a countable smaller set. In data conversion applications, the input can assume continuous values in a defined input range, modeled by the use of real valued variables in the bounded support $[a, b]$. The output takes discrete values that are often coded using binary words. The UT is a mathematical framework that models a continuous probability density function, $p_{\mathbf{y}}(y)$, into a discrete one, $p_{\mathbf{y}_q}(y_q)$. We propose in this work to use the UT as a model for the quantization process in data converters. Figure 4.1 illustrates this idea in which we design a quantizer that mimics the UT properties when we model its input and output signals by means of probability functions.

Papoulis [19] defines a random variable \mathbf{x} as the arbitrary process of assigning a number $\mathbf{x}(\xi)$ to every outcome ξ by satisfying the following two conditions: 1) the set $\{\mathbf{x} \leq x\}$ is an event for every x and; 2) the probabilities of the events $\{x = \infty\}$ and $\{x = -\infty\}$ equal zero. We can thus model a deterministic signal by means of a random variable.

Lets consider first how to model the time variable as a random variable. Consider a periodic signal $y = f(t)$ with period T_y . We want to take n samples of this signal in the interval $0 < t < T_y$ with sample period $T_s = T_y/n$, each $n \in \mathbb{N}_+$. In this way, we get a sequence $T_n = \{t_0 = 0, t_1 = T_y/n, \dots, t_n = (n-1)T_y/n\}$ of sample instants, each one occurring just once and thus T_n has a discrete uniform distribution with density function

$$p_t(t) = \begin{cases} 1/n & \text{if } t \in T_n \\ 0 & \text{otherwise} \end{cases} \quad (4.1)$$

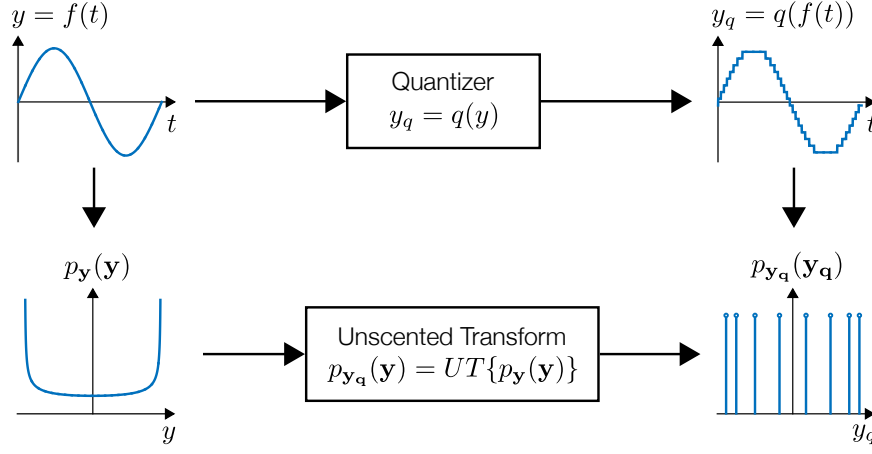


Figure 4.1: A quantizer takes an input signal $y = f(t)$ and maps it into a quantized signal $y_q = q(y)$. The UT, on the other hand, takes a continuous probability function $p_y(y)$ and maps it into a discrete probability function $p_{y_q}(y_q)$. We model a continuous signal as a probability function to show the link between the UT and the moment-preserving quantizer.

The cumulative distribution function of T_n is given by

$$P_n(x) = \frac{\left\lfloor n \left(\frac{x}{T_y} + \frac{1}{n} \right) \right\rfloor}{n}, \quad (4.2)$$

for $x \in [0, T_y]$ where $\lfloor a \rfloor$ denotes the floor function, that is, the largest integer less than or equal to a . We note that

$$n \left(\frac{x}{T_y} + \frac{1}{n} \right) - 1 \leq \left\lfloor n \left(\frac{x}{T_y} + \frac{1}{n} \right) \right\rfloor \leq n \left(\frac{x}{T_y} + \frac{1}{n} \right) \quad (4.3)$$

and thus

$$\lim_{n \rightarrow \infty} P_n(x) = \frac{x}{T_y} = P(x). \quad (4.4)$$

We conclude that, in the continuous case, time can be modeled by a continuous random variable \mathbf{t} with uniform distribution in the interval $[0, T_y]$. The same argument holds for any number of periods of the input signal y . According to Definition 11 and Theorem 2.1.1 this completely specifies the experiment and the RV for modeling time. Given the result above, we can now understand the signal $\mathbf{y} = f(\mathbf{t})$ by means of a nonlinear transformation applied over the uniform distribution RV \mathbf{t} . This is a standard result from probability theory given by Eq. (4.5)

$$p_y(y) = \sum_k \left(\frac{p_x(x_i)}{\left| \frac{dy}{dx_i} \right|} \right). \quad (4.5)$$

where the derivative is evaluated at $x = g^{-1}(y)$. Considering $g(x)$ to be many-to-one, then its inverse will have multiple roots x_i .

To give the reader a better understanding of the modeling process, let's consider a signal $y = A_0 \sin(2\pi f_0 t)$. As stated above, time can be modeled by a RV with uniform distribution with

probability density function $p_t(t) = f_0$ on the interval $[0, 1/f_0)$. We consider $t = \frac{1}{f_0}$ to be the first instant of the next period and thus y has two roots in this interval, $t = 0$ and $t = \frac{1}{2f_0}$. We also have $\frac{dy}{dt} = A_0 2\pi f_0 \cos(2\pi f_0 t)$ and we evaluate the derivative at $t = \frac{1}{2\pi f_0} \sin^{-1}(\frac{y}{A_0})$. We note that $\cos \theta = \sqrt{1 - \sin^2 \theta}$ and after some manipulation we get

$$p_y(y) = \frac{1}{\pi \sqrt{A_0 - y^2}}, \quad (4.6)$$

in the interval $(-A_0, A_0)$, that is, a sine signal can be modeled by means of an arcsine distribution [19]. It is important to note that this analysis is valid for any frequency as time is abstracted in the modeling process. Left side signals shown in Fig. 4.1 illustrate this relationship.

4.2 Quantizer Design

To define a N -level quantizer, we need to find a set of N output values and $N - 1$ threshold levels, th_n . Fig. 4.2 shows the characteristic curve for a 4-level quantizer. We note that the quantizer output levels will be taken directly to be the UT sigma points S_i .

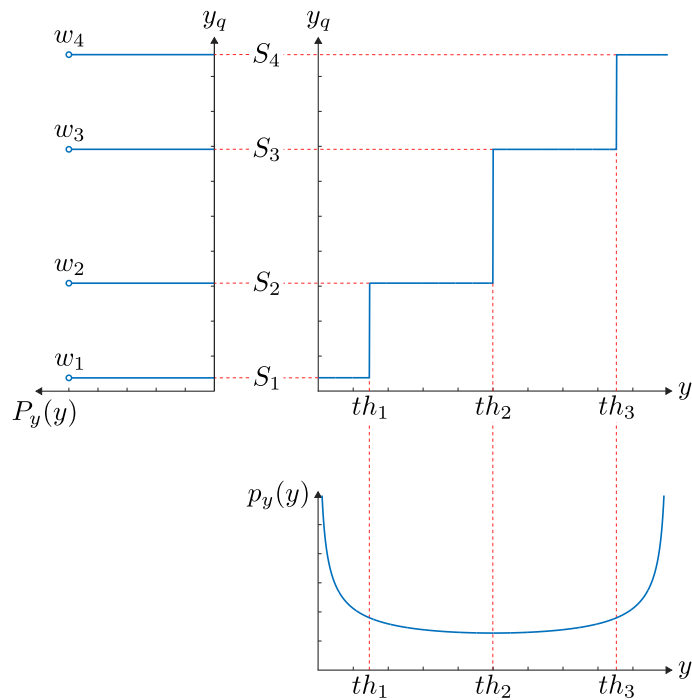


Figure 4.2: Characteristic curve for a 4-level arcsine distribution quantizer. The input y with arcsine distribution, $p_y(y) = \frac{1}{\pi \sqrt{y-y^2}}$ models a sinusoidal input signal in the interval $[0, 1]$. The quantizer maps the input distribution onto the output y_q according to Eq. (4.8) to give the sigma points S_i with weights w_i according to Eq. (2.27).

For the threshold levels we need to guarantee that the probability of finding the input signal

in the interval $[th_{n-1}, th_n]$ is the same probability w_n associated to the sigma-point S_n , that is:

$$\begin{aligned} P(th_{n-1} < x < th_n) &= \int_{th_{n-1}}^{th_n} p_x(x) dx = w_n \\ P_x(th_n) - P_x(th_{n-1}) &= w_n \end{aligned} \quad (4.7)$$

with $P_x(x) = \int_{-\infty}^x p_x(\tau) d\tau$ being the cumulative probability function associated with $p_x(\mathbf{x})$ in the interval $[a, b]$ and $t_0 = a$. By solving Eq. (4.7) for the general case we find that the threshold levels th_n will be located in

$$th_n = P_x^{-1} \left(\sum_{i=1}^n w_i \right) \quad (4.8)$$

where $P_x^{-1}(P_x(x)) = x$ is known as the quantile function. The same result holds for input probability functions with infinite support by considering that the quantizer with $[a, b]$ input range will map any input value below a to S_1 and any value above b to S_N .

The design of the moment preserving quantizer based on the UT can thus be summarized as follows:

1. Given an input signal, $y = f(t)$, use Eq. (4.5) to model the input signal as a probability density function $p_y(y)$;
2. Compute the UT of $p_y(y)$ to get the sigma-points and weights $\{s_i, w_i\}$;
3. Use Eq. (4.8) to design the threshold levels of the quantizer;
4. Use the sigma-points s_i as the output levels of the quantizer.

4.3 The Analysis of the Arcsine Quantizer

In this section, we analyse a quantizer based on the arcsine distribution to be implemented in a data conversion system from a signal processing perspective. We consider a single tone input signal y with unit amplitude resulting in an arcsine distribution (Eq. (4.6)) quantizer. We compare the arcsine distribution quantizer with the uniform quantizer characterized by equally spaced transition points [43]. To avoid any confusion with the uniform distribution quantizer, we will refer to the uniform quantizer as the linear quantizer.

First, we show in Figure 4.3 the input-output characteristic curves for the linear quantizer and for the arcsine distribution quantizer. The linear quantizer is characterized by constant quantization steps. We note that the arcsine quantizer have smaller quantization steps and thus finer resolution on the extrema of the input interval. On the other hand, it presents bigger quantization steps and thus coarser resolution on the center of the input dynamic range.

Figure 4.4 shows a transient simulation to show the effect of different quantization schemes on the output signals y_q . We note that the effect of the nonlinearity of the arcsine distribution quantizer is to provide a better fitting quantized signal where the sinusoidal input is slower and

thus presents higher probability of happening according to Eq. (4.6). This behavior resembles a variable resolution quantization scheme presented by Tsividis [44], but in our case we maintain the same number of quantization levels while improving error performance according to a statistical criterion.

Also, in Figure 4.4 we present the quantization error defined by $e_q = y - y_q$. For the linear quantizer we note the classical sawtooth plus bell shape [45]. The arcsine distribution quantizer, however, presents a modulated behavior where we have smaller quantization error near the input signal extrema and more error near the input zero crossings. In the sine case, this means the quantizer is opting to introduce less error where the signal presents slower variation and thus we can intuitively expect to have less quantization noise for lower frequencies.

For the arcsine quantizer, we can derive an analytical expression for the frequency spectrum. In the interval $[-1, 1]$, the UT sigma points will be the Chebyshev nodes given by

$$S_i = \cos\left(\pi \frac{2i-1}{2m}\right) \quad (4.9)$$

where m is the total number of sigma points. All points will have the same probability

$$w_i = \frac{1}{m}. \quad (4.10)$$

Applying Definition 16 in Eq. (4.6), the probability function for the arcsine distribution, in the interval $[-1, 1]$ will be given by

$$P_x(x) = \frac{2}{\pi} \sin^{-1}\left(\sqrt{\frac{x+1}{2}}\right) \quad (4.11)$$

and we get the inverse probability function

$$P_x^{-1}(y) = 2 \sin^2\left(\frac{\pi}{2}y\right) - 1. \quad (4.12)$$

The threshold levels for an m -level arcsine quantizer, given by Eq. (4.8) will be given by

$$th_n = 2 \sin^2\left(\frac{\pi}{2} \frac{n}{m}\right) - 1 \quad (4.13)$$

where $n = 1, \dots, m-1$.

Finally, we can calculate the transition instants of the quantized signal y_q , i.e., the time instants

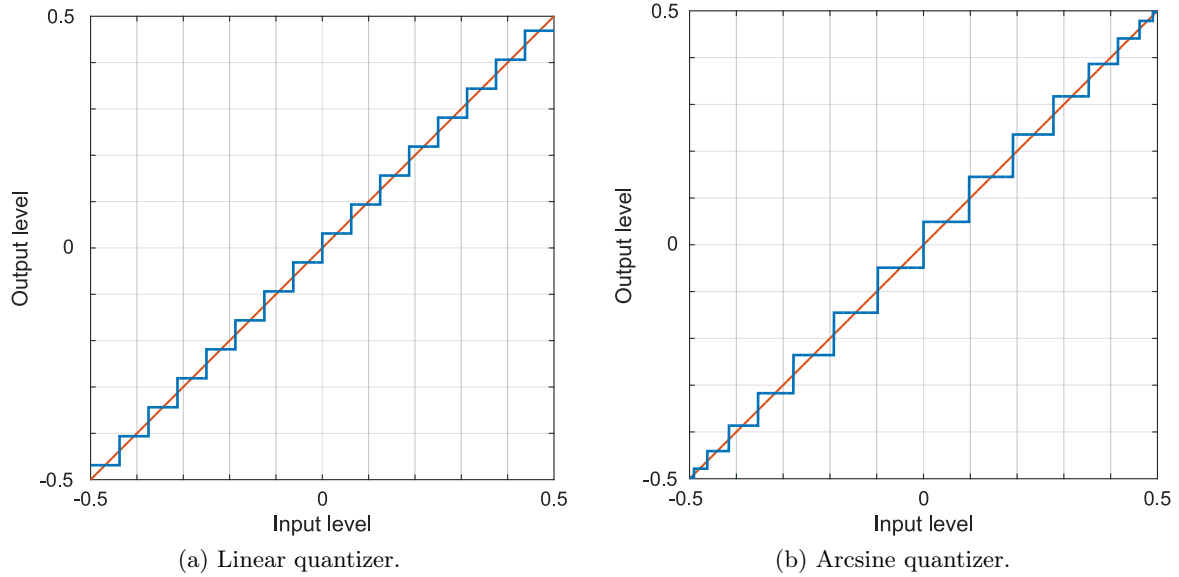


Figure 4.3: Characteristic curves for the linear and arcsine quantizers. The linear quantizer is characterized by a constant quantization step and the arcsine quantizer presents smaller quantization steps near the input signal extrema.

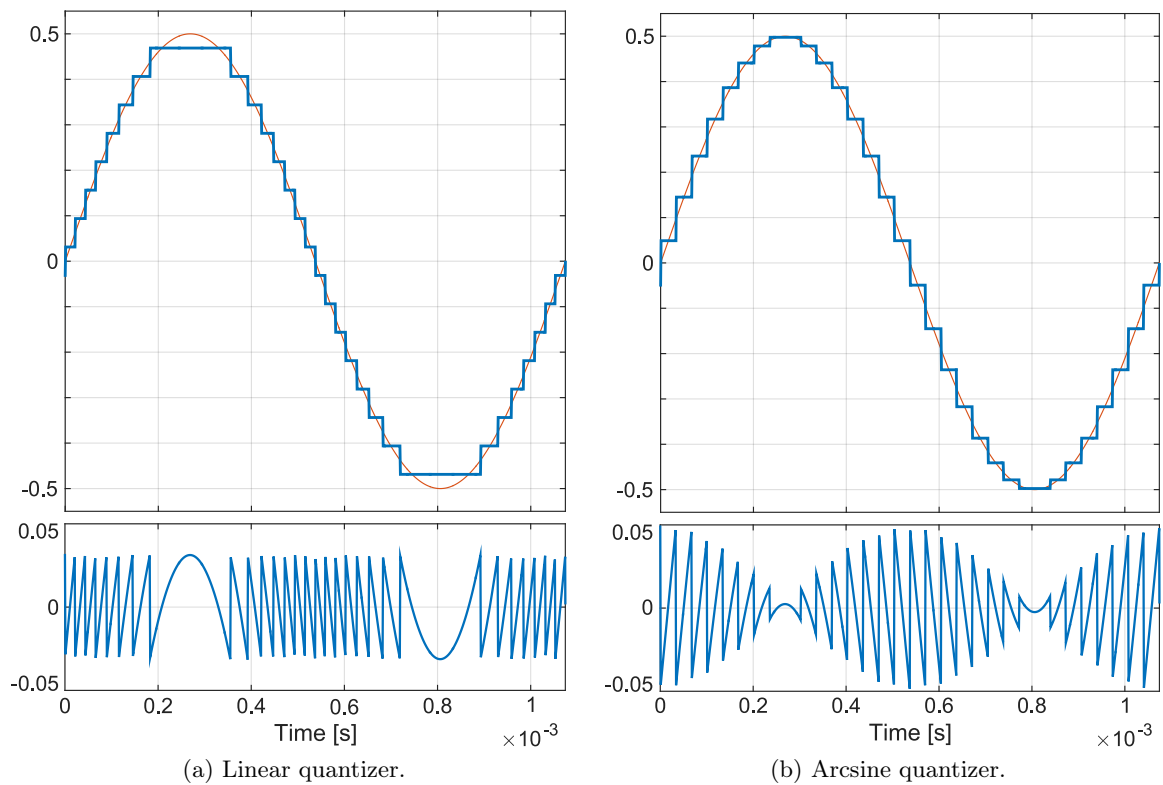


Figure 4.4: Transient analysis for a single tone input signal processed by the linear and by the arcsine distribution quantizers showing the quantization error. In 4.4a we get the classical bell/sawtooth quantization error pattern. In 4.4b we note that the quantization error is smaller at the signal extrema due to the smaller quantization steps. We note that the smaller error happens where the input signal presents slower variations.

where we get transitions on the output levels. As we are considering a signal $y = \sin(t_n)$, we get

$$\begin{aligned}
t_n &= \sin^{-1}(th_n) \\
&= \sin^{-1}\left(2 \sin^2\left(\frac{n\pi}{2m}\right) - 1\right) \\
&= \sin^{-1}\left(2 \left(\frac{1 - \cos(n\pi/m)}{2}\right) - 1\right) \\
&= \sin^{-1}\left(\cos\left(\frac{n\pi}{m} - \pi\right)\right) \\
&= \sin^{-1}\left(\sin\left(\frac{n\pi}{m} + \frac{\pi}{2} - \pi\right)\right) \\
&= \frac{n\pi}{m} - \frac{\pi}{2}
\end{aligned} \tag{4.14}$$

for t_n in the normalized interval $[-\pi/2, \pi/2]$. We conclude that the instants t_n are equally spaced. The last result suggests that the spectrum of the quantized signal will resemble the one of a flat top pulse amplitude modulation and we expect to find a continuous spectrum given by

$$S(f) = f_s \sum_{n=-\infty}^{\infty} M(f - kf_s)H(f) \tag{4.15}$$

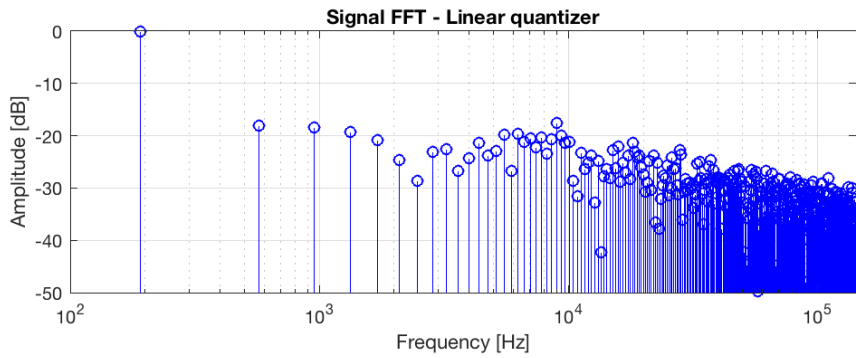
where $f_s = 2f_0 \times 2^N$ is the transition rate of the quantized signal, $M(f)$ is the Fourier Transform of the input signal and $H(f) = T \text{sinc}(fT) \exp(-j\pi ft)$ is the Fourier Transform of the rectangle function with pulse width $T = 1/(f_s)$ [46]. In the case of a sinusoidal input signal, Eq. (4.15) gives us a spectrum with an impulse at f_0 , the signal frequency, with double impulses centered at nf_s and spaced by $2f_0$ as shown in Figure 4.5b for a signal with $f_0 = 190.7\text{Hz}$ and a 4-bit quantizer. We used such a low resolution to exaggerate the visual aspects of the discussion. The analysis of the linear quantizer is more complex and we refer to the work of Abidi [47] and the classical works of Bennett [45] and Widrow [10] for more detailed descriptions.

Figure 4.5a presents the spectrum for the linear quantizer. We note that in the arcsine quantizer the relevant harmonics appear in a much higher frequency than in the linear quantizer suggesting that we can get a cleaner frequency spectrum after the sampling process in the ADC.

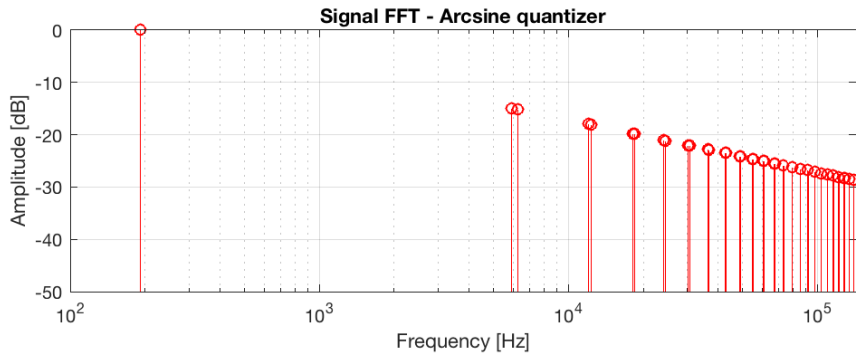
Actually, in the quantization error spectrum we observe this same behavior, as shown in Figure 4.6. Abidi [47] suggests that the first harmonics seen in Figure 4.6a are due to the bell portion of the quantization error seen in Figure 4.4a. We also note that the quantization error have reduced power in the input signal frequency compared to the linear quantizer.

We analyze next the spectral behavior of a sampled version of the quantized signal. As indicated in Eq. (4.15), the PAM signal has infinite band and any sampling frequency we choose will cause aliasing to make the higher harmonics to appear in the observed band as noise [48]. Also, we note that the quantization error for the linear and for the arcsine quantizers are correlated to the input signal. To improve the analysis of the spectral components of higher order ADCs, we introduce 1/3 LSB gaussian dither to the input signal [49][50][51].

Figure 4.7 shows the output spectra for a single tone input with frequency $f_0 = 999.1\text{Hz}$ and sample frequency $f_s = 100\text{kHz}$ for a system with $f_{3dB} = 1\text{MHz}$ bandwidth. We note the



(a) Linear quantizer output spectrum.



(b) Arcsine distribution quantizer output spectrum.

Figure 4.5: Spectral analysis for a single tone input frequency of $f_0 = 197.7\text{Hz}$ quantized by the 4-bit linear and the 4-bit arcsine quantizers.

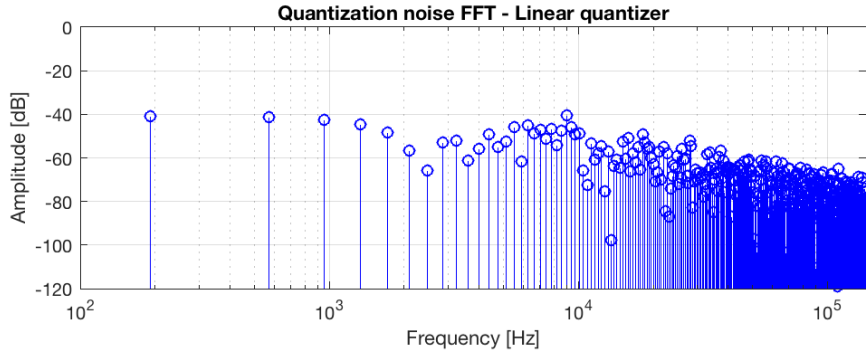
reduced harmonic components in the arcsine quantizer spectrum suggesting we can get a better reconstructed signal with a low pass filter.

Figure 4.8 show the signal-to-noise ratio – SNR – and the signal to noise and distortion ratio – SINAD – for increasing resolution along with the theoretical $SNR = 6.02N + 1.76\text{dB}$ result. We note that the arcsine distribution quantizer does not improve the SNR and SINAD.

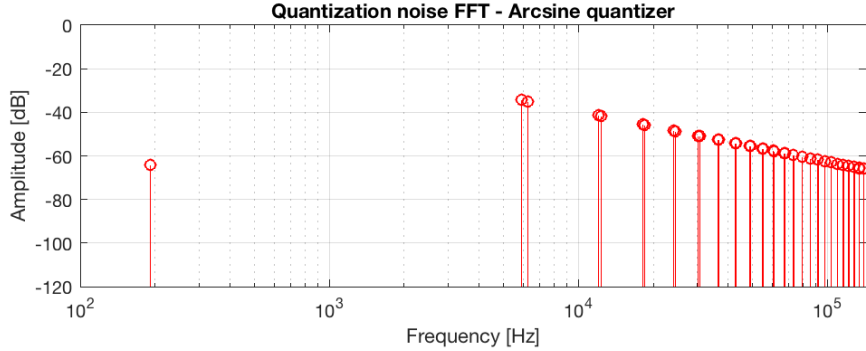
Nevertheless, Figure 4.9a show the total harmonic distortion – THD – taken to the 10th harmonic for increasing resolution. We note an improvement of about 16dB, which means about 3 bits improvement, in the THD in the studied resolution range. We also note that this improvement is quite insensitive to the resolution of the quantizer.

Finally, in Figure 4.9b we present the Spurious Free Dynamic Range of the quantizers along with the theoretical $SFDR = 9.03n$ result by Blachman [52] and Abidi [47]. We note an improvement of about 14dB for lower resolutions and a trend to lower this difference for higher resolutions.

These results indicate that we would better interpolation quality at the end of the processing chain due to lower distortion caused by the quantization process.



(a) Linear quantizer quantization error spectrum.



(b) Arcsine distribution quantizer quantization error spectrum.

Figure 4.6: Quantization error spectral analysis for a single tone input frequency of $f_0 = 197.7\text{Hz}$ quantized by the 4-bit linear and the 4-bit arcsine quantizers.

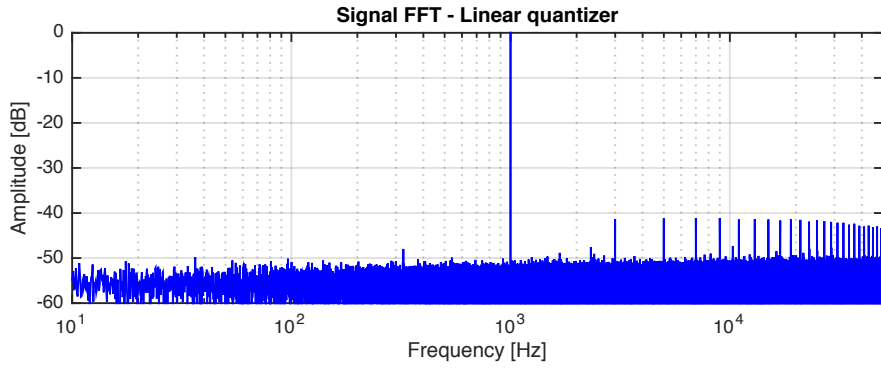
4.4 Circuit Implementation Proposal

In this section, to get the circuit implementation inclined reader a better understanding on what the threshold and output levels presented in Eq. (4.8) impact on circuit design, we present an implementation proposal based on a parallel architecture.

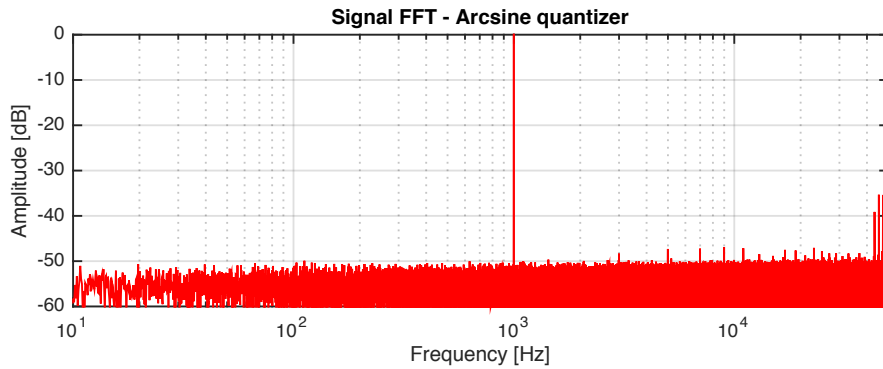
Figure 4.10 shows the system block diagram including the analog-to-digital and the digital-to-analog converters. The ADC can be implemented by a voltage/current scaling network in which the input dynamic range is properly processed according to the th_n threshold levels to give a set of $N - 1$ voltage/current levels that will be compared to the input signal. This process gives a binary thermometer code that can be properly encoded [43]. The DAC can be implemented using such the same scheme. A voltage/current scaling network takes the S_i output levels and process the input dynamic range to give a set of N voltage/current levels that can be selected by a proper decoder.

Figure 4.11a shows a conceptual circuit schematic of a parallel or flash ADC. In such a circuit, a resistive network implements the voltage scaling part. In such a circuit, for a N -level quantizer, the resistors values will be given by

$$R_n = th_n \times R - \left(\sum_{k=0}^{n-1} R_k \right), \quad (4.16)$$

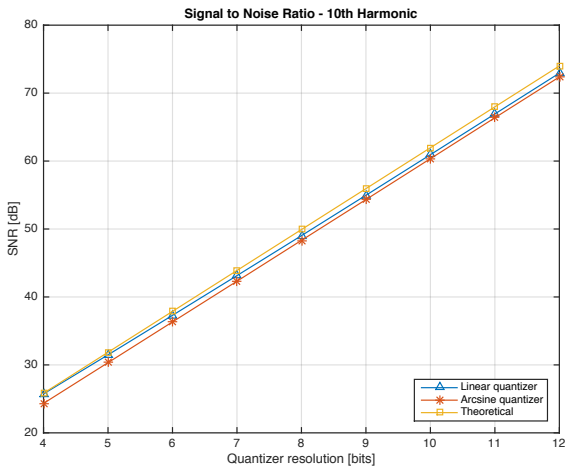


(a) Linear quantizer output spectrum.

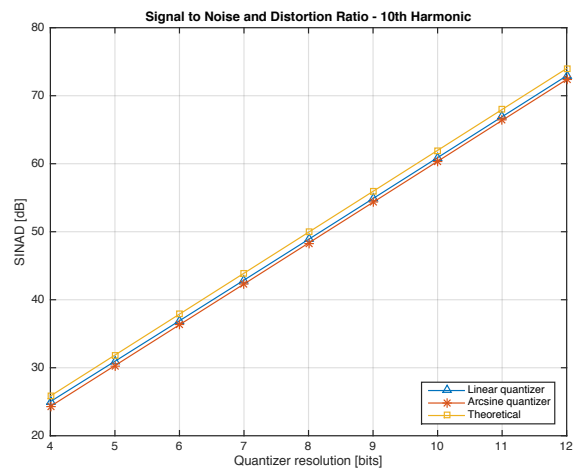


(b) Arcsine distribution quantizer output spectrum.

Figure 4.7: Spectral analysis for a single tone input frequency of $f_0 = 999.1\text{Hz}$ quantized by the 10-bit linear and the 10-bit arcsine quantizers.



(a) Signal to noise ratio – SNR.

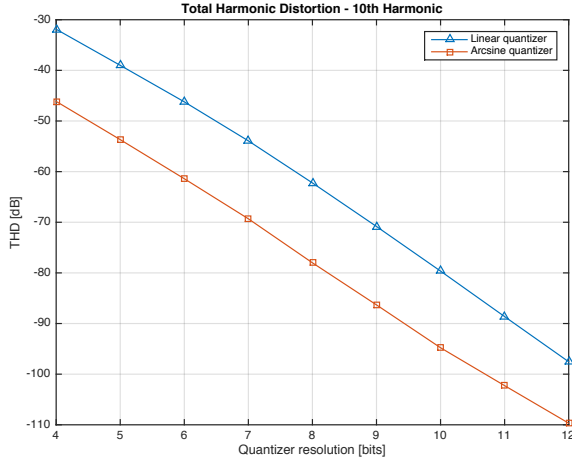


(b) Signal to noise and distortion ratio – SINAD.

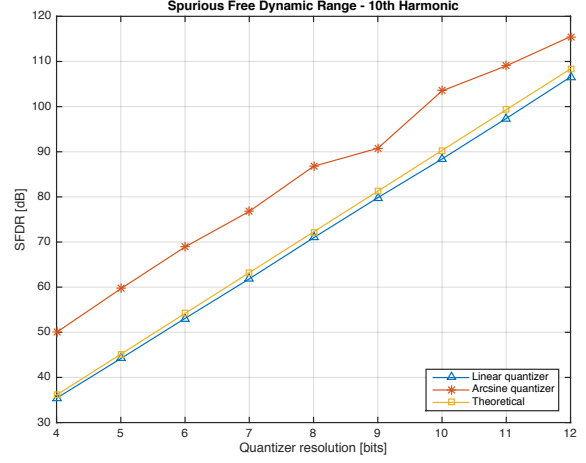
Figure 4.8: SNR and SINAD for increasing quantizers resolution.

for every $n = 1, 2, \dots, \dots, N$, where R is an arbitrary resistor scale factor, $R_0 = 0$ and $th_N = 1$.

Figure 4.11b shows a conceptual circuit schematic of a parallel or flash DAC. In this circuit, a resistive network implements the scaling function but here by considering the desired output levels

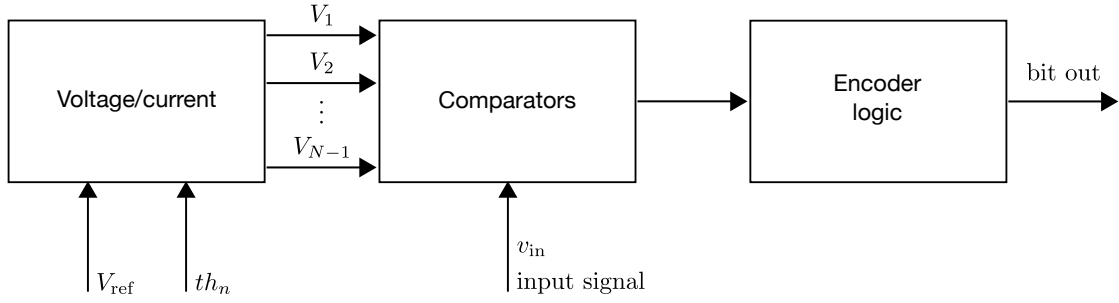


(a) Total harmonic distortion, taken up to the 10th harmonic.

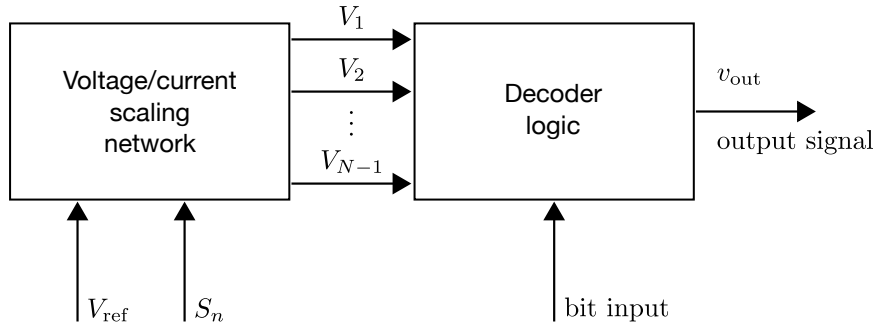


(b) Spurious free dynamic range.

Figure 4.9: THD and SFDR for increasing quantizers resolution.



(a) ADC block diagram.



(b) DAC block diagram.

Figure 4.10: ADC and DAC block diagrams showing the influence of the thresholds th_n and output points S_n given by Eq (4.8).

S_i . In such a circuit, for a N -level DAC, the resistor values will be given by

$$R_n = S_n \times R - \left(\sum_{k=0}^{n-1} R_k \right), \quad (4.17)$$

for every $n = 1, 2, \dots, \dots, N + 1$, where R is an arbitrary resistor scale factor, $R_0 = 0$ and $S_{N+1} = 1$. Despite their conceptual simplicity, flash topologies suffer from a number of drawbacks due to massive parallelism in its implementations that are out of the scope of this paper [43].

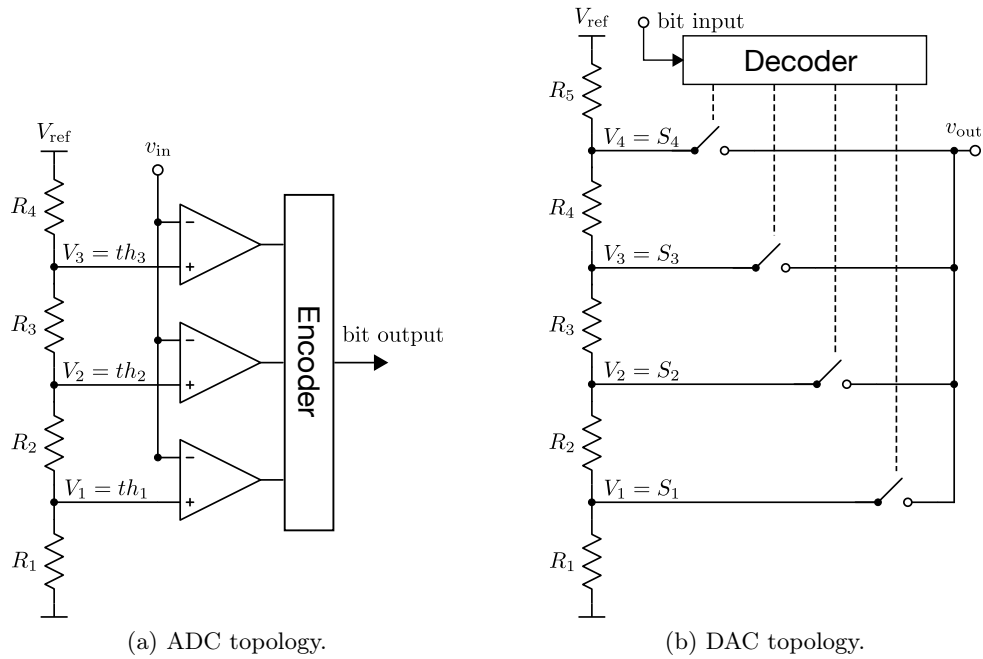


Figure 4.11: Schematic ADC and DAC circuit topologies based on parallel architectures for the implementation of the moment preserving quantizers.

4.5 Chapter Final Remarks

This chapter presented a design flow for signal specific nonlinear quantizers based on the UT. From a periodic signal, we show how to derive a probability function abstracting the time variable. It follows with the computation of the UT as discussed in Chapter 2. The design finishes with the derivation of the threshold levels and output values of the quantizer. It follows with the analysis of the arcsine quantizer which is based on the sinusoidal input signal. The chapter finishes showing that the threshold levels influence the ADC design and the output values are used on the DAC to recover the input signal.

Chapter 5

Related Work

This section presents a non-exhaustive survey of applications of the Unscented Transform. The goal is to give a critical overview about the state-of-the-art and provide means to better position the contributions of this thesis. We tried to stick to indexed journal papers as much as possible as they indicate more mature research and results but we don't exclude peer-reviewed conference proceedings papers when the subject is considered to be relevant to our discussion.

5.1 Filtering

The Unscented Transform was first proposed by Jeffrey Uhlmann back in 1995 [53]. In this work, the authors identify at least three well-known issues with the Extended Kalman Filter – EKF – that was at the time the standard tool to estimate the state of a system with nonlinear dynamics from noisy sensor information: 1) linearization on the filter requires sufficiently small timesteps; 2) small timesteps, on the other hand, imply high computational effort demanded for the generation of the Jacobian matrix on the filter and; 3) the derivation of Jacobian matrices is nontrivial for many applications. The authors then propose to approximate continuous probability functions by a set of discrete points and process them with the nonlinear dynamics model instead of linearizing the system model and process its effect on the continuous probability functions. In [54] Julier details the approach and provides the first analysis for the approximation of higher-order moments. The name Unscented Transformation came around 2004 with a much more detailed analysis of the transformation itself and of the Unscented Kalman Filter – UKF – [17].

Since the UKF filter is easier to implement while providing better estimation performance compared to the EKF, a plethora of applications were proposed since then. Farina et al. compare the performance of nonlinear filters including the UKF on the problem of tracking a ballistic object in reentry phase by processing radar measurements [55]. Jing Li et al. employ UKF filters with genetic algorithms in an in-flight alignment scheme for aircrafts inertial navigation system [56]. Garcia-Fernandez et al. employ the UKF for an active driving safety system based on automotive embedded sensors [57]. Nguyen and Nestorović employ the UKF as a tool for accelerating convergence of algorithms for the estimation of seismic waves [58].

It is interesting to note that it took some time to establish the UT as a form of quadrature and some recent published papers still don't treat it as such. Ito and Xiong showed a method to compute the UKF sigma-points based on the Gauss-Hermite quadrature, a special case of the Mechanical quadrature presented in Chapter 2 for Gaussian distributions [59]. Later, Daum still describes the UKF algorithm as an "approximation of multidimensional integrals using the unscented transform" even though he describes other nonlinear filter algorithms as "numerical integration" methods [60]. An attempt of systematization of different formulations of the UKF is presented by Menegaz et al. [61]. In this work, the authors classify the choice of sigma-points in numerical quadrature methods and Monte Carlo based methods. Also in this work, the authors present an extensive survey of UKF proposals showing that the use of moments of order higher than 3th is rarely employed on such systems.

5.2 Probability Functions Estimation

The Unscented Transform gained importance by itself out of the scope of the UKF and several applications based on estimating the statistical characteristics of a random variable over a nonlinear transformation. In this way, the UT is proposed as an alternative to the use of Monte Carlo Methods.

Menezes et al. presents a UT based approach for both time and frequency domain electromagnetic simulations [21]. The authors present a combination of preprocessing and post-processing stages in which the UT is used to choose parameters for a model that is evaluated by a standard simulator. Carneiro et al. combines uncertainty estimation via UT with a robust genetic circuit optimizer for the circuit level design of a Doherty amplifier [62]. Erlandson and Niklasson propose to use the UT to estimate a fighter aircraft combat survivability, i.e., the probability that the aircraft can fly a route inside hostile territory without getting hit by enemy fire [63]. Daigle et al. propose an adaptive prediction algorithm in prognostics analysis [64]. The author present a lithium-ion battery case study in which end-of-discharge is predicted from uncertain data using the UT. Menezes et al. present the UT as an alternative to Monte Carlo methods for the estimation of bit error rates in communication systems [40]. Let al. propose an asymmetric choice for the UT sigma points as a way to improve the UT performance to estimate remaining useful life – RUL – of aircraft systems from real data [65].

Some authors propose similar techniques, even though they don't explicitly claim their methods to be based on the UT. Nigam et al. present a sampling based method similar to the approach discussed in Chapter 3 to characterize integrated circuits [66]. However, their sampling scheme is based on uniformly spaced sigma-points which is known to present slower convergence than points selected according to a Gaussian quadrature rule as we propose. Gong et al. also analyze the statistical behavior of analog circuits by sampling probability distributions [67]. The authors propose an algorithm similar to the direct method presented in Section 2.2.1 for the selection of sigma-points.

5.3 Quantization

Delp and Mitchell present the generalization of the quantization scheme proposed for the block truncation coding – BTC. In BTC, a digital image is divided into $n \times x$ blocks and the pixels in each block are processed by a quantization function that preserves the first and second sample moments of the block [68]. Fig. 5.1 shows the BTC quantization process. Later, the authors generalize the design of the BTC moment preserving quantizers – MPQ – to an arbitrary number of moments [69]. Even though the UT had not been proposed yet at the time, the authors moment

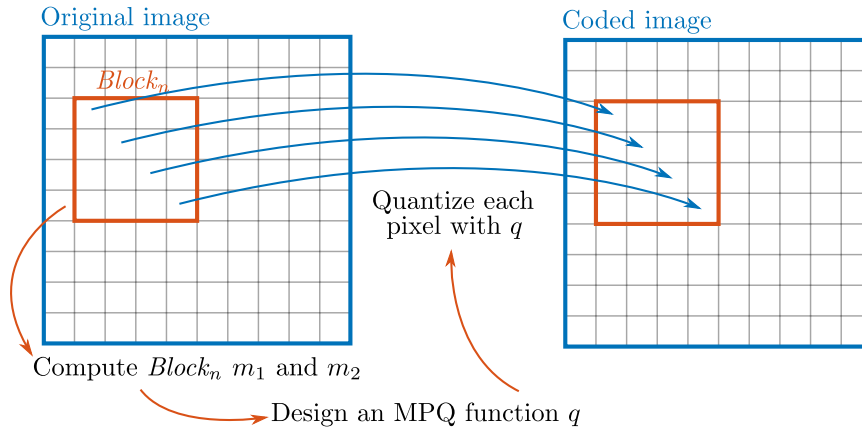


Figure 5.1: The BTC coding scheme is based on moment preserving quantizers.

preserving quantizers have similar properties to those developed in this thesis but the application differs. Note in Fig. 5.1 that each pixel in the original image is already quantized in the sense defined in Chapter 1, i.e., each pixel is represented by a discrete range. Quantization in the context of BTC means representing each pixel with a reduced number of bits compared to the original image. In this work, we develop moment preserving quantizers for data converter applications, i.e., our input signals have a continuous range and the quantized output will present a discrete range. To the best of our knowledge, the first work to propose the use of the UT in the design of quantizers was [42] and Chapter 4 of this thesis provides an extended discussion.

Chapter 6

Conclusion

Data converters are crucial components of the digital signal processing chain serving as the interfaces between the analog and the digital domains. Advancements in digital technology push the requirements of data converters as applications require faster and more accurate data converters. Also, the integration of analog front-ends with the digital back-end means that the analog-to-digital converters should be implemented in the newest CMOS technology process nodes. This presents new challenges for the advancement of data converters architectures. In this work we propose the use of the Unscented Transform as a framework for the design of quantizers. Such technique present the advantage of preserving the higher-order statistical moments of the input signal in the quantized output.

This thesis presented a definition of the Unscented Transform theory. To the best of our knowledge, this is the first attempt to express the UT in such a formal way. The proposed definition is sufficiently abstract to encompass many other proposals found in the literature. We show that the UT can be understood as a form of the classical mechanical quadrature and we provide the relevant theorems that justify the practical implementations shown.

The Unscented Transform has been shown to be an effective tool for the statistical analysis of nonlinear mappings in different applications. We presented the advantages of our formulation for the UT as an alternative to Monte Carlo methods. We also presented the Extended Unscented Transform as an extension for the UT theory as a way to increase the number of sigma points and improve the probability function approximation given by the UT. Even though our technique has exponential complexity, it was the drive to search for a more sound mathematical definition and it allows a more straightforward path to parallel implementations as we do not rely on random number generators.

We proposed to explore *a priori* information about an input signal to improve the design of the quantization process in data converters. The goal for our optimization criterion is to preserve as many statistical moments as possible. For this, we presented the Unscented Transform as a mathematical framework for the design of moment preserving quantizers. We established the Unscented Transform as a new mathematical formulation for quantization processes. From this point of view, we could show that the UT, the quantization process, and the numerical quadrature

theories are strongly linked.

Finally, we proposed the design procedure of a class of moment preserving quantizers. We analyze a case study which supports the evidence that our approach does not alter the amount of quantization error introduced by a quantizer of a given resolution but, in fact, distributes the distortion over higher frequency ranges when compared to the linear quantizer. Besides, our approach guarantees that we preserve the statistical moments of the analog input signal into the digital domain, a useful feature for applications based on the moments estimation such as the blind source separation applied to biomedical signals analysis. We also presented an initial proposal of circuit topology for implementation of such systems to be further detailed.

6.1 Future Research

Such a novel proposal poses many more questions than what can be answered during the development of a single PhD thesis. At first, the naive Monte Carlo method is rarely used in real world applications because of its slow convergence. Importance sampling and quasi-Monte Carlo are the actual techniques known to be more efficient to deal with high dimension problems. However, these techniques' performance for higher order moments estimation is not well understood and constitute an open research area. Also, a detailed comparison between importance sampling, quasi-Monte Carlo and the UT approach would be of interest to the research community.

Regarding the design of moment preserving quantizers based on the UT, we tried to determine if the linear quantizer is a special case of the moment preserving quantizer. We did not succeed in such an effort but we believe we did a step towards the answer by analyzing the UT as a special case of the mechanical quadrature. We conjecture that the answer is no, the linear quantizer is not a special case of the UT quantizer, but we could not find a proof for our intuition.

Also, we could not find a systematic way to design UT quantizers based on infinite support probability distributions, i.e., distributions in which the $[a, b]$ interval of definition is such that $a = -\infty$ or $b = \infty$. The normal and exponential distributions are examples of those. Such distributions imply the designed quantizer will have a potentially infinite dynamic range which is infeasible for physical implementation. We suggest future research to explore truncated distributions as a first step to solve such an issue.

Gaussian quadrature is optimal for the UT formulation in the sense that one gets the most moments for a fixed number of sigma-points but it is not the only contender in the realm of quadrature methods. Besides the Gaussian quadrature, the Clenshaw-Curtis quadrature seems to be a viable alternative for the UT formulation and we suggest future research to take advantage of the formal definition we provide on this thesis to support further developments using other quadrature formulations.

Additionally, we presented an integrated view between quantization and the UT. In data converters applications, when converting back from the digital to the analog domain, it is common to have an analog low-pass filter acting as an interpolation filter. Such filter could inspire new

algorithms for the probability function estimation discussed in Chapter 3 based on classical signal processing filters.

6.1.1 Publications

We present the Extended Unscented Transform approach in paper [35]. As discussed, the ability to compute more sigma-points has a two-fold importance in our work. First, it allows one to better characterize probability distributions on applications where the UT substitutes Monte Carlo methods. Second, in designing moment preserving quantizers, we need as many sigma-points as there are quantization levels on the final system. In the paper we present an Extended UT approach and a case study on characterization of circuits to prove its applicability.

In [42] we present the main contribution of this thesis, the procedure to design quantizers based on the Unscented Transform. Since the UT models a continuous probability function by means of a discrete one, we developed a design flow for using the UT as a framework for the design of quantizers for data converters. In the paper we present the main concept, we derive the design equations and present the analysis of the arcsine quantizer.

Paper [70] presents a hearing aid front-end based on the wavelet transform. The paper presents a system composed of an analog wavelet filter bank and an Automatic Gain Control (AGC), with a new topology for the decision logic and a new circuit design for the Programmable Gain Amplifier (PGA). Validation is achieved by means of circuit level simulations.

An asynchronous sampler based on the wavelet transform is presented in paper [71]. The paper presents a novel analog-to-digital converter architecture based on the use of Wavelet Transforms to localize critical points in the input signal. The estimated Lipschitz exponent describes the signal structure. Also, the paper presents a polynomial reconstruction algorithm for such a system.

Paper [72] presents a framework for assessing the use of off-the-shelf micro controllers as Physical Unclonable Functions – PUFs –. A PUF is an entity that explores physical structures to provide a function that is easy to evaluate but hard to predict and duplicate. It is the hardware analog of a one-way function. SRAM structures can be used as PUFs and the paper presents the characterization and analysis of an off-the-shelf microcontroller with embedded memory to assess its use as a practical implementation for PUFs.

REFERENCES

- [1] PELGROM, M. J. *Analog-to-digital conversion*. New York, NY: Springer, 2013. 325–418 p.
- [2] TSIVIDIS, Y. Continuous-time digital signal processing. *Electron. Lett*, v. 39, n. 21, p. 1551–1552, 2003.
- [3] NYQUIST, H. Certain topics in telegraph transmission theory. *Transactions of the American Institute of Electrical Engineers, IEEE*, v. 47, n. 2, p. 617–644, 1928.
- [4] SHANNON, C. E. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, ACM, v. 5, n. 1, p. 3–55, 2001.
- [5] MAX, J. Quantizing for minimum distortion. *IRE Transactions on Information Theory, IEEE*, v. 6, n. 1, p. 7–12, 1960.
- [6] WIDROW, B.; KOLLAR, I.; LIU, M.-C. Statistical theory of quantization. *IEEE Transactions on Instrumentation and Measurement*, New York: Institute of Electrical and Electronics Engineers., v. 45, n. 2, p. 353–361, 1996.
- [7] LLOYD, S. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, v. 28, n. 2, p. 129–137, mar 1982. ISSN 0018-9448.
- [8] ROSA, J. M. de la. Sigma-Delta Modulators: Tutorial Overview, Design Guide, and State-of-the-Art Survey. *IEEE Transactions on Circuits and Systems I: Regular Papers*, v. 58, n. 1, p. 1–21, jan 2011. ISSN 1549-8328, 1558-0806.
- [9] SANTOS, M.; HORTA, N.; GUILHERME, J. A survey on nonlinear analog-to-digital converters. *Integration, the VLSI Journal*, Elsevier, v. 47, n. 1, p. 12–22, 2014.
- [10] WIDROW, B. Statistical analysis of amplitude-quantized sampled-data systems. *Transactions of the American Institute of Electrical Engineers, Part II: Applications and Industry, IEEE*, v. 79, n. 6, p. 555–568, 1961.
- [11] CHEDED, L. Exact recovery of higher order moments. *IEEE Transactions on Information Theory*, v. 44, n. 2, p. 851–858, mar 1998. ISSN 00189448.
- [12] CHEDED, L.; PAYNE, P. A. The exact impact of amplitude quantization on multi-dimensional, high-order moments estimation. *Signal Processing*, v. 39, n. 3, p. 293–315, sep 1994. ISSN 01651684.

- [13] GIANNAKIS, G. B.; TSATSANIS, M. K. Signal detection and classification using matched filtering and higher order statistics. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, v. 38, n. 7, p. 1284–1296, Jul 1990. ISSN 0096-3518.
- [14] HASSAN, K. et al. Blind digital modulation identification for spatially-correlated mimo systems. *IEEE Transactions on Wireless Communications*, v. 11, n. 2, p. 683–693, February 2012. ISSN 1536-1276.
- [15] MILEOUNIS, G.; KALOUPSIDIS, N.; KOUKOULAS, P. Blind identification of hammerstein channels using qam, psk, and ofdm inputs. *IEEE Transactions on Communications*, v. 57, n. 12, p. 3653–3661, December 2009. ISSN 0090-6778.
- [16] MENDEL, J. M. *Use Of Higher-Order Statistics In Signal Processing And System Theory: An Update*. 1988. 0975 - 0975 - 19 p.
- [17] JULIER, S. J.; UHLMANN, J. K. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, IEEE, v. 92, n. 3, p. 401–422, 2004.
- [18] UHLMANN, J. K. *Dynamic map building and localization: New theoretical foundations*. Tese (Doutorado) — University of Oxford, 1995.
- [19] PAPOULIS, A.; PILLAI, S. U. *Probability, random variables, and stochastic processes*. New York, NY: Tata McGraw-Hill Education, 2002.
- [20] BILLINGSLEY, P. *Probability and measure*. New York, NY: John Wiley & Sons, 2008.
- [21] MENEZES, L. d. et al. Efficient computation of stochastic electromagnetic problems using unscented transforms. *IET Science, Measurement & Technology*, v. 2, n. 2, p. 88, 2008.
- [22] SHOHAT, J. A.; TAMARKIN, J. D. *The problem of moments*. Providence, Rhode Island: American Mathematical Soc., 1943.
- [23] AHEIZER, N. I.; KEMMER, N. *The classical moment problem and some related questions in analysis*. [S.l.]: Oliver & Boyd, 1965.
- [24] TABATABAI, A. J.; MITCHELL, O. R. Edge location to subpixel values in digital imagery. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, n. 2, p. 188–201, 1984.
- [25] JUNIOR, E. A. da C. *Propagação de incertezas em eletromagnetismo*. Tese (Doutorado) — University of Brasília, Brazil, 2009.
- [26] GAUTSCHI, W. Construction of gauss-christoffel quadrature formulas. *Mathematics of Computation*, JSTOR, v. 22, n. 102, p. 251–270, 1968.
- [27] SZEGO, G. *Orthogonal polynomials*. Providence, Rhode Island: American Mathematical Soc., 1939.
- [28] PONNUSAMY, S. *Foundations of mathematical analysis*. New York, NY: Springer Science & Business Media, 2011.

- [29] STOER, J.; BULIRSCH, R. *Introduction to Numerical Analysis*. New York, NY: Springer New York, 2002. 37–144 p.
- [30] OLVER, F. W. *NIST handbook of mathematical functions*. New York: Cambridge University Press, 2010.
- [31] GAUTSCHI, W. *Orthogonal Polynomials: Computation and Approximation*. Oxford, UK: Oxford Science Publications, 2004.
- [32] KALOS, M. H.; WHITLOCK, P. A. *Monte carlo methods*. Weinheim: John Wiley & Sons, 2008.
- [33] ROBERT, C.; CASELLA, G. *Monte Carlo statistical methods*. New York, NY: Springer Science & Business Media, 2013.
- [34] PEREYRA, M. et al. A survey of stochastic simulation and optimization methods in signal processing. *IEEE Journal of Selected Topics in Signal Processing*, IEEE, v. 10, n. 2, p. 224–241, 2016.
- [35] MEDEIROS, J. de; HADDAD, S.; MENEZES, L. de. Extended formulation for unscented transform and its application as Monte Carlo alternative. *Electronics Letters*, v. 52, n. 22, p. 1842–1843, oct 2016. ISSN 0013-5194.
- [36] CAFLISCH, R. E. Monte carlo and quasi-monte carlo methods. *Acta numerica*, Cambridge Univ Press, v. 7, p. 1–49, 1998.
- [37] HASTINGS, R. A. *The art of analog layout*. Upper Saddle River, NJ: Prentice Hall, 2006.
- [38] SINGHEE, A.; RUTENBAR, R. A. Why quasi-monte carlo is better than monte carlo or latin hypercube sampling for statistical circuit analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, IEEE, v. 29, n. 11, p. 1763–1776, 2010.
- [39] RAZAVI, B. *Design of analog CMOS integrated circuits*. New York, NY: McGraw-Hill, 2001.
- [40] MENEZES, L. D. et al. Using unscented transform as alternative to monte carlo in bit error rate calculations. *Electronics Letters*, The Institution of Engineering & Technology, v. 49, n. 10, p. 1, 2013.
- [41] HELLEKALEK, P. Don't trust parallel monte carlo! *SIGSIM Simul. Dig.*, ACM, New York, NY, USA, v. 28, n. 1, p. 82–89, jul. 1998. ISSN 0163-6103.
- [42] MEDEIROS, J. E. G.; HADDAD, S. A. P. Nonlinear quantizer design in data conversion systems using the unscented transform. In: *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. [S.l.]: IEEE, 2017. p. 1–4. ISBN 978-1-4673-6853-7.
- [43] RAZAVI, B. *Principles of data conversion system design*. Piscataway, NJ: IEEE press New York, 1995.

- [44] KURCHUK, M.; TSIVIDIS, Y. Signal-dependent variable-resolution quantization for continuous-time digital signal processing. In: *2009 IEEE International Symposium on Circuits and Systems*. [S.l.: s.n.], 2009. p. 1109–1112. ISSN 0271-4302.
- [45] BENNETT, W. R. Spectra of quantized signals. *Bell System Technical Journal*, Wiley Online Library, v. 27, n. 3, p. 446–472, 1948.
- [46] HAYKIN, S. *Communication systems*. New York, NY: John Wiley & Sons, 2008.
- [47] PAN, H.; ABIDI, A. A. Spectral spurs due to quantization in nyquist adcs. *IEEE Transactions on Circuits and Systems I: Regular Papers*, IEEE, v. 51, n. 8, p. 1422–1439, 2004.
- [48] KESTER, W. Mt-001: Taking the mystery out of the infamous formula," $\text{snr} = 6.02 n + 1.76$ db," and why you should care. *Analog Devices Tutorials*, 2005.
- [49] MOSCHITTA, A.; PETRI, D. Stochastic properties of quantization noise in memoryless converters affected by integral nonlinearity. *IEEE Transactions on Instrumentation and Measurement*, IEEE, v. 53, n. 4, p. 1179–1183, 2004.
- [50] KESTER, W. A. *Data conversion handbook*. Oxford, UK: Newnes, 2005.
- [51] KESTER, W. Mt-004: The good, the bad, and the ugly aspects of adc input noise: Is no noise good noise? *Analog Devices Tutorials*, 2005.
- [52] BLACHMAN, N. The intermodulation and distortion due to quantization of sinusoids. *IEEE transactions on acoustics, speech, and signal processing*, IEEE, v. 33, n. 6, p. 1417–1426, 1985.
- [53] JULIER, S. J.; UHLMANN, J. K.; DURRANT-WHYTE, H. F. A new approach for filtering nonlinear systems. In: *American Control Conference, Proceedings of the 1995*. [S.l.: s.n.], 1995. v. 3, p. 1628–1632 vol.3.
- [54] JULIER, S.; UHLMANN, J.; DURRANT-WHYTE, H. A new method for the nonlinear transformation of means and covariances in filters and estimators. *IEEE Transactions on Automatic Control*, v. 45, n. 3, p. 477–482, mar 2000. ISSN 00189286.
- [55] FARINA, A.; RISTIC, B.; BENVENUTI, D. Tracking a ballistic target: comparison of several nonlinear filters. *IEEE Transactions on Aerospace and Electronic Systems*, v. 38, n. 3, p. 854–867, jul 2002. ISSN 0018-9251.
- [56] LI, J. et al. In-flight initial alignment scheme for radar-aided SINS in the arctic. *IET Signal Processing*, v. 10, n. 8, p. 990–999, oct 2016. ISSN 1751-9675.
- [57] GARCIA-FERNANDEZ, A. F. et al. Bayesian Road Estimation Using Onboard Sensors. *IEEE Transactions on Intelligent Transportation Systems*, v. 15, n. 4, p. 1676–1689, aug 2014. ISSN 1524-9050.
- [58] NGUYEN, L. T.; NESTOROVIĆ, T. Unscented hybrid simulated annealing for fast inversion of tunnel seismic waves. *Computer Methods in Applied Mechanics and Engineering*, v. 301, p. 281–299, apr 2016. ISSN 00457825.

- [59] ITO, K.; XIONG, K. Gaussian filters for nonlinear filtering problems. *IEEE Transactions on Automatic Control*, v. 45, n. 5, p. 910–927, may 2000. ISSN 00189286.
- [60] DAUM, F. Nonlinear filters: beyond the Kalman filter. *IEEE Aerospace and Electronic Systems Magazine*, v. 20, n. 8, p. 57–69, aug 2005. ISSN 0885-8985.
- [61] MENEGAZ, H. M. T. et al. A Systematization of the Unscented Kalman Filter Theory. *IEEE Transactions on Automatic Control*, v. 60, n. 10, p. 2583–2598, oct 2015. ISSN 0018-9286, 1558-2523.
- [62] CARNEIRO, M. L. et al. Doherty amplifier optimization using robust genetic algorithm and unscented transform. In: IEEE. *New Circuits and Systems Conference (NEWCAS), 2011 IEEE 9th International*. [S.l.], 2011. p. 77–80.
- [63] ERLANDSSON, T.; NIKLASSON, L. Calculating uncertainties in situation analysis for fighter aircraft combat survivability. In: *2012 15th International Conference on Information Fusion*. [S.l.: s.n.], 2012. p. 196–203.
- [64] DAIGLE, M.; SAXENA, A.; GOEBEL, K. An efficient deterministic approach to model-based prediction uncertainty estimation. In: DTIC DOCUMENT. *Annual Conference of the Prognostics and Health Management Society 2012*. Minneapolis, MN; United States, 2012. p. 326–335.
- [65] LEAO, B. P.; YONEYAMA, T.; GOMES, J. P. P. Asymmetric Unscented Transform for Failure Prognosis. *IEEE Transactions on Reliability*, v. 65, n. 3, p. 1406–1415, sep 2016. ISSN 0018-9529, 1558-1721.
- [66] NIGAM, A. et al. Statistical Moment Estimation of Delay and Power in Circuit Simulation. *Journal of Low Power Electronics*, v. 6, n. 4, p. 578–587, dec 2010. ISSN 15461998.
- [67] GONG, F.; YU, H.; HE, L. Stochastic analog circuit behavior modeling by point estimation method. In: *Proceedings of the 2011 international symposium on Physical design - ISPD '11*. New York, New York, USA: ACM Press, 2011. p. 175. ISBN 9781450305501.
- [68] DELP, E.; MITCHELL, O. Image Compression Using Block Truncation Coding. *IEEE Transactions on Communications*, v. 27, n. 9, p. 1335–1342, sep 1979. ISSN 0096-2244.
- [69] DELP, E. J.; MITCHELL, O. R. Moment preserving quantization [signal processing]. *IEEE Transactions on Communications*, IEEE, v. 39, n. 11, p. 1549–1558, 1991.
- [70] MEDEIROS, J. E. et al. A fully analog low-power wavelet-based hearing aid front-end. In: IEEE. *2013 IEEE Biomedical Circuits and Systems Conference (BioCAS)*. [S.l.], 2013. p. 242–245.
- [71] MARTINS, I. F. et al. A novel wavelet-based analog-to-digital converter. In: *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. [S.l.]: IEEE, 2017. p. 1–4. ISBN 978-1-4673-6853-7.

- [72] SAMPAIO, A. M.; MEDEIROS, J. E. G. de. A framework for assessing the use of soc srams as physical unclonable functions. In: SBC. *XII SForum - Microelectronics Students Forum - Chip in Curitiba*. [S.l.], 2013.

Appendices

I. RESUMO ESTENDIDO

Este resumo estendido traz um resumo em português dos principais métodos e resultados contidos nesta tese.

I.1 Introdução

Circuitos eletrônicos analógicos são utilizados para implementar funções clássicas em processamento de sinais como amplificação de áudio, filtragem e para implementar interfaces de rádio. Em situações mais complexas, o processamento digital de sinais oferece algumas características vantajosas: maior relação sinal-ruído que possibilita melhor armazenamento de informação, a opção de executar operações complexas que não são óbvias no domínio analógico e a facilidade para adaptar algoritmos a mudanças no ambiente [1]. Para que uma aplicação se beneficie dessas vantagens, é preciso que sinais analógicos sejam convertidos para um formato digital, usualmente em um estágio inicial da cadeia de processamento. A Figura 1.1 mostra uma arquitetura típica de uma aplicação de processamento digital de sinais em que um sinal analógico é condicionado e convertido para o domínio digital por meio de um conversor analógico-digital — ADC. Após o processamento por um processador digital de sinais — DSP —, o sinal é convertido de volta para o domínio analógico por meio de um conversor digital-analógico — DAC — e é, por fim, recondicionado.

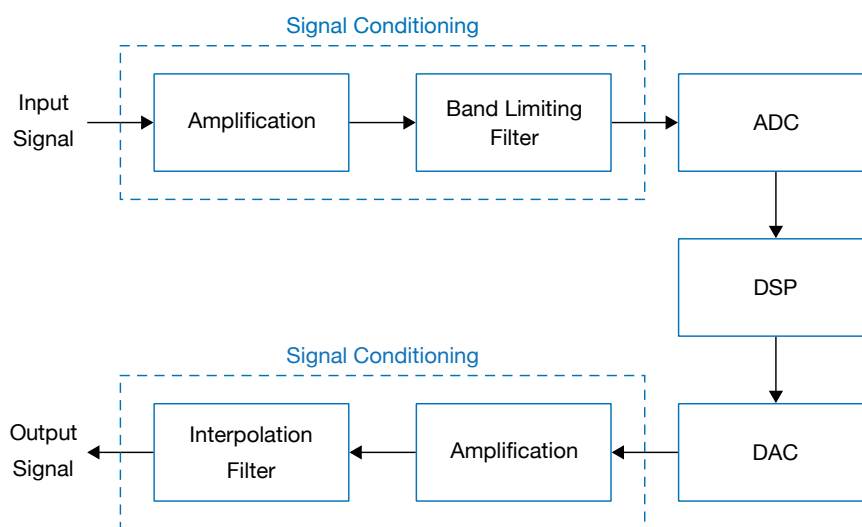


Figure I.1: Typical signal processing chain.

A conversão analógico-digital pode ser entendida por meio de dois processos, amostragem e quantização. Amostragem é o processo de discretizar o domínio de um determinado sinal. Quantização é o processo de discretizar a imagem (ou contra-domínio) de um determinado sinal. Vários critérios foram propostos para o projeto do processo de quantização, cada um procurando otimizar o processo para um objetivo diferente. Max propõe minimizar a distorção, definida como o valor

esperado de alguma função do erro entre a entrada e a saída do quantizador [5]. Esta é uma abordagem interessante uma vez que considera o comportamento estatístico do sinal de entrada no projeto do quantizador [6]. Lloyd estende essa ideia e propõe minimizar o erro quadrático médio causado pelo quantizador [7]. Em geral, estas técnicas resultam em quantizadores não-lineares, isto é, quantizadores em que o passo de quantização não é o mesmo para todos os níveis de quantização.

A quantização baseada na estatística do sinal é uma técnica padrão para codificação de sinais digitais. Entretanto, a maioria dos ADCs e DACs implementam quantizadores lineares (também chamados uniformes) [8]. O uso de quantizadores não-lineares em aplicações específicas pode prover melhor desempenho [9]. Ainda, a quantização linear altera a estrutura estatística do sinal de entrada [10, 11, 12]. Esta tese propõe o uso da Transformada da Incerteza para viabilizar o projeto de quantizadores para aplicações específicas cujo objetivo de otimização é preservar os momentos estatísticos de um dado sinal de entrada.

I.1.1 Objetivos e Escopo

The main objective of this work is to propose the use of the Unscented Transform as a framework for the design of quantizers for data conversion applications. In this way, we are setting the optimization goal of quantizer design to be the preservation of the statistical moments of the input signal and thus we are trying to carry structural information from the analog signal into the digital domain.

As an specific objective we want to analyze the performance of such quantizers and assess its usability in real applications. Specific objectives also include the development of an extension of the Unscented Transform theory and its applications as an alternative to Monte Carlo methods with better probability function estimation capabilities compared to the use of the UT.

I.2 A Transformada da Incerteza

Definição 1 A *Transformada da Incerteza* de ordem α de uma variável aleatória — VA — contínua \mathbf{x} com função densidade de probabilidade $p_{\mathbf{x}}(x)$ é um conjunto de n pares de pontos-sigma e pesos, $\{s_i, w_i\}$, que caracterizam uma distribuição de probabilidade discreta $F_{\mathbf{X}}(x)$ tal que

$$E\{\mathbf{X}^k\} = \sum_{i=1}^n s_i^k w_i = \int_{-\infty}^{+\infty} x^k p_x(x) dx = E\{\mathbf{x}^k\}, \quad (\text{I.1})$$

para $k = 1, 2, \dots, \alpha$, isto é, ambas VAs contínua e discreta apresentam os mesmos momentos até a α -ésima ordem.

I.2.1 The UT as a Mechanical Quadrature Problem

Começamos pela clássica definição de quadratura mecânica por Szego [27].

Definição 2 Considere $[a, b]$ um intervalo finito ou infinito e seja

$$S_n : x_1 < x_2 < \cdots < x_n, \quad (\text{I.2})$$

para $a \leq x_1$ e $x_n \leq b$, um conjunto de n pontos distintos em $[a, b]$. O conjunto S_n é chamado de partição do intervalo $[a, b]$. Ainda, seja

$$\Lambda_n : \lambda_1, \lambda_2, \dots, \lambda_n \quad (\text{I.3})$$

um conjunto de números reais. Chamamos a soma

$$Q_n(f) = \sum_{i=1}^n \lambda_i f(x_i) \quad (\text{I.4})$$

de *quadratura mecânica* de uma função $f(x)$ arbitrária definida sobre o intervalo $[a, b]$. Os números x_i serão chamados de *abscissas* da quadratura e λ_i de *números de Cotes* associados à quadratura.

A quadratura mecânica é um processo arbitrário que associa somas ponderadas de amostras de uma função $f(x)$ com números. É de especial interesse para este trabalho o caso em que os números de Cotes são definidos de forma que

$$Q_n(f) = \sum_{i=1}^n \lambda_i f(x_i) = \int_a^b f(x) du(x) \quad (\text{I.5})$$

é satisfeita se $f(x)$ é um polinômio arbitrário π_{n-1} de ordem até $n - 1$ e $u(x)$ é uma função não-decrescente. Neste caso, $Q_n(f)$ é chamada de quadratura interpolatória. Assim, de acordo com a definição 1, a UT é um caso particular de quadratura interpolatória em que $f(x) = x^k$ são monômios, os pontos-sigma $s_i = x_i$ são as abscissas da quadratura mecânica, os pesos $w_i = \lambda_i$ são os números de Cotes e $du(x) = p_{\mathbf{x}}(x)$ está relacionada com a função densidade de probabilidade que caracteriza a VA de interesse. A seguir estudamos um método para escolha das abscissas x_i e dos pesos λ_i .

I.2.2 A Quadratura Interpolatória

A quadratura interpolatória definida na Eq. I.5 é obtida substituindo o integrando $f(x)$ por um polinômio interpolador $P(x)$ convenientemente escolhido e considerando $\int_a^b P(x) du(x)$ como uma aproximação para $\int_a^b f(x) du(x)$. Interpolação pode ser entendida como o processo de escolher uma função contínua que concorda com dados discretos. Atualmente é utilizada como o fundamento para integração, resolução de equações diferenciais e problemas relacionados. A seguir definimos o problema de interpolação.

Definição 3 O *problema de interpolação* consiste em determinar um conjunto de parâmetros a_i de forma que dados $n + 1$ pares de números reais ou complexos (x_i, f_i) , $i = 0, \dots, n$, com $x_i \neq x_k$ para $i \neq k$, a equação

$$\Phi(x_i; a_0, \dots, a_n) = f_i \quad (\text{I.6})$$

é válida para $i = 0, \dots, n$ em que $\Phi(x; a_0, \dots, a_n)$ é uma família de funções de uma variável definidas pelos parâmetros a_i [29]. Os pares (x_i, f_i) são chamados de *pontos de suporte* da interpolação e os valores f_i de *ordenadas* da interpolação.

Estamos interessados no problema clássico de *interpolação polinomial* em que

$$\Phi(x; a_0, \dots, a_n) \equiv a_0 + a_1x + a_2x^2 + \dots + a_nx^n. \quad (\text{I.7})$$

Chamamos de π_n o conjunto de todos os polinômios reais ou complexos cujos graus não excedam n . A seguir reproduzimos um importante teorema descoberto por Edward Waring e disseminado pelo matemático Joseph Luis Lagrange.

Teorema I.2.1 *Para $n + 1$ pontos de suporte (x_i, f_i) arbitrários com $i = 0, \dots, n$ e $x_i \neq x_k$ para $i \neq k$, existe um único polinômio $P \in \pi_n$ tal que $P(x_i) = f_i$, para $i = 0, 1, \dots, n$.*

Demonstração *Existência:* Construimos a família de *polinômios de Lagrange* na forma

$$\begin{aligned} \ell_i(x) &= \frac{(x - x_0) \dots (x - x_{i-1})(x - x_{i+1}) \dots (x - x_n)}{(x_i - x_0) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_n)} \\ &= \frac{w(x)}{(x - x_i)w'(x_i)} \quad \text{com } w(x) = \prod_{i=0}^n (x - x_i). \end{aligned} \quad (\text{I.8})$$

Note que $\ell_i \in \pi_n$ para $i = 0, \dots, n$ e

$$\ell_i(x_k) = \delta_{ik} = \begin{cases} 1 & \text{se } i = k, \\ 0 & \text{se } i \neq k. \end{cases} \quad (\text{I.9})$$

Então, a solução para o problema de interpolação será

$$L(x) = \sum_{i=0}^n f_i \ell_i(x) = \sum_{i=0}^n f_i \prod_{\substack{k=0 \\ k \neq i}}^n \frac{x - x_k}{x_i - x_k}. \quad (\text{I.10})$$

A Eq. I.10 é chamada de *fórmula de interpolação de Lagrange*. □

I.2.3 A Quadratura Gaussiana

O teorema a seguir estabelece a existência de uma sequência de *polinômios ortogonais* associados a uma distribuição $p_x(x)$.

Teorema I.2.2 *Existem polinômios ortogonais $\rho_j \in \pi_j$, $j = 0, 1, 2, \dots$, tal que*

$$(\rho_i, \rho_k) = 0, \quad \text{para } i \neq k.$$

Estes polinômios são unicamente determinados por uma relação de recorrência de três termos dada

por

$$\rho_0(x) = 1 \quad (\text{I.11})$$

$$\rho_{i+1}(x) = (x - \alpha_{i+1})\rho_i(x) - \beta_{i+1}\rho_{i-1}(x) \quad (\text{I.12})$$

para $i \geq 0$, $\rho_{-1}(x) = 0$ e

$$\alpha_{i+1} = \frac{(x\rho_i(x), \rho_i(x))}{(\rho_i(x), \rho_i(x))} \quad \text{para } i \geq 0,$$

$$\beta_{i+1}^2 = \begin{cases} 1 & \text{para } i = 0, \\ \frac{(\rho_i(x), \rho_i(x))}{(\rho_{i-1}(x), \rho_{i-1}(x))} & \text{para } i \geq 1. \end{cases}$$

A sequência de polinômios ortogonais para uma dada distribuição depende somente da distribuição em si. Finalmente, o teorema a seguir relaciona a da quadratura numérica com um conjunto de polinômios ortogonais [27].

Teorema I.2.3 *Se $x_1 < x_2 < \dots < x_n$ denotam os zeros de $\rho_n(x)$, então existem números reais $\lambda_1, \lambda_2, \dots, \lambda_n$ tal que*

$$\int_a^b p(x) du(x) = \sum_{i=1}^n \lambda_i p(x_i) \quad (\text{I.13})$$

em que $p(x)$ é uma polinômio arbitrário de grau não mais do que $2n - 1$. A distribuição $du(x)$ e o número inteiro n são suficientes para determinar os números λ_n .

A definição 1 estabelece que a UT pode ser entendida como um caso especial da quadratura mecânica I.4 com $f(x) = x^k$, $k = 1, 2, \dots, \alpha$. Concluímos que os zeros do polinômio ortogonal constituem um conjunto de pontos-sigma com pesos $w_i = \lambda_i$. O terorema a seguir estabelece que esta escolha de pontos-sigma maximiza a ordem da UT, isto é, o número de momentos que são preservados na transformada.

Teorema I.2.4 *Não é possível encontrar números x_i , $i = 1, \dots, n$, tal que a Eq. (I.13) seja verdadeira para todos os polinômios $p(x) \in \pi_{2n}$.*

O dois teoremas a seguir estabelecem uma abordagem prática para o cálculo dos pontos-sigma e pesos $\{s_i, w_i\}$ com base na teoria da quadratura Gaussiana.

Teorema I.2.5 *As raízes x_i , $i = 1, \dots, n$, do polinômio ortogonal $\rho_n(x)$ são os autovalores da*

matriz tridiagonal

$$J_n = \begin{bmatrix} \alpha_1 & \sqrt{\beta_2} & 0 & 0 & 0 & \dots & 0 \\ \sqrt{\beta_2} & \alpha_2 & \sqrt{\beta_3} & 0 & 0 & \dots & 0 \\ 0 & \sqrt{\beta_3} & \alpha_3 & \sqrt{\beta_4} & 0 & \dots & 0 \\ & & \ddots & \ddots & \ddots & & \\ 0 & 0 & \dots & \sqrt{\beta_{n-2}} & \alpha_{n-2} & \sqrt{\beta_{n-1}} & 0 \\ 0 & 0 & \dots & 0 & \sqrt{\beta_{n-1}} & \alpha_{n-1} & \sqrt{\beta_n} \\ 0 & 0 & \dots & 0 & 0 & \sqrt{\beta_n} & \alpha_n \end{bmatrix} \quad (\text{I.14})$$

em que α_i e β_i são os coeficientes da relação de recorrência de três termos definida na Eq. (I.2.2).

Teorema I.2.6 Seja $v^{(i)} = (v_1^{(i)}, \dots, v_n^{(i)})^T$ um autovetor da matriz J_n definida na Eq. (I.14) para o autovalor x_i , isto é, $J_n v^{(i)} = x_i v^{(i)}$. Suponha que $v^{(i)}$ é escalonado de forma que

$$v^{(i)T} v^{(i)} = (\rho_0, \rho_0) = \int_a^b du(x).$$

Então, os pesos w_i serão dados por

$$w_i = (v_1^{(i)})^2, \quad i = 1, \dots, n. \quad (\text{I.15})$$

I.3 Estimação de Funções de Probabilidade

Considera um oscilador em anel composto por n inversores conectados em malha fechada como mostrado na Fig. I.2. Se n for um número ímpar, a frequência de saída do circuito será dada por

$$f_{\text{out}} = \frac{1}{2 \sum_n d_n} \quad (\text{I.16})$$

em que d_n é o atraso do n -ésimo inversor [39]. Variações no processo de fabricação fazem com que o atraso de cada inversor possa variar entre diferentes lotes. Por conta disso, é conveniente modelar d_n como uma variável aleatória. Para este estudo de caso, considere $n = 3$ e $d_n = \mathcal{N}(d_0, \Delta d)$ uma variável aleatória com distribuição Gaussiana com média $d_0 = 1\mu s$ e variância $\Delta d = 0.02\mu s$. Dessa forma, é esperada uma frequência de saída de 166.7kHz.

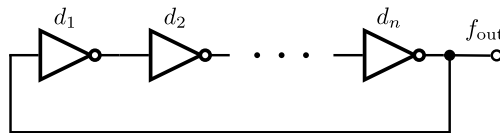


Figure I.2: Oscilador em anel utilizado no estudo de caso. Os atrasos d_n de cada inversor são modelados por uma variável aleatória Gaussiana com média $d_0 = 1\mu s$ e variância $\Delta d = 0.02\mu s$.

Para este circuito é possível encontrar uma expressão analítica para a função densidade de probabilidade que caracteriza a frequência de saída f_{out} :

$$p_y(y) = \frac{1}{2y^2} \times \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\left(\frac{1}{2y} - \mu\right)^2 / 2\sigma^2}. \quad (\text{I.17})$$

Utilizaremos este resultado como referência para as comparações a seguir entre o método de Monte Carlo e a UT.

A Figura I.3a mostra a implementação do método de Monte Carlo para este estudo de caso. O algoritmo começa por escolher o número de simulações N a serem executadas. A seguir, executa-se um laço em que sorteiam-se números aleatórios d_i de acordo com a distribuição de probabilidade que modela a VA e executa-se o modelo. Ao final, é construído o histograma de $f[n]$ e estima-se a função densidade de probabilidade a partir dele.

A Figura I.3b mostra o algoritmo utilizando a UT. Primeiro se escolhe o número de pontos-sigma m para representar cada VA. A seguir, constrói-se o conjunto $\{s_i, w_i\}$ que será utilizado para exercitar o modelo do circuito em todas as combinações de pontos-sigma ao invés de números escolhidos aleatoriamente. Cada ponto é ponderado ao final pelo peso relativo dado pelo valor de w_i . Ao final, $f[n]$ é inspecionado e simplificado.

A Figura I.4 mostra o erro relativo dos momentos estimados, definidos por

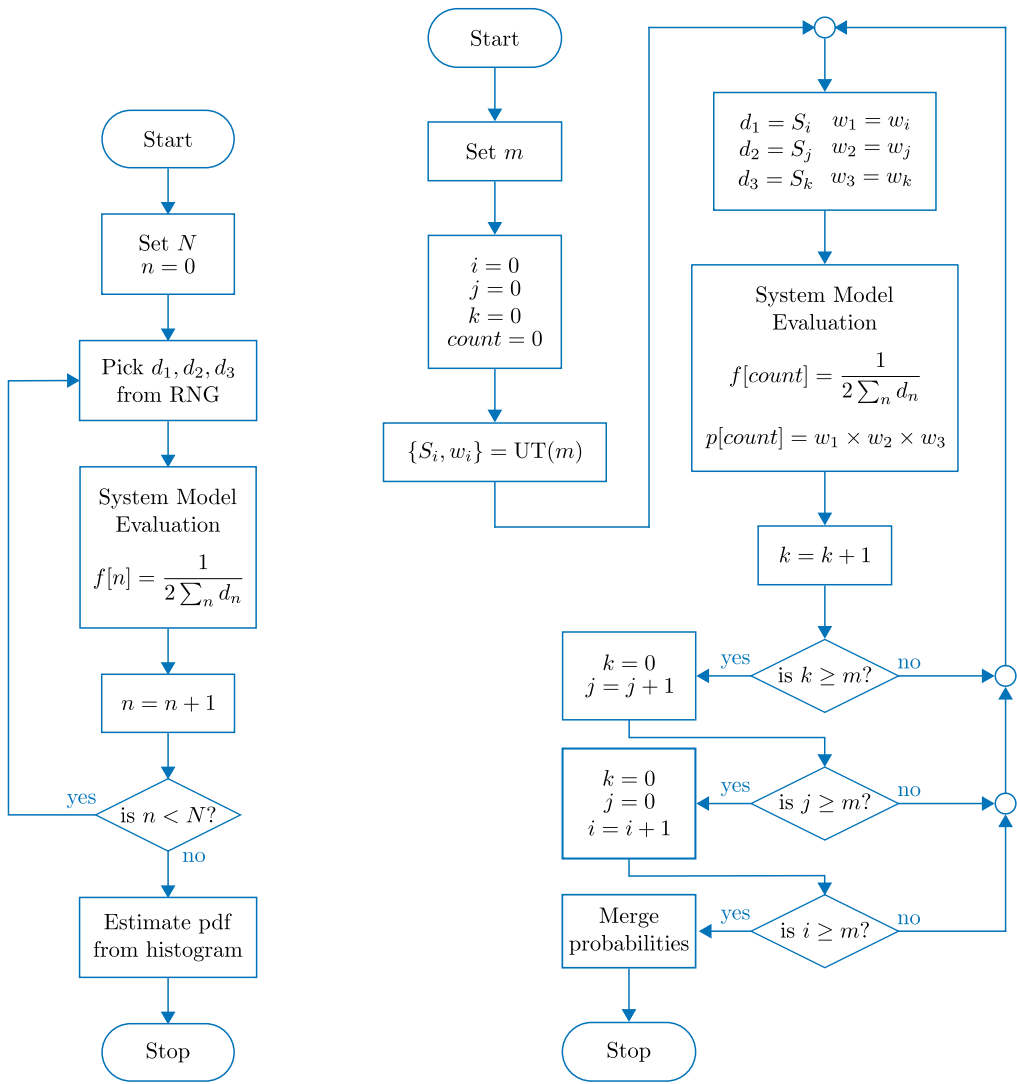
$$\eta = \left| \frac{m_{\text{exact}} - m_{\text{estimate}}}{m_{\text{exact}}} \right|$$

em que m_{exact} são os momentos exatos calculados de acordo com a Eq. (I.17) e m_{estimate} são as estimativas dadas pelo método da UT. Como esperado, quanto mais ponto-sigma utilizados no processo, mais momentos são corretamente estimados. Se o interesse for apenas nos primeiros momentos, um número pequeno de ponto-sigma pode ser utilizado.

Na Figura I.5 são apresentados os error relativos na estimação dos momentos para uma execução típica do método de Monte Carlo. É possível perceber a convergência do método de Monte Carlo com o aumento do número de pontos utilizados. Apesar disso, para o mesmo número de simulações executadas, o método da UT apresenta menores erros na estimação dos momentos de ordem superior.

I.3.1 A UT Estendida

A Transformada da Incerteza mapeia uma variável aleatória contínua, \mathbf{x} , em uma discreta, \mathbf{X} . Se o interesse for a aproximação da função distribuição de probabilidade contínua da VA $\mathbf{y} = g(\mathbf{x})$ após uma transformação não-linear $g(\cdot)$ a partir da distribuição de probabilidade discreta $\mathbf{Y} = g(\mathbf{X})$ será necessário interpolar os pontos-sigma após o mapeamento para obter a aproximação desejada. Menezes [40] observou que a função de probabilidade acumulada de \mathbf{X} irá apresentar um padrão de escada como mostrado na Figura I.6c e propôs utilizar um esquema de interpolação de primeira ordem. Esta abordagem é detalhada e estendida a seguir.



(a) Fluxograma para o método de Monte Carlo.

(b) Fluxograma para o método com a Transformada da Incerteza.

Figure I.3: Fluxogramas comparando o método de estimação de funções de probabilidade com Monte Carlo e com a Transformada da Incerteza.

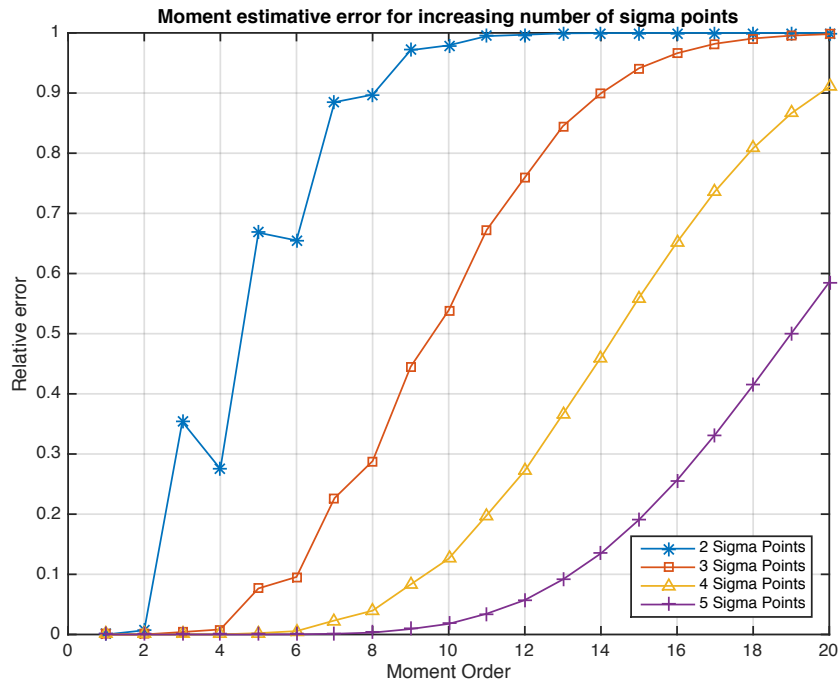


Figure I.4: Erro relativo da estimação dos momentos pelo método da UT.

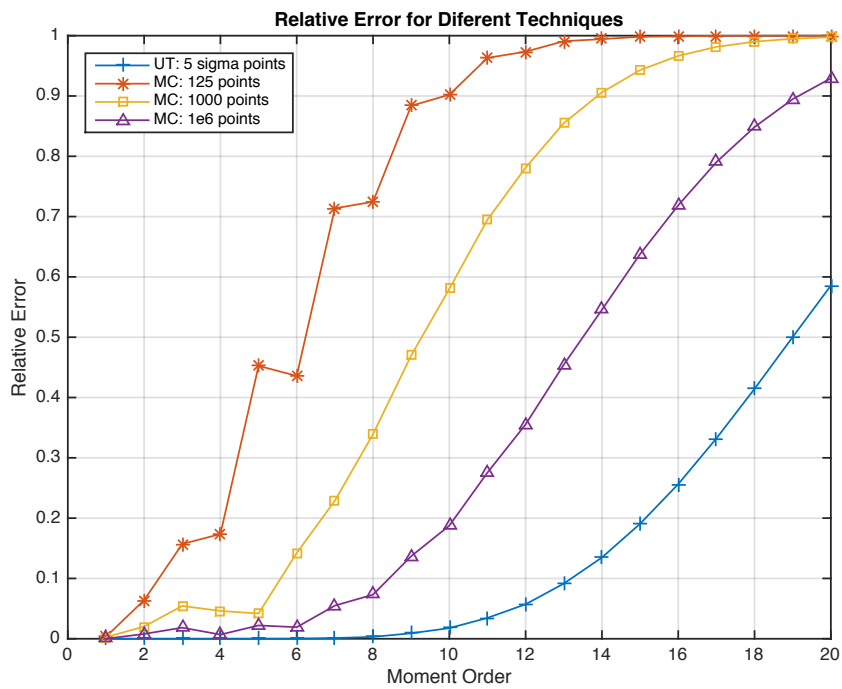


Figure I.5: Erro relativo para uma execução típica do método de Monte Carlo com aumento no número de execuções comparado com o resultado do método da UT.

Esta estratégia implica em uma nova formulação para a UT baseada numa nova variável aleatória \mathbf{X}_{bars} com uma função densidade de probabilidade de primeira ordem como mostrada na Figura I.6b

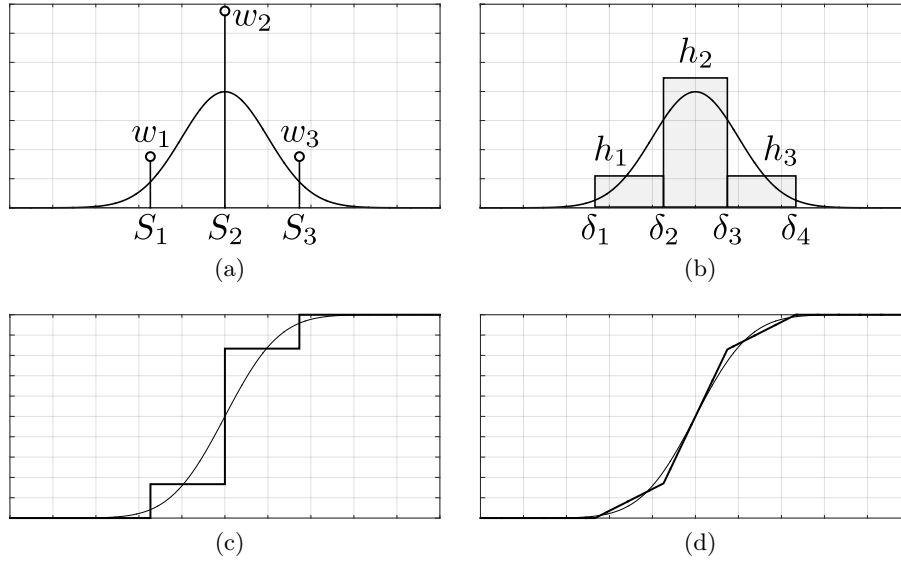


Figure I.6: Diferentes formulações para a UT. (a) Função densidade de probabilidade e os pontos-sigma da UT. (b) Função densidade de probabilidade e a aproximação de primeira ordem. (c) Funções de probabilidade acumulada para a VA contínua e para a UT. (d) Funções de probabilidade acumulada para a VA contínua e para a aproximação de primeira ordem. O comportamento observado em (d) é a inspiração para o desenvolvimento da UT Estendida.

e não mais em pontos discretos. Dessa forma, equacionando os momentos de ambas as VAs temos

$$E\{\mathbf{X}_{\text{bars}}^k\} = \sum_{i=1}^m h_i \int_{\delta_i}^{\delta_{i+1}} x^k dx = \int_{-\infty}^{+\infty} x^k p_x(x) dx = E\{\mathbf{x}^k\} \quad (\text{I.18})$$

para $k = 0, 1, 2, \dots, 2m + 1$ onde m representa o número de barras na função densidade de probabilidade de \mathbf{X}_{bars} , $p_x(x)$ é a função densidade de probabilidade da VA de entrada \mathbf{x} , h_i é a altura da i -ésima barras e δ_i são os pontos de transição entre a i -ésima e a $i + 1$ -ésima barras como mostrado na Figura ??b.

Esta formulação impões um novo desafio para a análise de mapeamentos não-lineares uma vez que precisamos analisar uma nova VA contínua e não há vantagem em relação a analisar a VA de origem. Para solucionar esta questão, propomos modelar esta nova VA contínua por pontos discretos como mostrado na Figura I.7.

É possível concluir que

$$E\{\mathbf{X}_{\text{bars}}^k\} = \sum_{k=1}^m h_k (\delta_{k+1} - \delta_k) E\{\mathbf{U}[\delta_k, \delta_{k+1}]^k\} \quad (\text{I.19})$$

em que m é o número de barras na função de densidade de probabilidade de \mathbf{X}_{bars} e $\mathbf{U}[\mathbf{a}, \mathbf{b}]$ é a função densidade de probabilidade de uma VA com distribuição uniforme contínua definida no suporte $[\delta_i, \delta_{i+1}]$ e os pontos-sigma para $\mathbf{X}_{\mathbf{s}}$ podem ser obtidos a partir da UT de uma VA contínua com distribuição uniforme. A este conjunto de pontos-sigma e os pesos associados a eles damos o nome de Transformada da Incerteza estendida. A Figura I.7 mostra os passos deste processo.

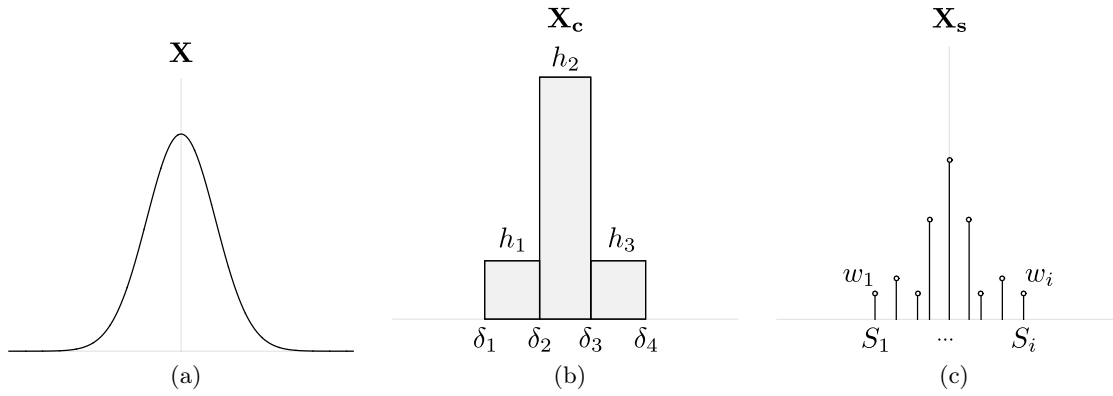


Figure I.7: Representação gráfica da Transformada da Incerteza Estendida. (a) A função densidade de probabilidade inicial da VA \mathbf{X} . (b) A função densidade de probabilidade contínua da VA \mathbf{X}_c . (c) OS pontos sigma da Transformada da Incerteza Estendida \mathbf{X}_s .

I.4 Projeto de Quantizadores Baseado na UT

Quantização é o processo que mapeia um conjunto de valores de entrada em outro conjunto menor contável. Em aplicações de conversão de dados, a entrada assume valores contínuos em um domínio definido (faixa dinâmica de entrada) e é usualmente modelada por VAs contínuas definidas em um suporte compacto $[a, b]$. A saída do quantizador assume valores discretos usualmente codificados utilizando um alfabeto binário. A UT é um processo matemático que modela uma função densidade de probabilidade contínua, $p_{\mathbf{y}}(y)$, em uma discreta, $p_{\mathbf{y}_q}(y_q)$. Neste trabalho é proposto o uso da UT como um modelo para o processo de quantização em conversores de dados. A Figura I.8 ilustra a ideia de projetar um quantizador que imita as propriedades da UT, isto é, o sinal de saída preserva os α primeiros momentos do sinal de entrada.

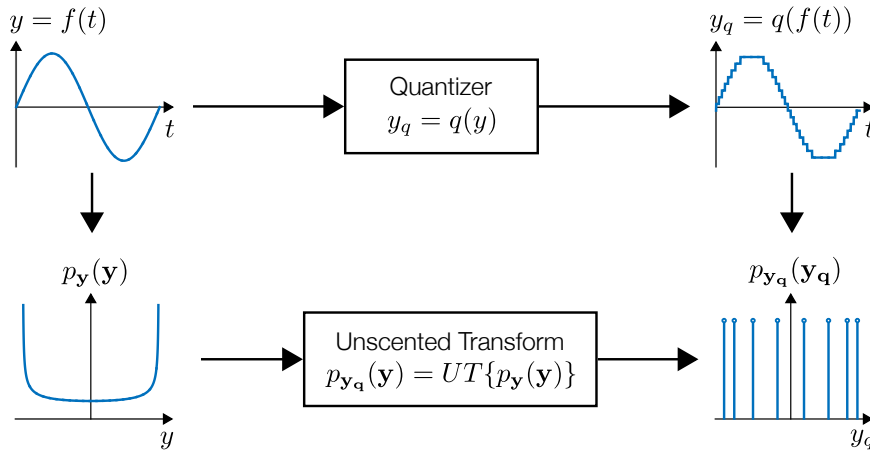


Figure I.8: Um quantizador mapeia um sinal de entrada, $y = f(t)$, em um sinal de saída quantizado, $y_q = q(y)$. A UT, por outro lado, mapeia uma função densidade de probabilidade contínua, $p_{\mathbf{y}}(y)$, em uma função de probabilidade discreta, $p_{\mathbf{y}_q}(y_q)$. O sinal de entrada é modelado como uma função de probabilidade para mostrar a relação entre o quantizador e a UT.

Vamos mostrar como modelar o sinal de entrada como uma variável aleatória. Considere um

sinal periódico $y = f(t)$ com período T_y . Desejamos coletar n amostras deste sinal no intervalo $0 < t < T_y$ com período de amostragem $T_s = T_y/n$, $n \in \mathbb{N}_+$. Dessa forma, obtemos a sequência $T_n = \{t_0 = 0, t_1 = T_y/n, \dots, t_n = (n-1)T_y/n\}$ de instantes de amostragem, cada um ocorrendo apenas uma vez e portanto T_n apresenta uma função de probabilidade uniforme

$$p_t(t) = \begin{cases} 1/n & \text{if } t \in T_n \\ 0 & \text{otherwise} \end{cases} \quad (\text{I.20})$$

Calculando a função de probabilidade acumulada de T_n , é possível concluir que, no caso contínuo, o tempo pode ser modelado por uma VA contínua \mathbf{t} com distribuição uniforme no intervalo $0, T_y$. O mesmo argumento é válido para qualquer número de períodos do sinal de entrada. A partir deste resultado, podemos entender um sinal de entrada arbitrário $\mathbf{y} = f(\mathbf{t})$ como uma transformação não-linear aplicada sobre a VA uniforme \mathbf{t} .

I.4.1 Projeto do Quantizador

Para definir um quantizador, precisamos encontrar um conjunto com N níveis de quantização e $N-1$ passos de quantização, th_n . A Figura I.9 mostra a curva característica para um quantizador com 4 níveis. É evidente que os níveis de quantização serão dados diretamente pelos pontos-sigma da UT, S_i .

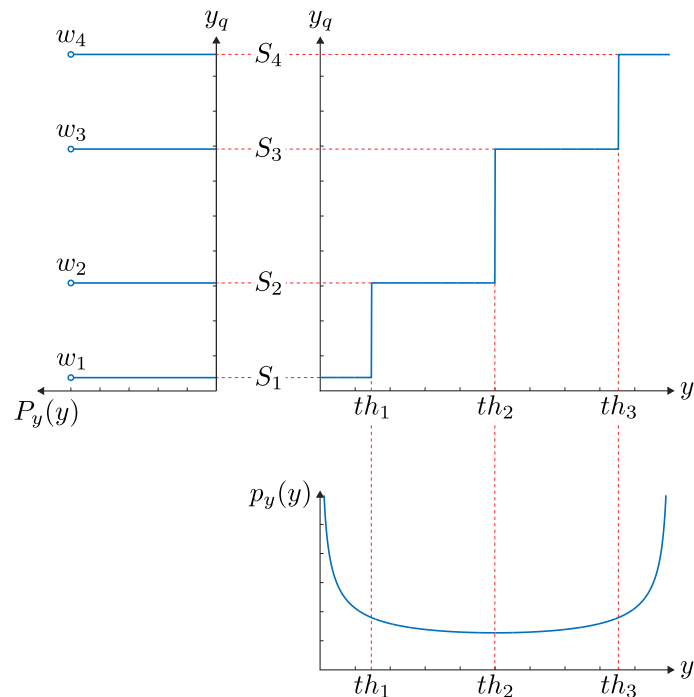


Figure I.9: Curva característica para um quantizador de 4 níveis para a distribuição arco-seno. A distribuição arco-seno, $p_y(y) = \frac{1}{\pi\sqrt{y-y^2}}$, modela um sinal de entrada senoidal no intervalo $[0, 1]$. O quantizador mapeia a distribuição de entrada na saída y_q de acordo com a Eq. (I.22).

Para definir os passos de quantização, equaciona-se a probabilidade de encontrar o sinal de

entrada no intervalo $[th_{n-1}, th_n]$ e a probabilidade descrita pelo peso w_n associado ao ponto-sigma S_n , isto é:

$$P(th_{n-1} < x < th_n) = \int_{th_{n-1}}^{th_n} p_x(x) dx = w_n \quad (\text{I.21})$$

$$P_x(th_n) - P_x(th_{n-1}) = w_n$$

em que $P_x(x) = \int_{-\infty}^x p_x(\tau) d\tau$ é a função de probabilidade cumulativa associada a $p_x(x)$ no intervalo $[a, b]$ e $t_0 = a$. Resolvendo a Eq. (I.21) no caso geral, concluímos que os passos de quantização th_n estarão localizados em

$$th_n = P_x^{-1} \left(\sum_{i=1}^n w_i \right) \quad (\text{I.22})$$

em que $P_x^{-1}(P_x(x)) = x$ é conhecida como a função quantil. O mesmo resultado é válido para funções de probabilidade com suporte infinito considerando que o quantizador com faixa dinâmica dada por $[a, b]$ irá mapear qualquer valor de entrada menor do que a para S_i e qualquer valor de entrada maior do que b para S_N .

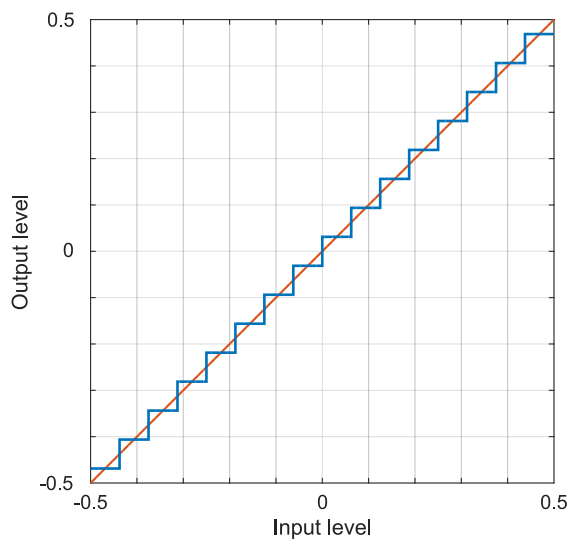
A Figura I.10 mostra as curvas características para um quantizador linear e para o quantizador da distribuição arco-seno. O quantizador linear é caracterizado por passos de quantização constantes. Percebemos que o quantizador da distribuição arco-seno apresenta passos de quantização mais finos, e portanto maior resolução, nos extremos do intervalo de entrada. Por outro lado, o quantizador não-linear apresenta passos de quantização mais largos, e portanto menor resolução, no centro da faixa dinâmica de entrada.

A Figura 4.4 mostra o efeito dos diferentes esquemas de quantização no sinal de saída y_q . Note que o efeito da não-linearidade do quantizador da distribuição arco-seno é prover uma melhor representação do sinal de entrada nas regiões em que o sinal é mais lento e, portanto, tem maior probabilidade de ocorrer. Este comportamento se assemelha ao esquema de quantização apresentado por Tsividis em [44], mas neste caso, o número de níveis de quantização é constante e os passos de quantização são determinados de acordo com um critério estatístico.

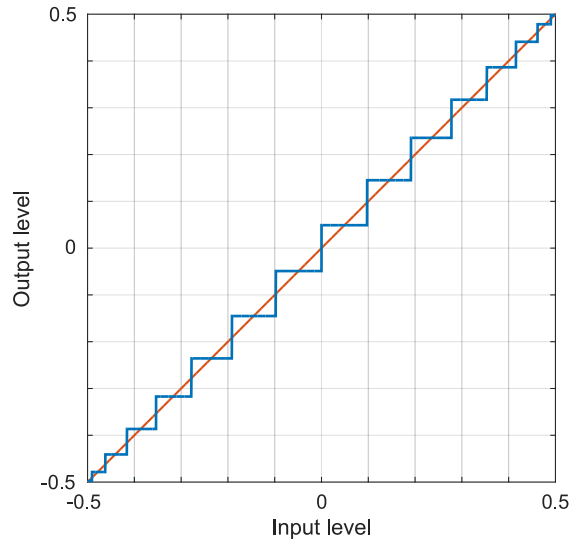
A Figura I.12 mostra a relação sinal-ruído — SNR —, a relação sinal-ruído e distorção — SINAD — e a curva teórica $SINAD = 6.02N + 1.76\text{dB}$. Percebe-se que o quantizador da distribuição arco-seno não altera a SNR nem a SINAD sugerindo que a quantidade de distorção gerada pelo quantizador não-linear é a mesma da gerada pelo quantizador linear.

A Figura I.13a, por outro lado, mostra que a distorção harmônica total — THD — calculada até o 10º harmônico para diferentes resoluções. É possível perceber uma redução de cerca de 16dB na THD, ou aproximadamente 2,3 bits, na faixa analisada. Esta melhoria se mostra razoavelmente insensível a resolução do quantizador.

Finalmente, a Figura I.13b mostra a faixa dinâmica livre de espúrios — SFDR — dos quantizadores junto da curva teórica $SFDR = 9.03n$ obtida por Blachman [52] e Abidi [47]. Percebe-se um aumento de aproximadamente 14dB para resoluções baixas.

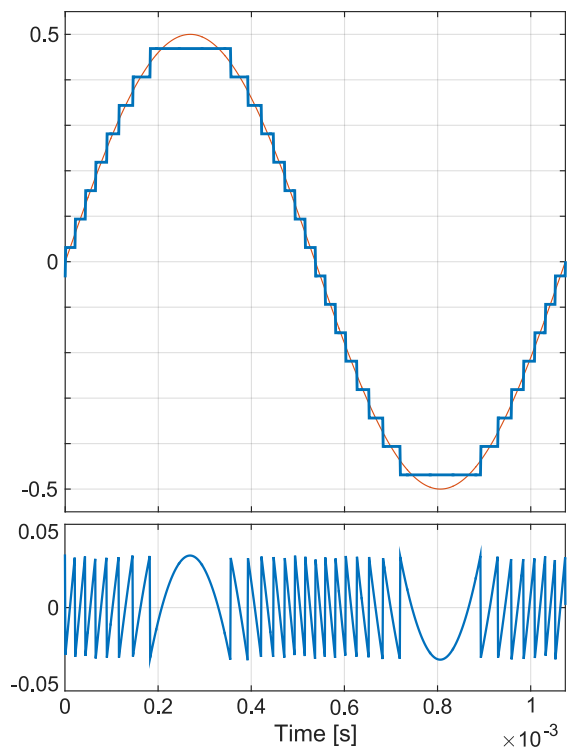


(a) Quantizador linear.

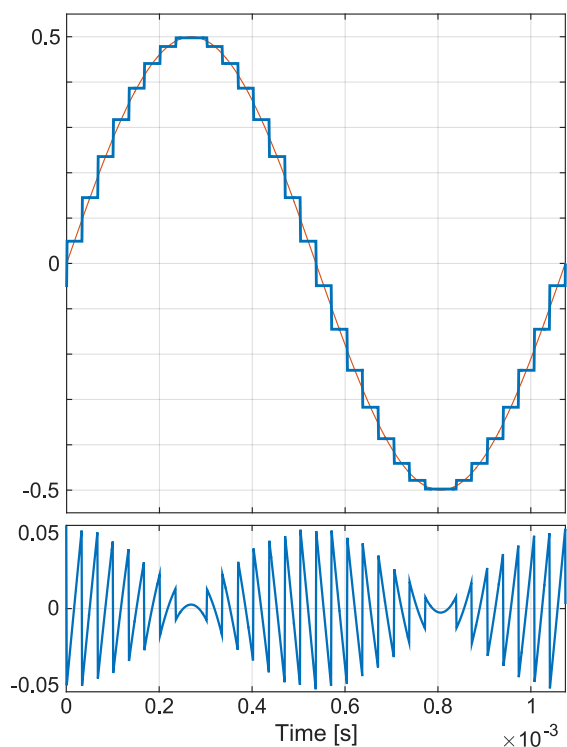


(b) Quantizador da distribuição arco-seno.

Figure I.10: Curvas características dos quantizadores linear e da distribuição arco-seno. O quantizador linear é caracterizado por passos de quantização constantes enquanto o quantizador da distribuição arco-seno apresenta passos de quantização mais finos nos extremos da faixa dinâmica de entrada.

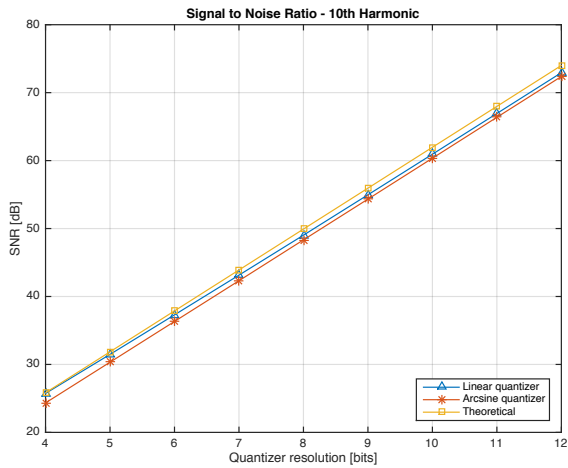


(a) Quantizador linear.

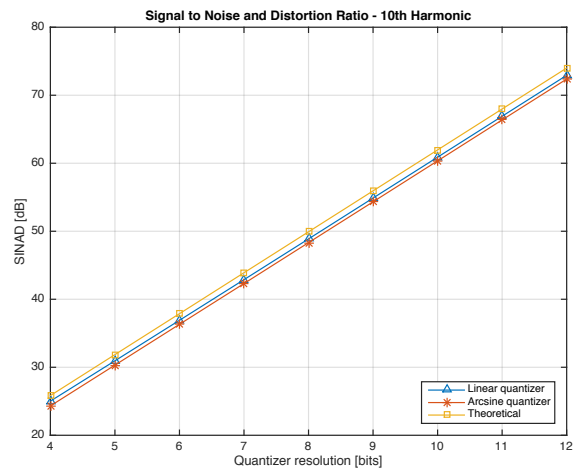


(b) Quantizador da distribuição arco-seno.

Figure I.11: Análise transiente para um sinal de entrada senoidal processado por um quantizador linear e por um quantizador da distribuição arco-seno mostrando o erro de quantização. Em I.11a é obtido o padrão clássico de sino/dente de serra para o erro de quantização. Em I.11b percebe-se que o erro de quantização é menor nos extremos do sinal de entrada por conta dos passos de quantização mais finos. Nota-se, ainda, que o menor erro de quantização ocorre quando o sinal de entrada apresenta variações mais lentas.

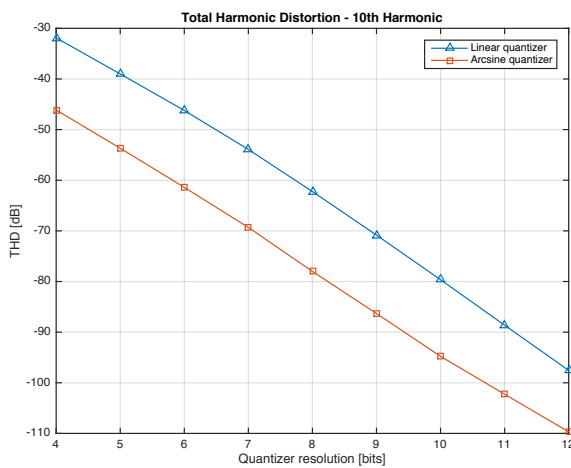


(a) Relação sinal-ruído.

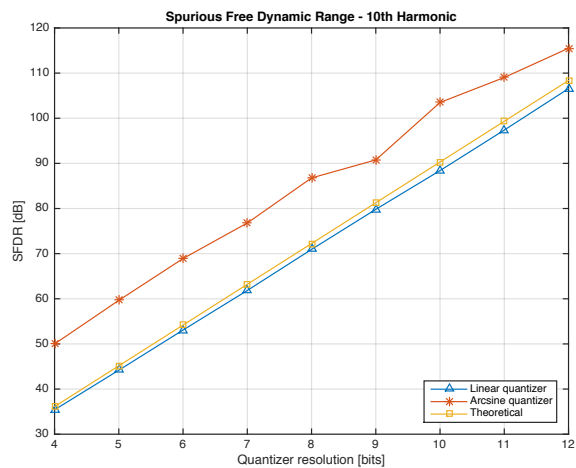


(b) Relação sinal-ruído e distorção.

Figure I.12: SNR e SINAD para quantizadores de diferentes resoluções.



(a) Distorção harmônica total, calculada até o 10^o harmônico.



(b) Faixa dinâmica livre de espúrios.

Figure I.13: THD e SFDR para quantizadores de diferentes resoluções.

I.5 Conclusões

Nesta tese foi apresentada uma definição formal da Transformada da Incerteza. Foi mostrado que a UT pode ser entendida como um caso particular da teoria clássica da quadratura interpolatória por meio de teoremas relevantes.

Foi mostrado que a Transformada da Incerteza pode ser uma ferramenta efetiva para a análise estatística de mapeamentos não-lineares em diferentes aplicações. A UT foi apresentada como alternativa a métodos de Monte Carlo. Foi proposta uma formulação estendida da UT que permite obter maior número de pontos-sigma. Apesar de apresentar complexidade exponencial, a técnica proposta aponta para implementações paralelas sem os problemas associados a geradores de números aleatórios.

Este trabalho propôs utilizar informação *a priori* contida num sinal de entrada para projetar o processo de quantização em conversores de dados. O objetivo é preservar o maior número de momentos estatísticos do sinal de entrada e para isso foi apresentado um método de projeto baseado na UT. Dessa forma, foi possível mostrar que as teorias da UT, da quadratura numérica e da quantização estão relacionadas. Por fim, foi analisado um estudo de caso que mostra evidências de que o erro de quantização causado por um quantizador baseado na UT é o mesmo que o causado por um quantizador linear enquanto a distorção se concentra em faixa de frequência diferentes.

II. UNSCENTED TRANSFORM IMPLEMENTATION

The following Matlab source code implement some of the recurrence relations shown in Table 2.1 and Theorems 2.3.7 and 2.3.8. To compute the UT, we start with this canonical implementations and scale them to the desired probability distributions.

II.1 Recurrence Relations

The following functions compute the three term recurrence relation coefficients for the Hermite, Laguerre and Jacobi polynomials.

```
1 function coeff = ttr_hermite(N)
2 # Parameter check
3 if N <= 0
4     error('You_should_supply_the_number_of_recurrence_coefficients_to_compute.');
```

```
5 end
6
7 beta0 = sqrt(pi);
8
9 if N == 1
10     coeff = [0 beta0];
11     return;
12 end
13
14 n = 1:(N-1);
15 nn = n/2;
16
17 # Coefficients
18 alpha_n = zeros(1,N);
19 beta_n = [beta0 nn];
20 coeff = [alpha_n' beta_n'];
```

Listing II.1: Hermite polynomials recurrence relation coefficients computation source code.

```

1 function coeff = ttr_laguerre(N)
2 # Parameter check
3 if (N<=0)
4     error('Parameter_out_of_range'),
5 end
6
7 if N==1,
8     coeff = [1 1];
9     return,
10 end
11
12 n = 1:(N-1);
13 na = 2*n+1;
14 nb = n.^2;
15
16 # Coefficients
17 alpha_n = [1 na];
18 beta_n = [1 nb];
19 coeff = [alpha_n' beta_n'];

```

Listing II.2: Laguerre polynomials recurrence relation coefficients computation source code.

```

1 function coeff = ttr_jacobi(N,a,b)
2 # Parameter check
3 if nargin<2
4     a = 0;
5 end;
6 if nargin<3
7     b = a;
8 end
9 if ((N<=0) | (a<=-1) | (b<=-1))
10     error('Parameter(s)_out_of_range')
11 end
12
13 alpha0 = (b-a)/(a+b+2);
14 beta0 = 2^(a+b+1)*gamma(a+1)*gamma(b+1)/gamma(a+b+2);
15
16 if N==1
17     coeff = [alpha0 beta0];
18     return;
19 endif
20
21 n = 1:(N-1);
22 j = 2*n + a + b;
23
24 alpha_n = [alpha0 (b^2-a^2)*ones(1,N-1)./(j.*(j+2))];
25
26 n = 2:(N-1);
27 j = j(n);
28
29 beta1 = 4*(a+1)*(b+1)/((a+b+2)^2*(a+b+3));
30 beta_n = 4*(n+a).*(n+b).*n.*(n+a+b)./((j.^2).*(j+1).*(j-1));
31
32 coeff = [alpha_n' [beta0; beta1; beta_n]];

```

Listing II.3: Jacobi polynomials recurrence relation coefficients computation source code.

II.2 Gaussian Quadrature

The following function compute the abscissas and weights of the Gaussian quadrature given the coefficients of the three term recurrence relation for the distribution of interest.

```
1 function sw = quadrature(N,coeff)
2 # Parameter check
3 if (size(coeff,1)) < N
4     error('Not_enough_coefficients.')
```

```
5 end
6
7 # Jacobi matrix diagonal
8 J = zeros(N);
9 for n = 1:N
10     J(n,n) = coeff(n,1)
11 end
12
13 # Jacobi matrix subdiagonals
14 for n = 2:N
15     J(n,n-1) = sqrt(coeff(n,2));
16     J(n-1,n) = J(n,n-1);
17 end
18
19 # Eigenvalues and eigenvectors
20 [V,D] = eig(J);
21 [D,I] = sort(diag(D));
22 V = V(:,I);
23 sw = [D coeff(1,2)*V(1,:)'.^2];
```

Listing II.4: Gaussian quadrature abscissas and weights computation source code.

II.3 Unscented Transform

The following functions compute the UT sigma-points and weights from the Gaussian quadrature abscissas.

```
1 function [sigma, weights] = UT_unif(a,b,n)
2 % UT_UNIF Computes n UT sigma points and weights for
3 %     uniform distribution in [a,b] interval.
4 %
5 % The UT is a form of gaussian quadrature. For the uniform
6 % distribution, the sigma points will be the zeros of
7 % the n-th order Legendre polynomial. In this implementation,
8 % we use the Jacobi weighting function with parameters
9 % a=0, b=0.
10
11 % Gaussian quadrature nodes and weights.
12 xx = gauss(n, ttr_jacobi(n,0,0));
13
14 % Scaling to the distribution.
15 sigma = (a*(1-xx(:,1))+b*(1+xx(:,1)))/2;
16 weights = xx(:,2)/2;
17 end
```

Listing II.5: UT sigma-points and weights computation for the uniform distribution.

```
1 function [sigma, weights] = UT_norm(mu, sd, n)
2 %UT_NORM Computes n UT sigma points and weights for
3 %     normal distribution with mean mu and
4 %     standard deviation sd.
5 %
6 % The UT is a form of gaussian quadrature. For the
7 % normal distribution, the sigma points will be the zeros
8 % of the n-th order Hermite polynomial.
9
10 % Gaussian quadrature nodes and weights.
11 xx = gauss(n, ttr_hermite(n));
12
13 % Scaling to the standard distribution.
14 s = xx(:,1)*sqrt(2);
15 weights = xx(:,2)/sqrt(pi);
16
17 % Linear map to the desired mu and sd.
18 sigma = sd * s + mu;
19 end
```

Listing II.6: UT sigma-points and weights computation for the normal distribution.

```

1 function [sigma, weights] = UT_exp(lambda, n)
2 %UT_EXP Computes n UT sigma points and weights for
3 %     exponential distribution with parameter lambda.
4 %
5 % The UT is a form of gaussian quadrature. For the
6 % exponential distribution, the sigma points will be the zeros
7 % of the n-th order Laguerre polynomial.
8
9 % Gaussian quadrature nodes and weights.
10 xx = gauss(n, ttr_laguerre(n));
11
12 % Scaling to the distribution.
13 sigma = xx(:,1)/lambda;
14 weights = xx(:,2)/sqrt(pi);
15 end

```

Listing II.7: UT sigma-points and weights computation for the exponential distribution.

The UT for the arcsine distribution has analytical solution.

```

1 function [sigma, weights] = UT_arcsine(a,b,n)
2 %UT_UNIF Computes n UT sigma points and weights for
3 %     the arcsine distribution with zero mean,
4 %     scaled to the interval [a,b].
5 %
6 % The UT is a form of gaussian quadrature. For the
7 % arcsine distribution, the sigma points will be the zeros
8 % of the n-th order Chebyshev polynomial. The analytical
9 % form for the nodes and weights is known.
10
11 % Uniform sampling on the unit circle.
12 xk = cos(pi*(2*(1:n)-1)/(2*n));
13
14 % Weights will be all equal.
15 w = (1/n)*ones(1,n);
16
17 % Adjust distribution interval and order of the sigma points.
18 temp = sortrows([xk' w']);
19
20 % Linear map from[-1,1] to [a,b]
21 sigma = (a*(1-temp(:,1))+b*(1+temp(:,1)))/2;
22 weights = temp(:,2);
23 end

```

Listing II.8: UT sigma-points and weights computation for the arcsine distribution.

II.4 The Extended UT

The following source code was used to generate data for Chapter 3 case study.

```
1 function [sigma, weights] = UT_hist3_norm(n_bars, n_points)
2 %UT_NORM Calculates N UT sigma points and weights for
3 %     normal distribution
4 %
5 % Based on the paper for the Electronics Letters
6
7 %%% Calculation of the moments for uniform distribution.
8 %%% For this, I used the Moment Generating Function for the uniform
9 %%% distribution.
10
11 N = n_bars + 1;
12 % Number of moments to calculate
13 Nm = 2*(N-1)+2;
14 Mx = zeros(Nm+1, 1);
15 Mx(1) = 1;
16
17 % Moment calculation
18 for k = 3:+2:Nm+1
19     Mx(k) = prod(1:2:k-1);
20 end;
21
22 %%% First order UT for the bars distribution
23 A = zeros(N);
24 B = zeros(N, 1);
25 P = zeros(N-1, 1);
26 D = zeros(N-1);
27
28 for k = 1:N
29     for m = 1:N
30         A(k,N-m+1) = Mx(k-1+m);
31         B(k,1) = -Mx(N+k);
32     end;
33 end;
34
35 C = A\B;
36
37 p = [1 C'];
38 r = sort(roots(p));
39
40 for k = 1:N-1
41     % (k*) Considers proper moment scaling phenomenon
42     P(k,1) = k*Mx(k);
43     D(k,:) = (diff(r.^k));
44 end;
45
46 w_temp = D\P;
47 w = [w_temp; (0)];
48
49
50 %%% First order UT for the uniform distribution
51 [s_unif,w_unif] = UT_unif(n_points);
52
53 % scaling to get [-1,+1] uniform distribution
54 s_unif = s_unif ./ sqrt(3);
55 sigma = zeros(n_bars * n_points,1);
```

```

56 weights = zeros(n_bars * n_points,1);
57
58 % For each bar, scale the uniform distribution UT
59 for i = 1:n_bars
60     s_temp = (s_unif .* (r(i+1)-r(i))/2) + (r(i+1)+r(i))/2;
61     w_temp = w_unif .* w(i) .* (r(i+1)-r(i));
62
63     for j = 1:n_points
64         sigma((i-1)*n_points+j,1) = s_temp(j);
65         weights((i-1)*n_points+j,1) = w_temp(j);
66     end
67 end
68
69 end % End function

```

Listing II.9: Source code for the Extended UT computation of the normal distribution based on the direct method algorithm.

```

1 clear all
2 close all
3
4 %Reset RNG
5 rng(1001);
6 %rng(10121);
7
8 %UT samples
9 for M = 2:5
10 %M = 3;
11
12 %Monte-Carlo samples
13 N = 1e6;
14 N2 = M^3;
15 %N2 = 100;
16 N3 = 1000;
17 N4 = 10000;
18 N5 = 100000;
19
20 %Number of inverters in cascade
21 m = 3;
22
23 %Specified delay for each inverter
24 t0 = 1e-6;
25
26 %Variance of delay
27 s = 0.1e-6;
28
29 %Floating point filtering
30 threshold = 1e-6;
31
32 %%%
33 %Monte Carlo
34 %%%
35
36 delay = s * randn(N,m);
37 delay2 = s * randn(N2,m);
38 delay3 = s * randn(N3,m);
39 delay4 = s * randn(N4,m);
40 delay5 = s * randn(N5,m);
41
42 % System model
43 f = 1./(2*sum(t0+delay,2));
44 f2 = 1./(2*sum(t0+delay2,2));
45 f3 = 1./(2*sum(t0+delay3,2));
46 f4 = 1./(2*sum(t0+delay4,2));
47 f5 = 1./(2*sum(t0+delay5,2));
48
49 %t = linspace(1e4,1.2e5,200);
50
51 [B,X] = hist(f,200);
52 B = B/(sum(B)*(X(2)-X(1)));
53 [B2,X2] = hist(f2,200);
54 B2 = B2/(sum(B2)*(X2(2)-X2(1)));
55 [B3,X3] = hist(f3,200);
56 B3 = B3/(sum(B3)*(X3(2)-X3(1)));
57 [B4,X4] = hist(f4,200);
58 B4 = B4/(sum(B4)*(X4(2)-X4(1)));
59 [B5,X5] = hist(f5,200);

```

```

60 B5 = B5 / (sum(B5) * (X5(2) - X5(1)));
61
62 % Analytical result
63 %t=X;
64 y = (1./sqrt(2*pi*(m*(s^2))))*exp(-1./(2*(m*(s^2)))*(1./(2*X)-m*t0).^2)*1./(2*X.^2);
65
66
67 %%%
68 %UT
69 %%%
70 [d, w] = UT_norm(M);
71 d = t0 + s .* d;
72
73 % System model
74 count = 0;
75 delay_ut = zeros(M^3,2);
76 temp2 = zeros(M^3,3);
77 for i = 1:M
78     for j = 1:M
79         for k = 1:M
80             count = count + 1;
81             temp = [d(i) d(j) d(k)];
82             temp2(count,:) = temp;
83             delay_ut(count,1) = 1./(2*sum(temp));
84             delay_ut(count,2) = w(i)*w(j)*w(k);
85         end
86     end
87 end
88
89 % Combine probabilities
90 delay_ut = sortrows(delay_ut);
91 for i = 1:M^3-1
92     for j = i+1:M^3-1
93         if (delay_ut(j,1) - delay_ut(i,1)) < threshold
94             delay_ut(i,2) = delay_ut(i,2) + delay_ut(j,2);
95             delay_ut(j,:) = [0,0];
96         end
97     end
98 end
99
100 % Remove zeros
101 test_ut = delay_ut(:,1) ~= 0;
102 delay_ut = delay_ut(test_ut,:);
103
104 %%%
105 % CDF
106 %%%
107 count = 0;
108 T = linspace(X(1), X(length(X)), 1000);
109 cdf_mc = zeros(1,length(X));
110 cdf_mc2 = zeros(1,length(X2));
111 cdf_mc3 = zeros(1,length(X3));
112 cdf_mc4 = zeros(1,length(X4));
113 cdf_mc5 = zeros(1,length(X5));
114 cdf_ut = zeros(1,length(X));
115 f = sortrows(f);
116
117 %Monte Carlo
118 for i = T
119     count = count + 1;

```

```

120     cdf_mc(count) = sum(f < i)/N;
121     cdf_mc2(count) = sum(f2 < i)/N2;
122     cdf_mc3(count) = sum(f3 < i)/N3;
123     cdf_mc4(count) = sum(f4 < i)/N4;
124     cdf_mc5(count) = sum(f5 < i)/N5;
125 end
126
127 %UT
128 p_count = 0;
129 for k = 1:length(delay_ut)
130     p_count = p_count + delay_ut(k,2);
131     for i = 1:length(T)
132         if(T(i) > delay_ut(k,1))
133             cdf_ut(i) = p_count;
134         end
135     end
136 end
137
138
139 %%
140 %Moment calculation
141
142 N_moment = 20;
143 moments = zeros(N_moment,7);
144 error_moments = zeros(N_moment,7);
145
146 media_ut = sum(delay_ut(:,1).*delay_ut(:,2));
147
148 %Analytical results
149 %Moment function handler
150 M_func = @(x,c) x.^c.*((1./sqrt(2*pi*(m*(s^2))))*...
151     exp(-1./(2*(m*(s^2)))*(1./(2*x)-m*t0).^2)*1./(2*x.^2));
152 media_analytic = integral(@(x)M_func(x,1), 1e5, 3e5);
153 %Central moment function handler
154 M_func2 = @(x,c) (x - media_analytic).^c.*((1./sqrt(2*pi*(m*(s^2))))*...
155     exp(-1./(2*(m*(s^2)))*(1./(2*x)-m*t0).^2)*1./(2*x.^2));
156
157 moments(1,1) = media_analytic;
158 moments(1,2) = media_ut;
159 moments(1,3) = mean(f2);
160 moments(1,4) = mean(f3);
161 moments(1,5) = mean(f4);
162 moments(1,6) = mean(f5);
163 moments(1,7) = mean(f);
164
165 for k = 2:N_moment
166     moments(k,1) = integral(@(x)M_func2(x,k), 1e5, 3e5);
167     moments(k,2) = sum((delay_ut(:,1) - media_ut).^k .* delay_ut(:,2));
168     moments(k,3) = moment(f2,k);
169     moments(k,4) = moment(f3,k);
170     moments(k,5) = moment(f4,k);
171     moments(k,6) = moment(f5,k);
172     moments(k,7) = moment(f,k);
173 end
174
175 for k = 1:N_moment
176     error_moments(k,1) = moments(k,1);
177     error_moments(k,2) = abs((moments(k,1) - moments(k,2))/moments(k,1))*1;
178     error_moments(k,3) = abs((moments(k,1) - moments(k,3))/moments(k,1))*1;
179     error_moments(k,4) = abs((moments(k,1) - moments(k,4))/moments(k,1))*1;

```



```

180 error_moments(k,5) = abs((moments(k,1) - moments(k,5))/moments(k,1))*1;
181 error_moments(k,6) = abs((moments(k,1) - moments(k,6))/moments(k,1))*1;
182 error_moments(k,7) = abs((moments(k,1) - moments(k,7))/moments(k,1))*1;
183 end
184
185
186 N = 0;
187 N = N+1; figure(N);
188
189 %for k = 1:4
190 h(M-1) = plot(error_moments(:,2), '-');
191 hold on
192 end % sigma points for
193
194 h(1).Marker = '*';
195 h(2).Marker = 's';
196 h(3).Marker = '^';
197 h(4).Marker = '+';
198
199 grid on;
200 title('Moment_estimative_error_for_increasing_number_of_sigma_points');
201 xlabel('Moment_Order');
202 ylabel('Relative_error');
203 legend('2_Sigma_Points', '3_Sigma_Points', '4_Sigma_Points', ...
204        '5_Sigma_Points', 'Location', 'SouthEast');
205 %
206
207
208 N = N+1; figure(N);
209 plot(error_moments(:,2), '-+');
210 grid on;
211 hold on;
212 plot(error_moments(:,3), '-*_');
213 plot(error_moments(:,4), '-s');
214 plot(error_moments(:,6), '-^');
215 %plot(error_moments(:,7), '-^');
216 xlabel('Moment_Order');
217 ylabel('Relative_Error');
218 title('Relative_Error_for_Diferent_Techniques')
219 legend('UT:_5_sigma_points', 'MC:_125_points', ...
220        'MC:_1000_points', 'MC:_1e6_points', ...
221        'Location', 'NorthWest');
222 %%
223 %Plot
224
225 %subplot(2,1,1)
226 N = N+1; figure(N);
227 [ax1,~,~] = plotyy(X,y, delay_ut(:,1), delay_ut(:,2), 'plot', 'stem');
228 pbaspect(ax1(1), [3 1 3]);
229 pbaspect(ax1(2), [3 1 3]);
230 xlim(ax1(1), [X(1) X(length(X))]);
231 xlim(ax1(2), [X(1) X(length(X))]);
232 hold on
233 %plot(X,B2, 'r')
234 grid on
235 title('Probability_density/mass_function');
236 xlabel('Output_Frequency_[Hz]');
237 ylabel(ax1(1), 'Probability_Density');
238 ylabel(ax1(2), 'Discrete_Probability');
239 legend('Analytical', 'Unscented_Transform');

```

```

240
241 N = N+1; figure(N);
242 subplot(4,1,1);
243 plot(X,B2, X,y);
244 grid on;
245 title('Monte_Carlo_Convergence');
246 xlabel('Output_Frequency_[Hz]');
247 ylabel('Probability_Density');
248 legend([num2str(N2) '_points'])
249 subplot(4,1,2);
250 plot(X,B3,X,y);
251 grid on;
252 title('Monte_Carlo_Convergence');
253 xlabel('Output_Frequency_[Hz]');
254 ylabel('Probability_Density');
255 legend([num2str(N3) '_points'])
256 subplot(4,1,3);
257 plot(X,B4,X,y);
258 grid on;
259 title('Monte_Carlo_Convergence');
260 xlabel('Output_Frequency_[Hz]');
261 ylabel('Probability_Density');
262 legend([num2str(N4) '_points'])
263 subplot(4,1,4);
264 plot(X,B5,X,y);
265 grid on;
266 title('Monte_Carlo_Convergence');
267 xlabel('Output_Frequency_[Hz]');
268 ylabel('Probability_Density');
269 legend([num2str(N5) '_points'])
270
271 %subplot(2,1,2)
272 N = N+1; figure(N);
273 plot(T,cdf_mc,'-',T,cdf_ut,'-')
274 pbaspect([3 1 3]);
275 hold on
276 plot(T, cdf_mc2, 'r-')
277 grid on
278 title('Cumulative_distribution_function');
279 xlabel('Output_Frequency_[Hz]');
280 ylabel('Probability');
281 legend('Analytical', 'Unscented_Transform', ...
282       ['Monte_Carlo:_' num2str(N2) '_points'], ...
283       'Location', 'SouthEast');
284
285 N = N+1; figure(N);
286 plot(T,cdf_mc - cdf_ut, '-');
287 pbaspect([3 1 3]);
288 grid on;
289 title('Probability_estimative_error');
290 xlabel('Output_Frequency_[Hz]');
291 ylabel('Estimative_Error');

```

Listing II.10: Source code for the case study presented in Section 3.2.

```

1 %%
2 %Esse script foi utilizado para o paper "An alternative formulation
3 %for the unscented transform and its applications as Monte-Carlo
4 %alternatives" submetido ao Electronics Letters em junho de 2016.
5 %%
6
7 clear all
8 close all
9
10 %UT samples
11 M_bars = 3;
12 M_points = 3;
13 M = M_bars * M_points;
14
15 %Monte-Carlo samples
16 N1 = M.^3;
17 N2 = 1e6;
18
19 %Number of inverters in cascade
20 m = 3;
21
22 %Specified delay for each inverter
23 t0 = 1e-6;
24
25 %Variance of delay
26 s = 0.02e-6;
27
28 %Floating point filtering
29 threshold = 1e-9;
30
31 %%%
32 %Monte Carlo
33 %%%
34
35 delay1 = s * randn(N1,m);
36 delay2 = s * randn(N2,m);
37
38
39 % System model
40 f1 = 1./(2*sum(t0+delay1,2));
41 f2 = 1./(2*sum(t0+delay2,2));
42
43 %t = linspace(1e4,1.2e5,200);
44
45 [B1,X1] = hist(f1,200);
46 B1 = B1/(sum(B1)*(X1(2)-X1(1)));
47 [B2,X2] = hist(f2,200);
48 B2 = B2/(sum(B2)*(X2(2)-X2(1)));
49
50
51 % Analytical result
52 %t=X;
53 y = (1./sqrt(2*pi*(m*(s^2))))*exp(-1./(2*(m*(s^2)))*(1./(2*X1)-m*t0).^2)*1./(2*X1.^2);
54
55
56 %%%
57 %UT
58 %%%
59 [d_bars, w_bars] = UT_hist3_norm(M_bars, M_points);

```

```

60 d_bars = t0 + s .* d_bars;
61 [d_ut, w_ut] = UT_norm(M_bars);
62 d_ut = t0 + s .* d_ut;
63
64
65 %%
66 % System model simulation
67
68 count = 0;
69 delay_ut_bars = zeros(M^3,2);
70 temp2 = zeros(M^3,3);
71 for i = 1:M
72     for j = 1:M
73         for k = 1:M
74             count = count + 1;
75             temp = [d_bars(i) d_bars(j) d_bars(k)];
76             temp2(count,:) = temp;
77             delay_ut_bars(count,1) = 1./(2*sum(temp));
78             delay_ut_bars(count,2) = w_bars(i)*w_bars(j)*w_bars(k);
79         end
80     end
81 end
82
83 count = 0;
84 delay_ut = zeros(M_bars^3,2);
85 temp2 = zeros(M_bars^3,3);
86 for i = 1:M_bars
87     for j = 1:M_bars
88         for k = 1:M_bars
89             count = count + 1;
90             temp = [d_ut(i) d_ut(j) d_ut(k)];
91             temp2(count,:) = temp;
92             delay_ut(count,1) = 1./(2*sum(temp));
93             delay_ut(count,2) = w_ut(i)*w_ut(j)*w_ut(k);
94         end
95     end
96 end
97
98 % Combine probabilities
99 delay_ut_bars = sortrows(delay_ut_bars);
100 for i = 1:M^3-1
101     for j = i+1:M^3-1
102         if (delay_ut_bars(j,1) - delay_ut_bars(i,1)) < threshold
103             delay_ut_bars(i,2) = delay_ut_bars(i,2) + delay_ut_bars(j,2);
104             delay_ut_bars(j,:) = [0,0];
105         end
106     end
107 end
108
109 delay_ut = sortrows(delay_ut);
110 for i = 1:M_bars^3-1
111     for j = i+1:M_bars^3-1
112         if (delay_ut(j,1) - delay_ut(i,1)) < threshold
113             delay_ut(i,2) = delay_ut(i,2) + delay_ut(j,2);
114             delay_ut(j,:) = [0,0];
115         end
116     end
117 end
118
119 % Remove zeros

```

```

120 test_ut_bars = delay_ut_bars(:,1) ~= 0;
121 delay_ut_bars = delay_ut_bars(test_ut_bars,:);
122 test_ut = delay_ut(:,1) ~= 0;
123 delay_ut = delay_ut(test_ut,:);
124
125 %%
126 % CDF
127
128 T = linspace(X1(1), X1(length(X1)), 1000);
129 cdf_mc1 = zeros(1,length(X1));
130 cdf_mc2 = zeros(1,length(X1));
131 cdf_ut_bars = zeros(1,length(X1));
132 cdf_ut = zeros(1,length(X1));
133 f1 = sortrows(f1);
134
135 %Monte Carlo
136 count = 0;
137 for i = T
138     count = count + 1;
139     cdf_mc1(count) = sum(f1 < i)/N1;
140 end
141
142 count = 0;
143 for i = T
144     count = count + 1;
145     cdf_mc2(count) = sum(f2 < i)/N2;
146 end
147
148 %UT
149 p_count = 0;
150 for k = 1:length(delay_ut_bars)
151     p_count = p_count + delay_ut_bars(k,2);
152     for i = 1:length(T)
153         if(T(i) > delay_ut_bars(k,1))
154             cdf_ut_bars(i) = p_count;
155         end
156     end
157 end
158
159 p_count = 0;
160 for k = 1:length(delay_ut)
161     p_count = p_count + delay_ut(k,2);
162     for i = 1:length(T)
163         if(T(i) > delay_ut(k,1))
164             cdf_ut(i) = p_count;
165         end
166     end
167 end
168
169 %%
170 %Moment calculation
171
172 N_moment = 15;
173 moments = zeros(N_moment,5);
174 error_moments = zeros(N_moment,5);
175
176 media_ut = sum(delay_ut(:,1).*delay_ut(:,2));
177 media_ut_bars = sum(delay_ut_bars(:,1).*delay_ut_bars(:,2));
178
179 %Analytical results

```

```

180 %Moment function handler
181 M_func = @(x,c) x.^c.*(1./sqrt(2*pi*(m*(s^2))))*...
182         exp(-1./(2*(m*(s^2)))*(1./(2*x)-m*t0).^2)*1./(2*x.^2));
183 media_analytic = integral(@(x)M_func(x,1), 1e5, 3e5);
184 %Central moment function handler
185 M_func2 = @(x,c) (x - media_analytic).^c.*(1./sqrt(2*pi*(m*(s^2))))*...
186         exp(-1./(2*(m*(s^2)))*(1./(2*x)-m*t0).^2)*1./(2*x.^2));
187
188 moments(1,1) = media_analytic;
189 moments(1,2) = mean(f2);
190 moments(1,3) = mean(f1);
191 moments(1,4) = media_ut;
192 moments(1,5) = media_ut_bars;
193 for k = 2:N_moment
194     moments(k,1) = integral(@(x)M_func2(x,k), 1e5, 3e5);
195     moments(k,2) = moment(f2,k);
196     moments(k,3) = moment(f1,k);
197     moments(k,4) = sum((delay_ut(:,1) - media_ut).^k .* delay_ut(:,2));
198     moments(k,5) = sum((delay_ut_bars(:,1) - media_ut_bars).^k .* delay_ut_bars(:,2));
199 end
200
201 %moments(1,1) = 0;
202
203 for k = 1:N_moment
204     error_moments(k,1) = moments(k,1);
205     error_moments(k,2) = abs(moments(k,1) - moments(k,2))/moments(k,1)*100;
206     error_moments(k,3) = abs(moments(k,1) - moments(k,3))/moments(k,1)*100;
207     error_moments(k,4) = abs(moments(k,1) - moments(k,4))/moments(k,1)*100;
208     error_moments(k,5) = abs(moments(k,1) - moments(k,5))/moments(k,1)*100;
209 end
210
211 %%
212 % Yield estimatives
213 error_spec = 0.03;
214 m_sup = media_analytic + media_analytic*error_spec;
215 m_inf = media_analytic - media_analytic*error_spec;
216
217
218 Y_func = @(x) ((1./sqrt(2*pi*(m*(s^2))))*...
219             exp(-1./(2*(m*(s^2)))*(1./(2*x)-m*t0).^2)*1./(2*x.^2));
220 yield_analytic = integral(Y_func, m_inf, m_sup);
221
222 %Specification interval
223 Q = (T > m_inf)&(T < m_sup);
224 temp_T = T(Q);
225 temp_ut = cdf_ut(Q);
226 yield_ut = temp_ut(end) - temp_ut(1);
227 temp_ut_bars = cdf_ut_bars(Q);
228 yield_ut_bars = temp_ut_bars(end) - temp_ut_bars(1);
229 temp_mcl = cdf_mcl(Q);
230 yield_mcl = temp_mcl(end) - temp_mcl(1);
231 temp_mc2 = cdf_mc2(Q);
232 yield_mc2 = temp_mc2(end) - temp_mc2(1);
233
234 yield_results = [yield_analytic; yield_mc2; yield_mcl; yield_ut; yield_ut_bars];
235
236 %%
237 %Plot Results
238 figure(1);
239 subplot(2,1,1);

```

```

240 plotyy(X1,B1, delay_ut_bars(:,1), delay_ut_bars(:,2), 'plot', 'stem');
241 hold on
242 plot(X2,B2);
243 grid on
244 title('Probability_density/mass_function');
245 xlabel('Frequency_[Hz]');
246 subplot(2,1,2)
247 plot(T,cdf_mc1, T, cdf_mc2);
248 hold on
249 %plot(delay_ut(:,1),cumsum(delay_ut(:,2)), '-');
250 plot(T,cdf_ut_bars);
251 plot(T,cdf_ut);
252 grid on
253 title('Cumulative_distribution_function');
254 xlabel('Frequency_[Hz]');
255
256 figure(2);
257 subplot(3,1,1);
258 plot(T,cdf_mc1, T,cdf_mc2);
259 subplot(3,1,2)
260 plot(T,cdf_ut, T,cdf_mc2);
261 subplot(3,1,3);
262 plot(T,cdf_ut_bars, T,cdf_mc2);
263
264 figure(3);
265 %subplot(2,1,1)
266 plot(T,cdf_ut, T,cdf_mc2);
267 pbaspect([3 1 3]);
268 %hold on;
269 %stem(temp_T(1),temp_ut(1));
270 %stem(temp_T(end),temp_ut(end));
271 title('Cumulative_distribution_function_-_UT');
272 xlabel('Frequency_[Hz]');
273 ylabel('Probability');
274 grid on;
275 %subplot(2,1,2);
276
277 figure(4)
278 plot(T,cdf_ut_bars, T,cdf_mc2);
279 pbaspect([3 1 3]);
280 hold on;
281 %stem(temp_T(1),temp_ut_bars(1));
282 %stem(temp_T(end),temp_ut_bars(end));
283 title('Cumulative_distribution_function_-_ExUT');
284 xlabel('Frequency_[Hz]');
285 ylabel('Probability');
286 grid on;
287
288 figure(5);
289 plot(T,cdf_mc2 - cdf_ut, '-');
290 hold on
291 plot(T,cdf_mc2 - cdf_ut_bars, '-');
292 pbaspect([3 1 3]);
293 grid on;
294 title('Probability_estimative_error');
295 xlabel('Output_Frequency_[Hz]');
296 ylabel('Estimative_Error');
297
298 % figure(6);
299 % plot(T,cdf_mc2 - cdf_ut_bars, '-');

```

```
300 % pbaspect([3 1 3]);
301 % grid on;
302 % title('Probability estimative error');
303 % xlabel('Output Frequency [Hz]');
304 % ylabel('Estimative Error');
```

Listing II.11: Source code for the case study presented in Section 3.3.

II.5 Arcsine Quantizer

The following function computes the arcsine quantizer based on the UT.

```
1 function [outCode, outValue] = quant_arcsine(in, N, DR)
2 %QUANTIZER Quantizes an input stream of samples with
3 %   the arcsine quantizer.
4 %   input = input samples vector.
5 %   N = ADC resolution.
6 %   DR = vector specifying the input range.
7
8 % Get the UT. The weights will be used to get the
9 % threshold levels and the sigma points vector is the codebook.
10 [s,w] = UT_arcsine(0, 1, 2^N);
11
12 % Threshold levels.
13 th = (sin(pi/2*cumsum(w))).^2;
14 th(end) = [];
15 codebook = s;
16 % Map to the desired range.
17 th = range(DR)*(th-1/2) + (DR(1)+DR(2))/2;
18
19 % Quantize the input.
20 [code, value] = quantizer(in, th, codebook);
21
22 % Output values.
23 outCode = code;
24 outValue = value;
25 end
```

Listing II.12: Arcsine quantizer based on the UT.

```
1 function [output_code] = quant_ADC(input, N)
2 %QUANT_ADC Returns the output of an 2^N quantizer.
3 %   input = [0,1]
4 %   output_code = [0,1]
5
6 LSB = 1/(2^N);
7
8 code = zeros(1,length(input));
9
10 % Quantizer process
11 for j = 1:length(input)
12     for k = 1:(2^N)-1
13         if(input(j) > k*LSB)
14             code(j) = k*LSB;
15         else
16             break;
17         end
18     end
19 end
20
21 % Output vector
22 output_code = code + 0.5*LSB;
23
24 end
```

Listing II.13: Linear quantizer.

```

1  % Script used to plot the arcsine quantizer features.
2
3  clear all
4  close all
5
6  dither = false;    % Insert dithering?
7  print_plots = false; % Print results?
8  plot_charcurve = true; % Plot characteristic curves?
9  plot_transient = true; % Plot transient curves?
10 plot_error = true; % Plot quantization error?
11 plot_signalfft = true; % Plot signal FFT?
12 plot_noisefft = true; % Plot noise FFT?
13
14 M = 2^20;          % Number of input points
15 N = 4;            % ADC resolution
16 LSB = 1/(2^N);
17 fx = 1e2;         % Input Frequency
18 fs = 1e8;         % Sampling frequency
19 band = 1e5;       % Low Pass Filter Bandwidth
20 NFFT = 2^nextpow2(M); %FFT depth
21
22 cycles = floor(fx/fs*NFFT)+1; % Number of input cycles
23 cycles = max(primes(cycles)); % to guarantee coherent FFT
24 fx = fs*cycles/NFFT;    % Actual input frequency
25
26 t = linspace(0, (M-1)/fs, M); % Time vector
27 A = 1;                    % Input amplitude
28
29 % Insert dithering?
30 if dither == true
31     A_dither = (1/3)*LSB;
32 else
33     A_dither = 0;
34 end
35
36 % Input signal
37 i = A.*sin(2*pi*fx*t) + A_dither*randn(1,length(t));
38
39 % Quantization
40 out_ADC = 2*(quant_ADC(i/2 + 0.5, N) - 0.5); %Linear
41 out_UT_arcsine = 2*(quant_UT_sin(i/2 + 0.5, N) - 0.5);%Arcsine
42
43 %% Sort the results for characteristic curves
44 ADC = sortrows([i' out_ADC']);
45 UT_arcsine = sortrows([i' out_UT_arcsine']);
46
47 %% Signal FFT
48 fft_IN = fft(i, NFFT)/M;          %Input signal FFT
49 fft_ADC = fft(out_ADC, NFFT)/M;    %Linear quantizer FFT
50 fft_UT_arcsine = fft(out_UT_arcsine, NFFT)/M; %Arcsine quantizer FFT
51
52 %% Quantization noise FFT
53 fft_qe_ADC = fft(i - out_ADC, NFFT)/M; %Linear quantizer
54 fft_qe_UT_arcsine = fft(i - out_UT_arcsine, NFFT)/M; %Arcsine quantizer
55 f = fs/2*linspace(0,1,NFFT/2+1); % Frequency axis
56
57
58 % Start figuring
59 fig = 1;

```

```

60
61 ov_plot = [];
62 SNDR_linear_plot = [];
63 SNDR_UT_plot = [];
64
65 % Characteristic curve
66 if plot_charcurve == true
67     figure(fig); fig = fig + 1;
68     plot(ADC(:,1), ADC(:,2));
69     hold on;
70     plot(ADC(:,1), ADC(:,1));
71     grid on;
72     title('Characteristic_curve_-_Linear_quantizer');
73     xlabel('Input_level');
74     ylabel('Output_level');
75     xlim([-1 1]);
76     ylim([-1 1]);
77     axis square
78     if print_plots == true
79         print('-~/Desktop/char_curve_linear', '-dpdf');
80     end
81
82     figure(fig); fig = fig + 1;
83     plot(UT_arcsine(:,1), UT_arcsine(:,2));
84     hold on;
85     plot(ADC(:,1), ADC(:,1));
86     grid on;
87     title('Characteristic_curves_-_Arcsine_quantizer');
88     xlabel('Input_level');
89     ylabel('Output_level');
90     xlim([-1 1]);
91     ylim([-1 1]);
92     axis square
93     if print_plots == true
94         print('-~/Desktop/char_curve_arcsine', '-dpdf');
95     end
96 end % Characteristic curve
97
98 % Transient curves
99 if plot_transient == true
100     figure(fig); fig = fig + 1;
101     plot(t, i, t, out_ADC);
102     grid on;
103     axis([0 1/fx -1.1 1.1]);
104     axis square
105     title('Transient_curve_-_Linear_quantizer');
106     xlabel('Time_[s]');
107     ylabel('Amplitude');
108     if print_plots == true
109         print('-~/Desktop/tran_linear', '-dpdf');
110     end
111
112     figure(fig); fig = fig + 1;
113     plot(t, i, t, out_UT_arcsine);
114     grid on;
115     axis([0 1/fx -1.1 1.1]);
116     axis square
117     title('Transient_curve_-_Arcsine_quantizer');
118     xlabel('Time_[s]');
119     ylabel('Amplitude');

```

```

120     if print_plots == true
121         print('-~/Desktop/tran_arcsine', '-dpdf')
122     end
123 end % Transient curves
124
125 % Quantization error curve
126 if plot_error == true
127     figure(fig); fig = fig + 1;
128     plot(t, i - out_ADC);
129     xlim([0 1/fx]);
130     pbaspect([3 1 3]);
131     grid on;
132     xlabel('Input_level');
133     ylabel('Quantization_Error');
134     title('Quantization_error_-_Linear_Quantizer');
135     if print_plots == true
136         print('-~/Desktop/error_linear', '-dpdf')
137     end
138
139     figure(fig); fig = fig + 1;
140     plot(t, i - out_UT_arcsine);
141     xlim([0 1/fx]);
142     pbaspect([3 1 3]);
143     grid on;
144     xlabel('Input_level');
145     ylabel('Quantization_Error');
146     title('Quantization_error_-_Arcsine_Quantizer');
147     if print_plots == true
148         print('-~/Desktop/error_arcsine', '-dpdf')
149     end
150 end % Quantization error curve
151
152 % Signal spectrum
153 if plot_signalfft == true
154     figure(fig); fig = fig + 1;
155     [y,bin] = max(abs(fft_ADC));
156     hh = stem(f, 10*log10(2*abs(fft_ADC(1:NFFT/2+1)+realmin)), ...
157         'o','BaseValue', -50, 'color', 'blue');
158     axis([1e2 f(500*bin) -50 0]);
159     hb = get(hh,'Baseline');
160     set(hb,'Visible','off')
161     set(gca,'xscal','log')
162     title('Signal_FFT_-_Linear_quantizer');
163     xlabel('Frequency_[Hz]');
164     ylabel('Amplitude_[dB]');
165     grid on;
166     pbaspect([3 1 3])
167     if print_plots == true
168         print('-~/Desktop/fft_signal_linear', '-dpdf')
169     end
170
171     figure(fig); fig = fig + 1;
172     [y,bin] = max(abs(fft_UT_arcsine));
173     hh = stem(f, 10*log10(2*abs(fft_UT_arcsine(1:NFFT/2+1)+realmin)), ...
174         'o', 'BaseValue', -50, 'color', 'red');
175     axis([1e2 f(500*bin) -50 0]);
176     hb = get(hh,'Baseline');
177     set(hb,'Visible','off')
178     set(gca,'xscal','log')
179     grid on;

```

```

180     title('Signal_FFT_-_Arcsine_quantizer');
181     xlabel('Frequency_[Hz]');
182     ylabel('Amplitude_[dB]');
183     pbaspect([3 1 3])
184     if print_plots == true
185         print('~\Desktop/fft_signal_arcsine', '-dpdf')
186     end
187 end % Signal spectrum
188
189
190 % Quantization noise spectrum
191 if plot_signalfft == true
192     figure(fig); fig = fig + 1;
193     hh = stem(f, 10*log(2*abs(fft_qe_ADC(1:NFFT/2+1))), ...
194         'o', 'BaseValue', -120, 'color', 'blue');
195     axis([1e2 f(500*bin) -120 0]);
196     hb = get(hh, 'Baseline');
197     set(hb, 'Visible', 'off')
198     set(gca, 'xscal', 'log')
199     grid on;
200     title('Quantization_noise_FFT_-_Linear_quantizer');
201     xlabel('Frequency_[Hz]');
202     ylabel('Amplitude_[dB]');
203     pbaspect([3 1 3])
204     if print_plots == true
205         print('~\Desktop/fft_error_linear', '-dpdf')
206     end
207
208     figure(fig); fig = fig + 1;
209     hh = stem(f, 10*log(2*abs(fft_qe_UT_arcsine(1:NFFT/2+1))), ...
210         'o', 'BaseValue', -120, 'color', 'red');
211     axis([1e2 f(500*bin) -120 0]);
212     hb = get(hh, 'Baseline');
213     set(hb, 'Visible', 'off')
214     set(gca, 'xscal', 'log')
215     grid on;
216     title('Quantization_noise_FFT_-_Arcsine_quantizer');
217     xlabel('Frequency_[Hz]');
218     ylabel('Amplitude_[dB]');
219     pbaspect([3 1 3])
220     if print_plots == true
221         print('~\Desktop/fft_error_arcsine', '-dpdf')
222     end
223 end % Quantization noise spectrum

```

Listing II.14: Source code for the analysis of the arcsine quantizer presented in Section 4.3.

```

1 clear all
2 close all
3
4 % Number of input points
5 M = 2^20;
6
7 % ADC resolution
8 N = 10;
9 LSB = 1/(2^N);
10
11 % Input Signal Frequency
12 fx = 1e3;
13
14 % Sampling Frequency
15 fs = 1e5;
16
17 % FFT setup
18 NFFT = 2^nextpow2(M);
19 cycles = floor(fx/fs*NFFT)+1;
20 cycles = max(primes(cycles));
21 fx = fs*cycles/NFFT;
22 t = linspace(0, (M-1)/fs, M);
23
24 % Sine wave
25 A = 0.5;
26 %A=1;
27 %A_dither = 0;
28 A_dither = (1/2)*LSB;
29 i = A.*sin(2*pi*fx*t) + A_dither*randn(1,length(t));
30
31 % Quantizes
32 out_ADC = quant_ADC(i + 0.5, N) - 0.5;
33 out_UT_arcsine = quant_UT_sin(i + 0.5, N) - 0.5;
34
35 i = i*2;
36 out_ADC = out_ADC*2;
37 out_UT_arcsine = out_UT_arcsine*2;
38
39 % Sort the results
40 ADC = sortrows([i' out_ADC']);
41 UT_arcsine = sortrows([i' out_UT_arcsine']);
42
43 %% Transient curves
44 fig = 1;
45
46 % Signal FFT
47 Fs = M;
48 fft_IN = fft(i*2, NFFT)/NFFT;
49 fft_ADC = fft(out_ADC*2, NFFT)/M;
50 fft_ADC_filter = fft(out_ADC_filter*2, NFFT)/M;
51 fft_UT_arcsine = fft(out_UT_arcsine*2, NFFT)/M;
52 fft_UT_arcsine_filter = fft(out_UT_arcsine_filter*2, NFFT)/M;
53 fft_UT_input = fft(i*2, NFFT)/M;
54 f = fs/2*linspace(0,1,NFFT/2+1);
55
56 figure(fig); fig = fig + 1;
57 temp_linear = (2*abs(fft_ADC(1:NFFT/2+1)+realmin));
58 [y,bin] = max(temp_linear);
59 temp_linear(bin) = realmin;

```

```

60 NDR_linear = 10*log10((sum(temp_linear.^2)));
61 SNDR_linear = 10*log10((y.^2)/(sum(temp_linear.^2)))
62 ENOB_linear = (SNDR_linear-1.76)/6.02
63 semilogx(f, 10*log10(2*abs(fft_ADC(1:NFFT/2+1)+realmin)), 'b');
64 axis([1e1 max(f) -60 0]);
65 title('Signal_FFT_-_Linear_quantizer');
66 xlabel('Frequency_[Hz]');
67 ylabel('Amplitude_[dB]');
68 grid on;
69 pbaspect([3 1 3])
70 %print('~/Desktop/fft_signal_linear', '-dpdf')
71
72 figure(fig); fig = fig + 1;
73 temp_UT = (2*abs(fft_UT_arcsine(1:NFFT/2+1)+realmin));
74 [y,bin] = max(temp_UT);
75 temp_UT(bin) = realmin;
76 NDR_UT = 10*log10((sum(temp_UT.^2)));
77 SNDR_UT = 10*log10((y.^2)/(sum(temp_UT.^2)))
78 ENOB_UT = (SNDR_UT-1.76)/6.02
79 semilogx(f, 10*log10(2*abs(fft_UT_arcsine(1:NFFT/2+1)+realmin)), 'r');
80 axis([1e1 max(f) -60 0]);
81 grid on;
82 title('Signal_FFT_-_Arcsine_quantizer');
83 xlabel('Frequency_[Hz]');
84 ylabel('Amplitude_[dB]');
85 pbaspect([3 1 3])
86 %print('~/Desktop/fft_signal_arcsine', '-dpdf')
87 % close all

```

Listing II.15: Source code for the data shown in Figure 4.5.

```

1 clear all
2 close all
3 format compact
4 tic
5
6 dither = true;      % Insert dithering?
7 print_plots = false; % Print results?
8 plot_THD = true;   % Plot THD?
9 plot_SINAD = true; % Plot SINAD?
10 plot_SFDR = true; % Plot SFDR?
11 plot_SNR = true;  % Plot SNR?
12
13 harmonics = 20;    % Harmonics to consider in calculations
14
15 M = 2^20;          % Number of input points
16 fx = 1e2;         % Input Frequency
17 fs = 1e5;         % Sampling frequency
18 band = 1e5;       % Low Pass Filter Bandwidth
19 NFFT = 2^nextpow2(M); %FFT depth
20 cycles = floor(fx/fs*NFFT)+1; % Number of input cycles
21 cycles = max(primes(cycles)); % to guarantee coherent FFT
22 fx = fs*cycles/NFFT; % Actual input frequency
23
24 t = linspace(0, (M-1)/fs, M); % Time vector
25 A = 1;            % Input amplitude
26
27 m = 9;            % Loop depth
28 resolution_plot = zeros(1,m);
29 thd_linear_plot = zeros(1,m);
30 thd_UT_plot = zeros(1,m);
31 snr_linear_plot = zeros(1,m);
32 snr_UT_plot = zeros(1,m);
33 sinad_linear_plot = zeros(1,m);
34 sinad_UT_plot = zeros(1,m);
35 sfdr_linear_plot = zeros(1,m);
36 sfdr_UT_plot = zeros(1,m);
37
38 for k = (1:m)
39     N = 3 + k      %ADC resolution
40     LSB = 1/(2^N);
41     % Insert dithering?
42     if dither == true
43         A_dither = (1/3)*LSB;
44     else
45         A_dither = 0;
46     end
47
48     % Input signal
49     i = A.*sin(2*pi*fx*t) + A_dither*randn(1,length(t));
50
51     % Quantization
52     out_ADC = 2*(quant_ADC(i/2 + 0.5, N) - 0.5); %Linear
53     out_UT_arcsine = 2*(quant_UT_sin(i/2 + 0.5, N) - 0.5);%Arcsine
54
55     % Periodogram
56     [s_ADC, f_UT] = periodogram(out_ADC, rectwin(length(out_ADC)), ...
57         length(out_ADC), fs, 'power');
58     rbw_ADC = enbw(hamming(length(out_ADC)), fs);
59     [s_UT, f_ADC] =

```



```

60 periodogram(out_UT_arcsine,rectwin(length(out_UT_arcsine)), ...
61     length(out_UT_arcsine),fs,'power');
62 rbw_UT = enbw(hamming(length(out_UT_arcsine)),fs);
63
64 % Resolution
65 resolution_plot(k) = N;
66
67 %THD
68 thd_linear_plot(k) = thd(s_ADC, f_ADC, rbw_ADC, harmonics, 'power');
69 thd_UT_plot(k) = thd(s_UT, f_UT, rbw_UT, harmonics, 'power');
70
71 %SNR
72 snr_linear_plot(k) = snr(s_ADC, f_ADC, rbw_ADC, harmonics, 'power');
73 snr_UT_plot(k) = snr(s_UT, f_UT, rbw_UT, harmonics, 'power');
74
75 %SINAD
76 sinad_linear_plot(k) = sinad(s_ADC, f_ADC, rbw_ADC, 'power');
77 sinad_UT_plot(k) = sinad(s_UT, f_UT, rbw_UT, 'power');
78
79 %SFDR
80 [y,bin] = max(s_ADC); %Locate signal
81 sfdr_linear_plot(k) = sfdr(s_ADC(1:harmonics*bin), ...
82     f_ADC(1:harmonics*bin), 'power');
83 sfdr_UT_plot(k) = sfdr(s_UT(1:harmonics*bin), ...
84     f_UT(1:harmonics*bin), 'power');
85 end %Resolution loop
86
87 % Start figuring
88 fig = 1;
89
90 if plot_THD
91     figure(fig); fig = fig + 1;
92     plot(resolution_plot, thd_linear_plot, '-^', ...
93         resolution_plot, thd_UT_plot, '-s'),
94     grid on
95     xlabel('Quantizer_resolution_[bits]');
96     ylabel('THD_[dB]');
97     title('Total_Harmonic_Distortion_-_10th_Harmonic');
98     legend('Linear_quantizer', 'Arcsine_quantizer');
99     if print_plots == true
100         print('-/Desktop/matlab_THD', '-dpdf');
101     end
102 end
103
104 if plot_SNR
105     figure(fig); fig = fig + 1;
106     plot(resolution_plot, snr_linear_plot, '-^', ...
107         resolution_plot, snr_UT_plot, '-*', ...
108         resolution_plot, 6.02*resolution_plot+1.76, '-s'),
109     grid on
110     xlabel('Quantizer_resolution_[bits]');
111     ylabel('SNR_[dB]');
112     title('Signal_to_Noise_Ratio_-_10th_Harmonic');
113     legend('Linear_quantizer', 'Arcsine_quantizer', ...
114         'Theoretical', 'Location', 'SouthEast');
115     if print_plots == true
116         print('-/Desktop/matlab_SNR', '-dpdf');
117     end
118 end
119

```

```

120 if plot_SINAD
121     figure(fig); fig = fig + 1;
122     plot(resolution_plot, sinad_linear_plot, '-^', ...
123         resolution_plot, sinad_UT_plot, '-*', ...
124         resolution_plot, 6.02*resolution_plot+1.76, '-s'),
125     grid on
126     xlabel('Quantizer_resolution_[bits]');
127     ylabel('SINAD_[dB]');
128     title('Signal_to_Noise_and_Distortion_Ratio_-_10th_Harmonic');
129     legend('Linear_quantizer', 'Arcsine_quantizer', ...
130         'Theoretical', 'Location', 'SouthEast');
131     if print_plots == true
132         print('-/Desktop/matlab_SINAD', '-dpdf');
133     end
134 end
135
136 if plot_SFDR
137     figure(fig); fig = fig + 1;
138     plot(resolution_plot, sfdr_linear_plot, '-^', ...
139         resolution_plot, sfdr_UT_plot, '-*', ...
140         resolution_plot, 9.03*resolution_plot, '-s'),
141     grid on
142     xlabel('Quantizer_resolution_[bits]');
143     ylabel('SFDR_[dB]');
144     title('Spurious_Free_Dynamic_Range_-_10th_Harmonic');
145     legend('Linear_quantizer', 'Arcsine_quantizer', ...
146         'Theoretical', 'Location', 'SouthEast');
147     if print_plots == true
148         print('-/Desktop/matlab_SFDR', '-dpdf');
149     end
150 end
151
152 toc
153 format; % Reset formatting

```

Listing II.16: Source code for the data shown in Figures 4.8 and 4.9.