Departamento de Electrónica, Sistemas e Informática

MAESTRÍA EN DISEÑO ELECTRÓNICO

ADAPTIVE FUNCTION SEGMENTATION METHODOLOGY FOR RESOURCES OPTIMIZATION OF HARDWARE-BASED FUNCTION EVALUATORS

Tesis que para obtener el grado de

MAESTRO EN DISEÑO ELECTRÓNICO

Presenta: Juan Martín Trejo Arellano

Director: Dr. Omar Humberto Longoria Gándara
Co-Director: Dr. Javier Vázquez Castillo

San Pedro Tlaquepaque, Jalisco. Febrero del 2017.

Agradecimientos

Agradezco a dios por la vida, el gran amor y fortaleza que me da día a día para afrontar los retos que me llenan de satisfacción y jubilo. Agradezco infinitamente a todas las personas con quienes me he encontrado a lo largo de mi vida, quienes han sido un aliciente y me han impulsado crecer como persona y como profesionista.

Agradezco a mi amada esposa María Isabel Baena Amador, por ser mi fiel compañera, amiga y colega quien me ha apoyado de forma incondicional en los que alguna vez fueron momentos muy difíciles; los cuales ahora, recapitulando, gracias a su amor, cuidado y comprensión, se han convertido en momentos memorables llenos de aprendizaje y satisfacción.

Agradezco a mi padre, Jesús Trejo Ortiz, quien con su ejemplo me ha enseñado que las cosas que realmente importan en esta vida se logran por medio de la tenacidad y trabajo arduo. Agradezco a mi madre, Yolanda Arellano Almeida, por los sacrificios que ha realizado para traerme a éste mundo y criarme en un hogar lleno de buenos valores y quien siempre se ha dado un momento para brindarme un abrazo, un beso y las palabras de aliento que en más de una ocasión he necesitado para afrontar los retos de la vida bajo una perspectiva más positiva y enriquecedora. A mis hermanas Minerva Susana, Aurora Berenice, María Lú, y Erika Cecilia, por todos los momentos me han permitido compartir y el cariño que me han brindado.

Agradezco a mi suegra, la señora Teresa Amador Ortiz, quien me ha brindado la confianza de ser el guardián de uno de sus más grandes tesoros y quien con su ejemplo me ha mostrado que las más grandes barreras en nuestra vida se superan con amor incondicional, sacrificio y tenacidad. A mis cuñados Rigel, Raúl y a mi cuñada Lileana, quienes con amor han acompañaron a mi esposa durante su vida y que ahora de igual forma me han aceptado como parte de su hermosa familia.

Agradezco a mis asesores, el Dr. Omar Humberto Longoria Gándara y el Dr. Javier Vázquez Castillo quienes me han brindado su confianza y quienes con gran paciencia y entusiasmo me guiaron, compartido sus conocimientos para el desarrollo de mi tesis. Agradezco al Dr. Roberto Carrasco Álvarez quien dedico gran cantidad de su tiempo para la revisión de la redacción y el sustento matemático de los artículos publicados como parte de este trabajo de tesis.

Agradezco al Instituto Tecnológico de Estudios Superiores de Occidente (ITESO) por el acceso a la educación y los valores que me instruyó durante el transcurso de mi maestría para poder consolidarme como un ciudadano y profesionista altamente comprometido con el desarrollo de la tecnología en México.

A SEP-PRODEP 2014 y CONACYT por el apoyo económico otorgado bajo el fondo CB-241272, para el desarrollo del proyecto que se aborda en éste trabajo de tesis y las publicaciones derivadas.

Abstract

This thesis presents a new adaptive function segmentation methodology (AFSM), for the evaluation of mathematical functions through piecewise polynomial approximation (PPA) methods. This methodology is planned to be employed for the development of an efficient hardware-based channel emulator in future development steps of the current project. In contrast to state-of-art segmentation methodologies, which applicability is limited because these are highly dependent on the function shape and require significant intervention from the user to setup appropriately the algorithm, the proposed segmentation methodology is flexible and applicable to any continuous function within an evaluation interval. Through the analysis of the first and second order derivatives, the methodology becomes aware of the function shape and adapts the algorithm behavior accordingly.

The proposed segmentation methodology aims towards hardware architectures of limited resources that resort to fixed-point numeric representation where the hardware designer should make a compromise between resources consumption and output accuracy. An optimization algorithm is implemented to assist the user in searching the best segmentation parameters that maximize the outcome of the design trade-offs for a given signal-to-quantization-noise ratio requirement. When compared to state-of-the-art segmentation methodologies, the proposed AFSM delivers better performance of approximation for the hardware-based evaluation of transcendental functions given that fewer segments and consequently fewer hardware resources are required.

Table of Content

1 Introduction

The development and implementation of modern wireless communication systems are highly complex tasks that require exhaustive simulation during the design and verification of the building blocks to develop a system that is cost effective and performs reliably under a broad set of operational conditions. Under these circumstances, software-based simulation tools are not adequate given the excessive amount of time required to complete numerically intensive types of simulations.

The physical layer of a wireless communication system can be broken down into two blocks, the baseband section, and the Tx/Rx RF front-end section. Although both blocks present intrinsic undesired characteristics that limit the overall performance of the system, the greatest impact is imposed by the degrading propagation phenomena of the communication channel, such as scattering, reflections, diffraction and attenuation [1]. These propagation phenomena can be modeled as noise with certain statistical properties, which can be efficiently imprinted to the signal through hardware-based channel emulators.

In this sense, the bit error rate (BER) over the desired range of signal-to-noise ratio (SNR) is the metric employed to evaluate the performance of the baseband wireless receiver under test. The BER to SNR characteristic is generated through pervasive Monte Carlo simulations that can take several days to weeks or even months if performed through software-based simulators [1]. On the other hand, the verification of the wireless communication systems' physical layer can be sped up several orders of magnitude if highly flexible and efficient wireless channel emulators are implemented in hardware using field programmable gate arrays (FPGA) or application specific integrated circuits (ASIC). Consequently, a broad set of configurations and transmission environments such as indoor, urban, suburban, rural, and mobile, can be tested under controlled conditions that warrant the repeatability of subsequent measurements [2]; something that is nearly impossible to achieve through in-the-field testing methods.

1. INTRODUCTION

If the reader is interested in obtaining the MATLAB code of the implementation presented in this study, please feel free to send an email request at MD687149@iteso.mx.

1.1 Motivation

The Nakagami, Suzuki, and Weibull channel emulators are noise generators widely used for generating stochastic processes with specific characteristics associated with the different communication channels or environments. The Weibull processes are utilized to model power variation of the signal multi-paths in vehicle-to-vehicle (V2V) applications [2] under urban environments (land-mobile channels) [3]. The Suzuki processes are suitable to simulate a mobile wireless channel affected by fading (small-scale process) and shadowing (large-scale process). Additionally, the Suzuki processes are considered to be more precise for modeling channels in urban environments where the specular component or line of sight (LoS) is not present [4]. Finally, the Nakagami processes are used to represent a channel where multiple Rayleigh processes are present (Channels with great temporal dispersion) such as in V2V communication channels.

Wireless channel models implement mathematical expressions and transcendental functions that are evaluated to generate the statistical channel noise description when carrying out the testing and simulation of a wireless communication system. In general, one of the simplest methods to evaluate a transcendental function is through look-up tables (LUT); where a broad set of output values obtained from a fine-grained pre-evaluation of the function are stored in advanced in the LUT, and then retrieved back according to the input argument of the functions. However, the downside of this evaluation method is that the hardware resources occupied by the LUT increase exponentially along with increments in the accuracy requirements of the output [5].

With the objective to reduce the hardware resources footprint, this work proposes the evaluation of the transcendental functions through piece-wise polynomial approximation methods (PPA) where the function subject to evaluation is segmented out, and each segment is approximated using a low-degree polynomial. Consequently, through this evaluation method, the LUT only stores the coefficients of the polynomial that best fit each of the segments that encompass the evaluation domain of the functions. However, the output accuracy achieved through PPA methods heavily depends on the segmentation strategy employed to segment out the function at hand; in other words, the quality of approximation of the original function is determined by both, the location of the segments boundaries as well as the number of segments required.

1.2 State of the Art

Modern digital signal processing algorithms use high complexity building blocks, which are associated with the evaluation of transcendental functions. In wireless communication channel modeling, the channel emulation is carried out using models based on sum-of-cissoids (complex exponentials), where the accuracy of evaluation of the $\sin(\cdot)$ and $\cos(\cdot)$ functions within the models is a primary concern [6]. As an example, in Weibull fading channel emulators, which are widely used for modeling V2V channels [7], the hardware implementation is significantly complex due to the evaluation of $\ln(\cdot)$, $\sqrt{\cdot}$, $1/x$, and $\exp(\cdot)$ functions [7], [3]. Likewise, the efficient hardware implementation of algorithms based on algebraic matrix operations such as QR decomposition (QRD), commonly used for matrix inversion, is highly sensitive to the accuracy of evaluation of the function $\sqrt{\cdot}$ and $1/x$ [8].

Currently, there are several methods for the evaluation of transcendental functions. Although some of them offer certain advantages, they are also subject to disadvantages that make them unsuitable for applications that require high accuracy and substantial computing throughput. The iterative methods such as CORDIC (COordinate Rotation DIgital Computer) allow the evaluation of transcendental functions [9], [10], [11] and [12] in a flexible manner. However, a significant drawback that limits the development of hardware architectures for real-time computing applications is that the output accuracy of the iterative methods is highly dependent on the number of iteration that the algorithm is executed. An alternative methodology for evaluating transcendental functions is via look-up tables, [3] and [13]; this is arguably the simplest and easiest way to implement function evaluation blocks; however, the amount of memory needed for allocating the function values increases significantly with increments on the output accuracy requirement.

1. Introduction

On the other hand, PPA is an alternative method for evaluating transcendental functions. It offers flexible design trade-offs between computing speed, area, output accuracy, and hardware architecture reuse because the design of the polynomials evaluator does not change across functions. Approximating a function using PPA methods requires the input evaluation interval to be partitioned into multiple segments. Each of these segments is approximated using a low-degree polynomial, which is addressed through the hardware polynomial evaluator according to the input values of the function. In this sense, the accuracy achieved using the PPA approximation methodology significantly depends on the segmentation methodology utilized; i.e., sizable approximation errors might be introduced when an inadequate segmentation strategy is employed, resulting in reduced signal-to-quantization-noise ratio (SQNR) performance of the function evaluation block.

Today, the most popular segmentation methodology for PPA is called hierarchical segmentation method (HSM), [14], which embed the more basic segmentation methodologies known as uniform and non-uniform-by-powers-of-two. In principle, any function could be segmented out through these methodologies; however, the downside is that these are not sensitive to the shape of the function, therefore, causing substantial accuracy loss and SQNR degradation of the desired architecture.

Consider a continuous function *f(x)*, with first and second order derivatives, where $x \in X$ and $X = \left[x_{\mathrm{L}}, x_{\mathrm{H}}\right]$. The uniform segmentation methodology divides the function interval X, in equally sized segments; whereas, the non-uniform-by-powers-of-two segmentation methodology, decreases the size of subsequent segments within X according to the geometric progression with a common ratio of 1/2; the segmentation can be started either from x_{L} to x_{H} or vice-versa.

Fig. 1-1 and Fig. 1-2 show that the basic segmentation methodologies do not perform quite well when dealing with functions that present non-monotonic curvature features. For example, the uniform segmentation methodology is only suitable for functions that present a mostly constant or slightly changing curvature within the evaluation interval. Otherwise, if the function exhibits both fast-changing and slow-changing curvature features, an excessive amount of small segments are also created around the regions with slow-changing curvature. The reason of this is that the high density of segments that is needed to approximate the fast-changing curvature features appropriately is kept uniform along the whole evaluation interval. On the other hand, the non-uniform-by-powers-of-two segmentation methodology is only adequate for functions that present a curvature that either increases or decreases in the same direction. As a result, the direction in which the segments decrease in size is of utmost importance to appropriately approximate the function; in this sense, the density of segments should increase as the curvature of the function increases.

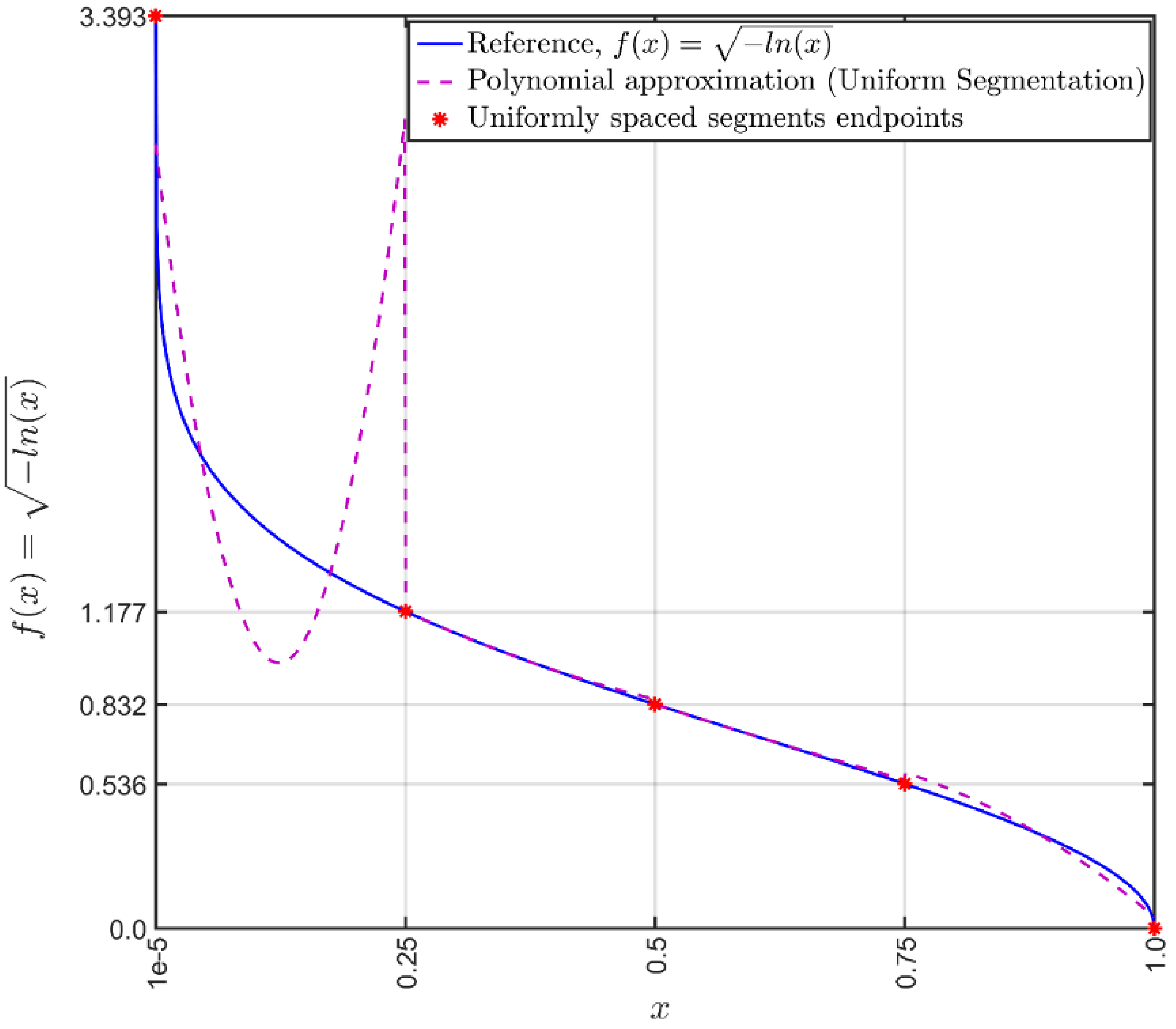


Fig. 1-1: Uniform segmentation, poor approximation accuracy to $f(x)=\sqrt{-\ln(x)}$

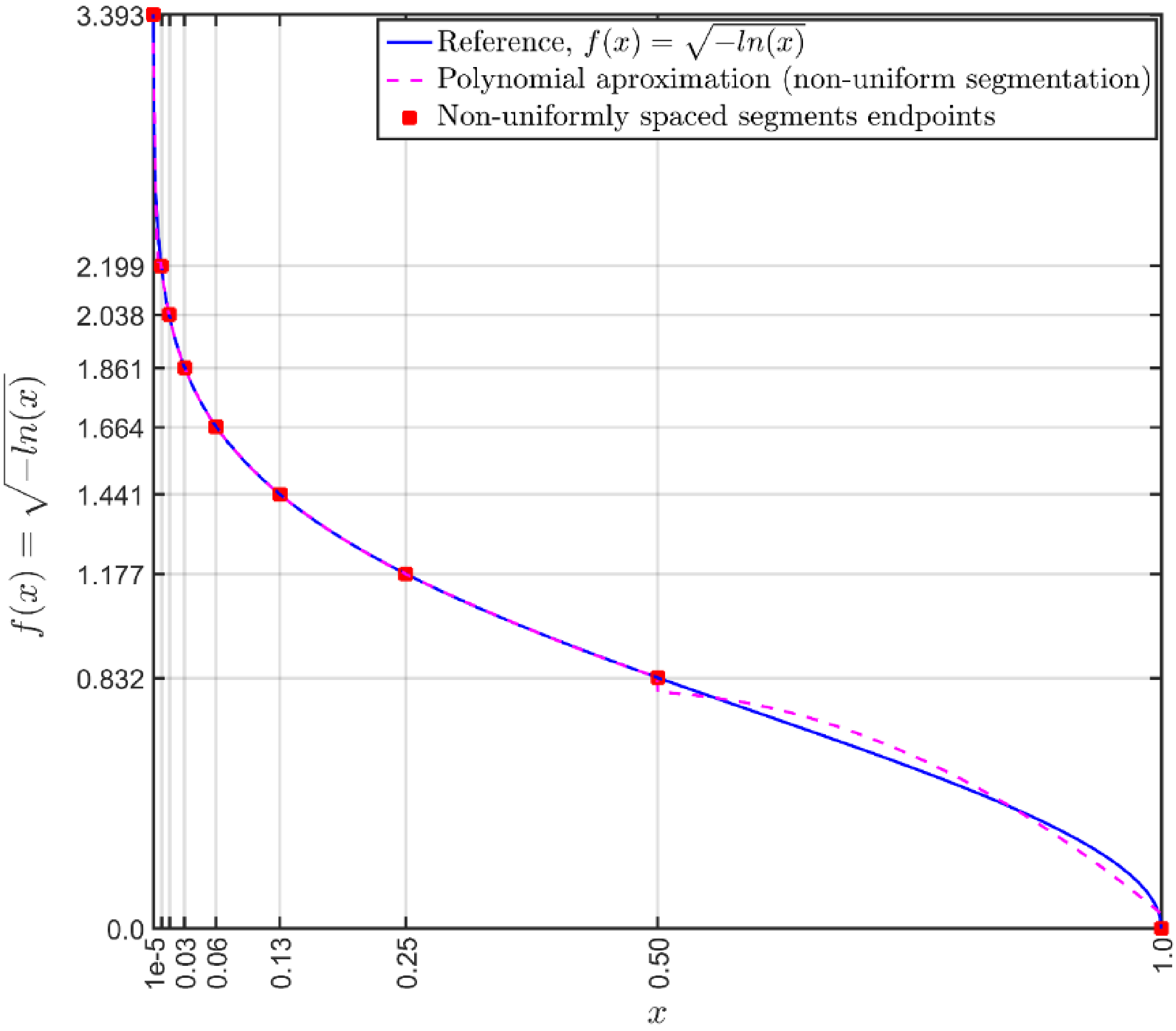


Fig. 1-2: Non-uniform segmentation, insufficient approximation accuracy to $f(x)=\sqrt{-\ln(x)}$

The HSM is a hybrid segmentation methodology that employs both uniform and non-uniform-by-powers-of-two segmentation methodologies to improve the approximation accuracy to functions with non-monotonic curvature behaviors; however, since the segmentation methodologies are employed in hierarchical levels, the control logic required for addressing the hierarchy of segments is it too complex and requires a significant amount of hardware resources in comparison to the proposed single level AFSM.

To employ the previously discussed state-of-art segmentation methodologies, the user should properly select the segmentation strategy (or a combination of them), and the minimum numbers of segments based on the shape of the function at hand. In many cases, this is an iterative trial and error process carried out by the user until the SQNR requirement (accuracy) is satisfied. Employing the inappropriate segmentation strategy results in a suboptimal trade-off between hardware resource consumption and SQNR degradation. In contrast, the proposed AFSM, through the analysis of the functions' first and second order derivatives, tackles these issues given that the algorithm automatically adapts the segmentation strategy and the density of segments to the shape of the function at hand.

1.3 Problem Statement

Several segmentation methodologies have been proposed for the evaluation of mathematical functions through PPA methods [14]; however, these segmentation methodologies are unsuitable for the segmentation of arbitrary functions because the segmentation strategy employed only delivers good approximation results if the function at hand presents specific curvature characteristics. Consequently, the employment of the inappropriate segmentation methodology causes the degradation of the SQNR, as well as, the usage of an excessive number of segments in an attempt to satisfy the output accuracy requirements.

On the other hand, the state-of-art hierarchical segmentation methodologies that define a segmentation hierarchy employing the more basic uniform and non-uniform methodologies, require a complex segment addressing which consumes a considerable amount of logic resources; furthermore, the segmentation solution, as well as the segment addressing logic is function-specific, and it cannot be reused.

This work proposes a segmentation methodology that allows segmenting out an arbitrary function based on the first order and second order derivatives of the function to be approximated within a continuous interval *X*. The introduced segmentation methodology allows optimizing the number of segments needed to satisfy the requirements of an objective function that best balances the maximum approximation error, memory, and logic hardware resources.

1.4 Research Contribution

This thesis presents a new segmentation methodology for the approximation through PPA methods of arbitrary transcendental functions through an automated function shape analysis based on the functions' first and second order derivatives. The proposed segmentation methodology addresses the segmentation process as a constrained optimization problem to minimize the number of segments according to design objectives such as SQNR and hardware area.

The simulation results show that the adaptive function segmentation methodology (AFSM) provides better segmentation performance and higher SQNR with lower hardware resources consumption in comparison to state of the art segmentation methodologies; therefore, the AFSM represents an excellent alternative for implementing high accuracy PPA based transcendental function evaluators embedded in sophisticated digital signal processing algorithms.

1.5 Thesis Objectives

The objectives of this thesis are the following:

- To develop an adaptive function segmentation methodology, for the evaluation of arbitrary mathematical functions via PPA.
- To develop a shape analysis methodology for the efficient segmentation of arbitrary functions based on the functions' chordal length and the functions' first order and second order derivatives.
- The implementation of an optimization algorithm and the introduction of a cost function for the optimization of hardware resources through the minimization of the number of segments according to SQNR requirements of the application.

1.6 Derived Publications

As part of this thesis work, two papers were developed:

1) A conference paper for the IEEE Latin America Microwave Conference 2016 titled: “A novel function segmentation methodology for implementing affordable channel emulators”. The published paper can be found in Appendix A.

2) A journal paper for the IEEE LAMC-2016 Mini-Special Issue in IEEE Transactions on Microwave Theory and Techniques titled: “An adaptive function segmentation methodology based on first and second order derivatives for hardware optimization of function evaluators”. The submitted paper can be found in Appendix B.

2 Adaptive Function Segmentation for Hardware Resources Optimization

The development of an algorithm that automatically adapts the segmentation strategy requires precise knowledge about the shape of the function under analysis and its curvature speed of change within the evaluation interval. A convenient way to get such an insight is through the implementation of an exploratory algorithm that analyzes the first and second order derivatives of the function and identifies the points within the evaluation interval X where to split the function into segments to maximize the accuracy of approximation through low-degree piecewise polynomials. In this sense, the density of segments along X is automatically balanced according to the progression of the functions' curvature; consequently, the algorithm automatically allocates a greater amount of segments around the regions that present a more pronounced curvature.

The calculations carried out by the algorithm are solved numerically; therefore, the following sections utilize a discrete nomenclature for referring to the equations, functions, and procedures used to describe the proposed segmentation algorithm.

2.1 Function Shape Analysis Through First and Second Order Derivatives

The shape of the function *f(x)* and its curvature speed of change are analyzed through the first and second order derivatives in a simple but yet powerful manner. To simplify the segmentation process and to achieve improved approximation accuracy, the first step is to perform a coarse segmentation by splitting the evaluation interval *X* at the critical points where the function presents a local minimum, a local maximum or an inflection point. In this work, the segments defined by this coarse segmentation stage are called main segments. The objective of the coarse segmentation process is to define segments with a curvature that evolves monotonically (in the same direction), either growing or decreasing, in order better approximate it through a 2-degree polynomial.

The computation of the first and second order derivative is performed numerically through (2-1) and (2-2). For the numerical computation of the functions' derivatives, the interval X is quantized into N points addressed as x_i, where $1 \le i \le N$.

$$g(x_i) = \left.\frac{df(x)}{dx}\right|_{x=x_i} \approx \frac{f(x_{i+1}) - f(x_{i-1})}{2\Delta x} \tag{2-1}$$

$$h(x_i) = \left.\frac{d^2 f(x)}{dx^2}\right|_{x=x_i} \approx \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{\Delta x^2} \tag{2-2}$$

where, $\Delta x = \left|x_{i+1} - x_i\right| \ \forall \ i$.

The local minimum, maximum or inflection points of $f(x)$ within the evaluation interval X are found at a given point x_i where there is a change of sign in $g(x)$ or $h(x)$ relative to the next point x_{i+1}, i.e., $\operatorname{sign}\left(g\left(x_{i+1}\right)\right) \neq \operatorname{sign}\left(g\left(x_i\right)\right)$ OR $\operatorname{sign}\left(h\left(x_{i+1}\right)\right) \neq \operatorname{sign}\left(h\left(x_i\right)\right)$. Therefore, the set of main segments endpoints S encompasses the boundary points of the evaluation interval $\left[x_L, x_H\right]$ and any other intermediate critical points x_i, identified through the coarse segmentation process. However, if no critical points are identified, then the entire evaluation interval delimited by the segment endpoints at x_L and x_H is passed to the second segmentation step for further segmentation tuning to achieve the SQNR requirement.

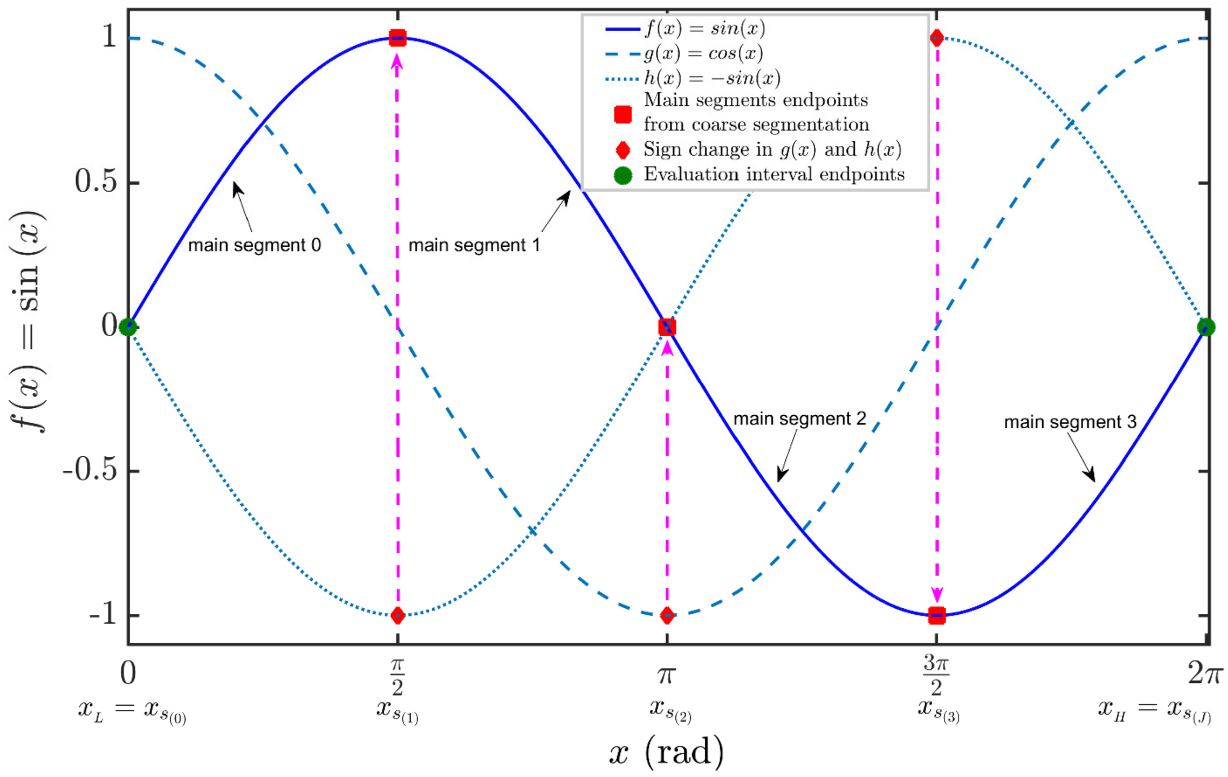


Fig. 2-1: Coarse segmentation (square marks) of $f(x)=\sin(x)$ at the critical points $x_{s_{(0)}}, x_{s_{(1)}}, x_{s_{(2)}}, \ldots, x_{s_{(J)}}$.

To exemplify the previous point, let us think on $f(x)=\sin(x)$ in Fig. 2-1, which is to be segmented out within an interval that stretches along a full cycle, $X=[0,\ 2\pi]$. The limiting points x_L and x_H of the evaluation interval are called the evaluation interval endpoints (circle marks), which are automatically created by the segmentation algorithm and identified as $x_L = x_{s_0}$, and $x_H = x_{s_{(J)}}$ where x_{s_0} represents the initial endpoint of the first main segment and $x_{s_{(J)}}$ represents the last endpoint of the J^{th} main segment identified. The square marks in Fig. 2-1, at $x_{s_{(1)}} = \pi/2$, $x_{s_{(2)}} = \pi$ and $x_{s_{(3)}} = 3\pi/2$ correspond to a local maximum, an inflection point, and a local minimum of the function *f(x)* within the interval *X*. These locations are identified during the coarse segmentation stage by the sign changes in either *g(x)* or *h(x)* and represent endpoints of the main segments in *f(x)* where its curvature changes direction (marked by the vertical purple arrows). After the coarse segmentation processes, the set of segments is defined as $S = [x_{s_{(0)}}, x_{s_{(1)}}, \ldots, x_{s_{(J)}}]$.

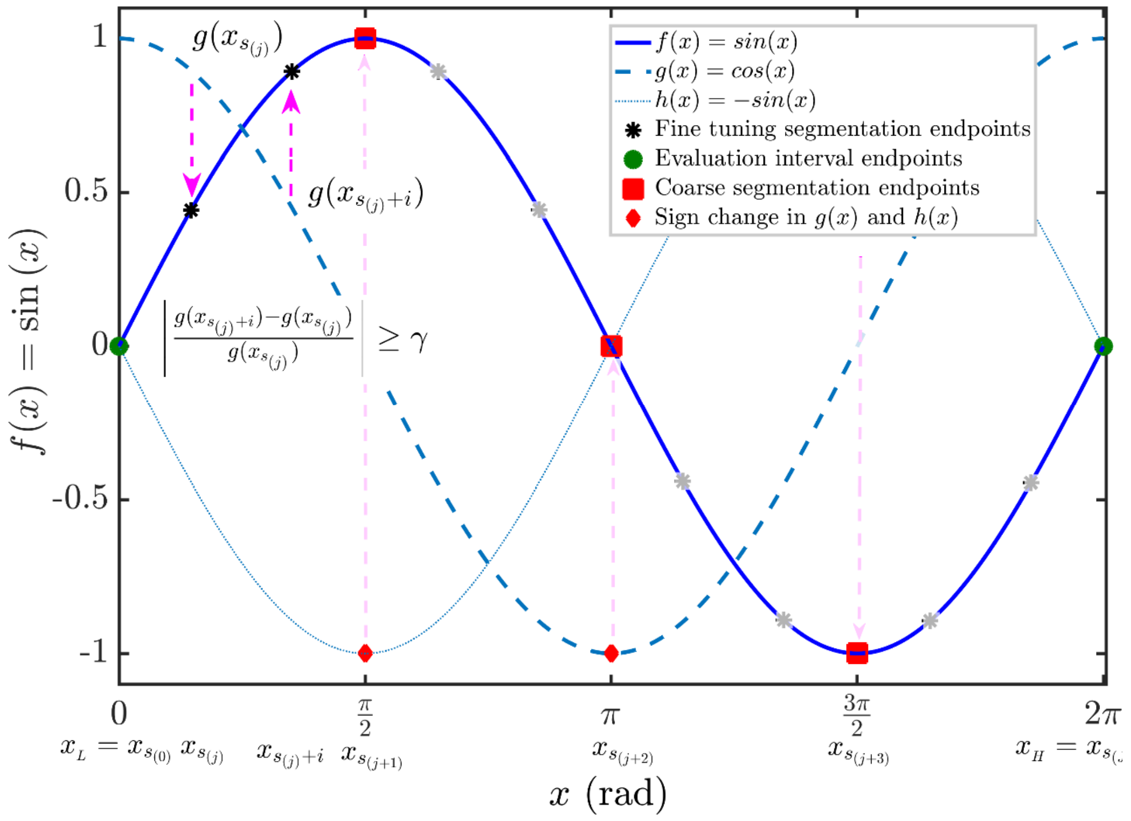


Fig. 2-2: Fine segmentation (asterisk marks) within the main segments.

The second step, as depicted in Fig. 2-2, has the purpose of further splitting the previously defined main segments to achieve the SQNR requirement. This fine-tuning segmentation process defines internal segments endpoints inside the main segments, which are bounded by the consecutive main segment endpoints $\left[x_{s_{(j)}}, x_{s_{(j+1)}}\right]$ identified through the previous coarse segmentation step. A new internal endpoint is defined at any x_i where the relative change of value on the first order derivative between the previously defined endpoint at $x_{s_{(j)}}$ and the nearest subsequent point $x_i \; \forall \; x_{s_{(j)}} < x_i < x_{s_{(j+1)}}$ exceeds a given γ threshold. The next mathematical expression synthesizes the previous description.

$$\left| \frac{g(x_{s_{(j)}+i}) - g(x_{s_{(j)}})}{g(x_{s_{(j)}})} \right| \geq \gamma \tag{2-3}$$

If the condition expressed in (2-3) is satisfied, then the current x_i is defined as a new segment endpoint in the set *S*; therefore, the newly defined segment endpoint is now identified as $x_{s_{(j)}}$ where $x_{s_{(j)}} = x_i$. From this stage, the search for the next internal endpoint continues repeating the previously mentioned steps until reaching the end of the current main segment that is identified as $x_{s_{(j+1)}}$.

2.1.1 Bidirectional Function Shape Analysis

As depicted in Fig. 2-3, to improve the accuracy of approximation to *f(x)*, the bidirectional fine tuning segmentation of each main segment according to the γ threshold is performed, from $x_{s_{(j)}}$ to $x_{s_{(j+1)}}$ (forward segmentation), and from $x_{s_{(j+1)}}$ to $x_{s_{(j)}}$ (backward segmentation). The bidirectional exploration of the fine-tuning segmentation is carried out given that the location of the segments endpoints, x_i, where the γ threshold is met differs depending on the starting point of the segmentation process; therefore, the approximating polynomials and consequently the accuracy of approximation obtained from each direction of segmentation are different. After performing both forward and backward fine-tuning segmentation exploration processes for each segment, the polynomials that deliver the best approximation accuracy are selected. The implementation of the bidirectional fine-tuning segmentation allows independently maximizing the approximation accuracy for each main segment given that the direction of segmentation that delivers the best approximation results is independent between main segments.

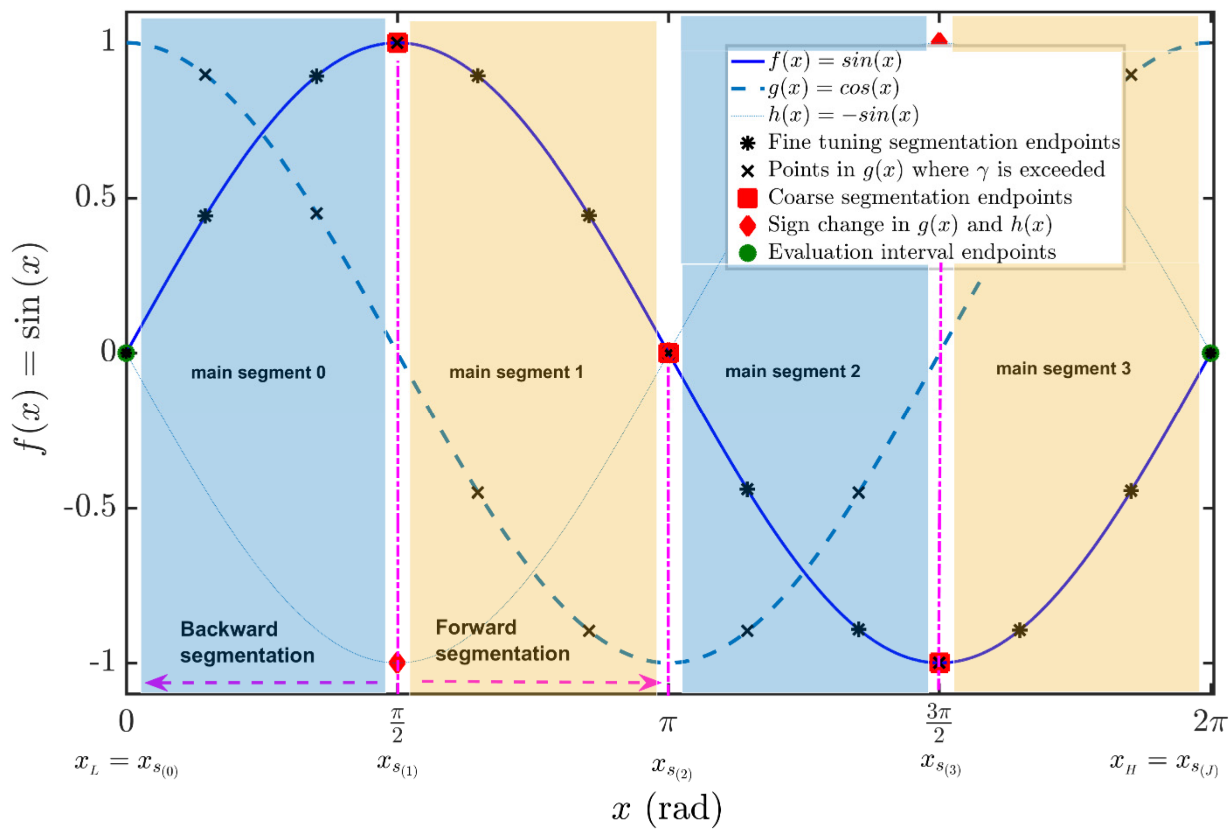


Fig. 2-3: Independent bi-directional fine-tuning segmentation.

2.2 Chordal Segment Length Tuning

In addition to the shape analysis based on the γ threshold, the proposed algorithm also implements a minimum chordal length control that serves as a design knob for the optimization process through the minimum chordal length threshold κ. The κ threshold is expressed as a percentage of the functions' total chordal length within the evaluation interval; therefore, $0\% < \kappa \leq 100\%$.

The κ threshold serves two purposes; the first one is to achieve a better balance in the density of segments allocated when dealing with functions that present both regions of pronounced curvature as well as regions of subtle curvature. In this sense, the κ threshold makes it possible to avoid having an excessive amount of tightly spaced segments around areas with pronounced curvature when the value of the γ threshold is too small. Inconveniently small values for the γ

threshold can result due to a poor selection from the user of the initial γ value or because the optimization process itself has taken γ towards the design space of small values.

The second purpose of the κ threshold is to prevent having too small segments that would cause the PPA algorithm to become unstable and fail in finding a suitable set of coefficients. This failure manifests itself when the integer part of the generated coefficients is too big that its fixed-point representation requires most available bits from the word length. A consequence of this is a severe loss of accuracy given that only a few bits remain for the fractional part of the coefficients. Consequently, the definition of a new segment endpoint at a given x_i requires that both γ and κ thresholds be satisfied.

Fig. 2-4 exemplifies how the chordal length of a function within the interval limited by x_a and x_b is approximated by summing up the length of the hypotenuse of the many small triangles that fit within such interval.

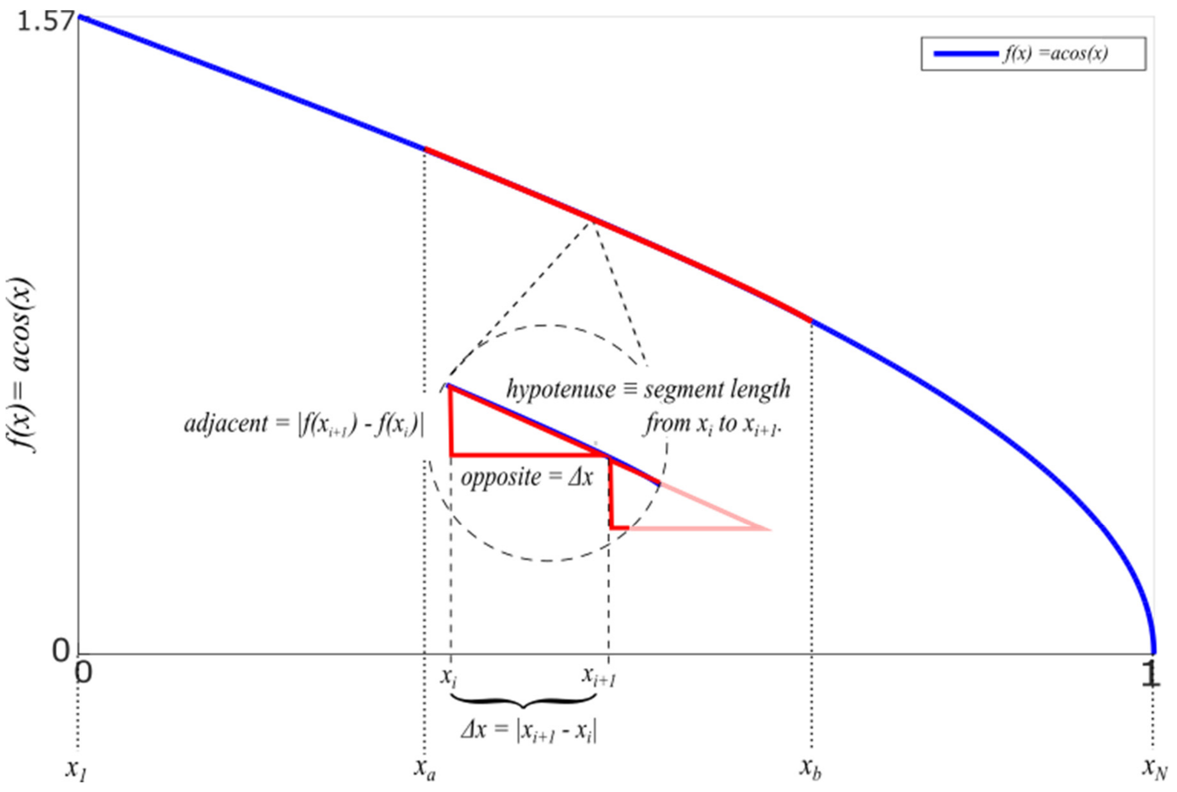


Fig. 2-4: Chordal segment length approximation.

The length of the triangles' hypotenuse is computed through the Pythagoras theorem expressed in (2-4), where the length of the triangles opposite and adjacent sides is defined as $\Delta x = |x_{i+1} - x_i|$ and $|f(x_{i+1}) - f(x_i)|$ respectively.

$$\text{length}\left(\left[x_a, x_b\right]\right) = \sum_{i=a}^{b} \sqrt{(\Delta x)^2 + \left(f(x_{i+1}) - f(x_i)\right)^2} \tag{2-4}$$

2.3 Polynomials Coefficient Generation

After each iteration of the AFSM splitting the function interval X into the set with J segments (such that $X = \bigcup_{j=0}^{J} [x_{s_{(j)}}, x_{s_{(j+1)}}]$, where $x_{s_{(j)}} < \ldots < x_{s_{(j+1)}} < \ldots < x_{s_{(J)}}$, are the endpoints computed according to the γ and κ thresholds), the m^{th} order polynomials coefficients that best fit each segment are computed. The polynomials employed to approximate the function segments can be of any order, $m \geq 1$ for $m \in \mathbb{Z}$. However, the usage of low-even-order polynomials is advised for the proposed segmentation methodology given that the coarse segmentation step already ensures that the curvature of the function evolves monotonically within each segment. Therefore, low-even-order polynomials fit well the curvature of the segments and require less memory than odd-order polynomials to store the coefficients as well as fewer logic resources to carry on the coefficients multiplications.

For the proposed AFSM, two PPA methods were tested for the computation of the best fit polynomials coefficients. The polynomial least square approximation method (LSPA) [15, p. 28] and the miniMax polynomial approximation method (mMPA) [15, p. 32], which is based on the Remez algorithm [16]. Each of the employed PPA methods treats the approximation error differently and consequently provide different levels of SQNR and accuracy between the original function *f(xi)* and the polynomial-based approximation function $\hat{f}_j(x_i)\,|_{\vec{p}_j}$ in (2-5).

$$\hat{f}_j(x_i)|_{\vec{p}_j} = p_{j_{(0)}} + p_{j_{(1)}} x_i + \ldots + p_{j_{(m)}} x_i^m \tag{2-5}$$

where,

- j represents the segment index for the set of segments S.
- $\vec{p}_j = \left[p_{j_{(0)}}, p_{j_{(1)}}, \ldots, p_{j_{(m)}} \right]$ are the polynomial coefficients of the *m*th order polynomial used to approximate to the *j*th segment of the function *f(x)*.

2.3.1 Least Square Polynomial Approximation and Error Treatment

From the set of data points $(x_i, f(x_i))$ within the segment delimited by $\left[x_{s_{(j)}},\ x_{s_{(j+1)}} \right]$, the objective of the LSPA is to determine the $m+1$ coefficients of an *m*-degree polynomial, as expressed in (2-5), that minimize the error of approximation in the least square sense between the original function *f(x)* and the approximating polynomial. Therefore, the sum of squared residuals of the *j*th segment is minimum when the condition expressed in (2-6) is satisfied [5] for all the polynomial coefficients.

$$\frac{\partial(R_j)}{\partial \vec{p}_j} = 0, \ for\ j = 0, \ldots, J \tag{2-6}$$

where,

$$R_j \equiv \sum_{i=s_j}^{s_{(j+1)}} \left[f(x_i) - \hat{f}_j(x_i)|_{\vec{p}_j} \right]^2 \tag{2-7}$$

The polynomial coefficients values are obtained by solving the partial derivatives in (2-6) for all p_j. This procedure yields the following set of normal equations.

$$\begin{aligned}
&p_0 \sum_{i=s_j}^{s_{(j+1)}} 1 + p_1 \sum_{i=s_j}^{s_{(j+1)}} x_i + \ldots + p_m \sum_{i=s_j}^{s_{(j+1)}} x_i^m = \sum_{i=s_j}^{s_{(j+1)}} f(x_i) \\
&p_0 \sum_{i=s_j}^{s_{(j+1)}} x_i + p_1 \sum_{i=s_j}^{s_{(j+1)}} x_i^2 + \ldots + p_m \sum_{i=s_j}^{s_{(j+1)}} x_i^{m+1} = \sum_{i=s_j}^{s_{(j+1)}} x_i f(x_i) \\
&\vdots \\
&p_0 \sum_{i=s_j}^{s_{(j+1)}} x_i^m + p_1 \sum_{i=s_j}^{s_{(j+1)}} x_i^{m+1} + \ldots + p_m \sum_{i=s_j}^{s_{(j+1)}} x_i^{2m} = \sum_{i=s_j}^{s_{(j+1)}} x_i^m f(x_i)
\end{aligned} \tag{2-8}$$

For (2-8) the right-hand side of the set of normal equations can be represented as $\vec{b} = \left[\sum_{i=s_j}^{s_{(j+1)}} f(x_i) \quad \sum_{i=s_j}^{s_{(j+1)}} x_i f(x_i) \quad \ldots \quad \sum_{i=s_j}^{s_{(j+1)}} x_i^m f(x_i) \right]$. Therefore, by defining a matrix A as follows,

$$A = \begin{bmatrix} 1 & x_{s_j} & x_{s_j}^2 & \cdots & x_{s_j}^m \\ 1 & x_{s_{(j)}+1} & x_{s_{(j)}+1}^2 & \cdots & x_{s_{(j)}+1}^m \\ 1 & x_{s_{(j)}+2} & x_{s_{(j)}+2}^2 & \cdots & x_{s_{(j)}+2}^m \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{s_{(j+1)}} & x_{s_{(j+1)}}^2 & \cdots & x_{s_{(j+1)}}^m \end{bmatrix} \tag{2-9}$$

the set of normal equations in (2-8) can be condensed as the following linear system (the A matrix is known as the Vendermonde matrix), which can be solved using the well-known Gauss-Jordan method [17] to obtain the values of the polynomial's coefficients that minimize the error of approximation.

$$\left(A^T A\right)\vec{p} = \vec{b} \tag{2-10}$$

An important remark is that from (2-6) and (2-7) one can observe that the error treatment strategy of the LSPA algorithm provides direct benefit to the improvement of the SQNR because it explicitly minimizes the sum of squared residuals expression in (2-11). Such error expression, in fact, represents the quantization noise energy; a factor that lies as the denominator of the SQNR expression that is presented in Section 2.4.

$$E[R] = \sum_{j=0}^{J} R_j \tag{2-11}$$

2.3.2 miniMax Polynomial Approximation and Error Treatment

The objective of the miniMax polynomial approximation algorithm (mMPA) is to minimize the maximum absolute error or discrepancy between the approximation polynomial $\hat{f}_j(x_i)|_{\vec{p}_j}$ and the original function $f(x_i)$ in the uniform norm sense L_∞. The mMPA algorithm employs Chebyshev polynomials [18] of order m within the interval $\left[x_{s_j}, x_{s_{j+1}}\right]$ that delimits the j^{th} segment.

The computation of the polynomials coefficient that minimizes the maximum error of approximation is performed by solving the following optimization problem.

$$\overrightarrow{p}_j = \arg\min_{\vec{p}_j} \left\{ \left\| \vec{e}_j(\vec{p}_j) \right\|_\infty \right\}, \text{ for } j = 0, \ldots, J \tag{2-12}$$

where,

$$\vec{e}_j(\vec{p}_j) = [e_{s_j}, \ldots, e_i, \ldots, e_{s_{j+1}}] \tag{2-13}$$

$$e_i = \left| f(x_i) - \hat{f}_j(x_i)|_{\vec{p}_j} \right|, \text{ for } s_j \le i < s_{j+1} \tag{2-14}$$

In this sense, $\hat{f}_j(x_i)|_{\vec{p}_j}$ is a miniMax polynomial with coefficients $\overrightarrow{p}_j$ if it satisfies the condition that there are at least $m + 2$ points within the segment evaluation interval $\left[x_{s_j}, x_{s_{j+1}}\right]$ where:

$$\hat{f}_j(x_i)\,|_{\vec{p}_j} - f(x_i) \;=\; (-1)^i\left[\hat{f}_j(x_0)\,|_{\vec{p}_j} - f(x_0)\right] \;=\; \pm\left\| f(x) - \hat{f}_j(x)\,|_{\vec{p}_j} \right\|_\infty \qquad \text{(2-15)}$$

The expression in (2-15) means that for $\hat{f}_j(x_i)\,|_{\vec{p}_j}$ to be a miniMax polynomial, it should satisfy the condition that the maximum error is reached $m+2$ times (the total number of minimum and maximum extrema points) and that the sign of such error alternates at each error extrema. Henceforth, the Remez exchange algorithm, which is summarized in the flow diagram of Fig. 2-5, determines the coefficients of miniMax polynomials by exploiting this important property; for more detail on the implementation of the Remez´s algorithm refer to [15].

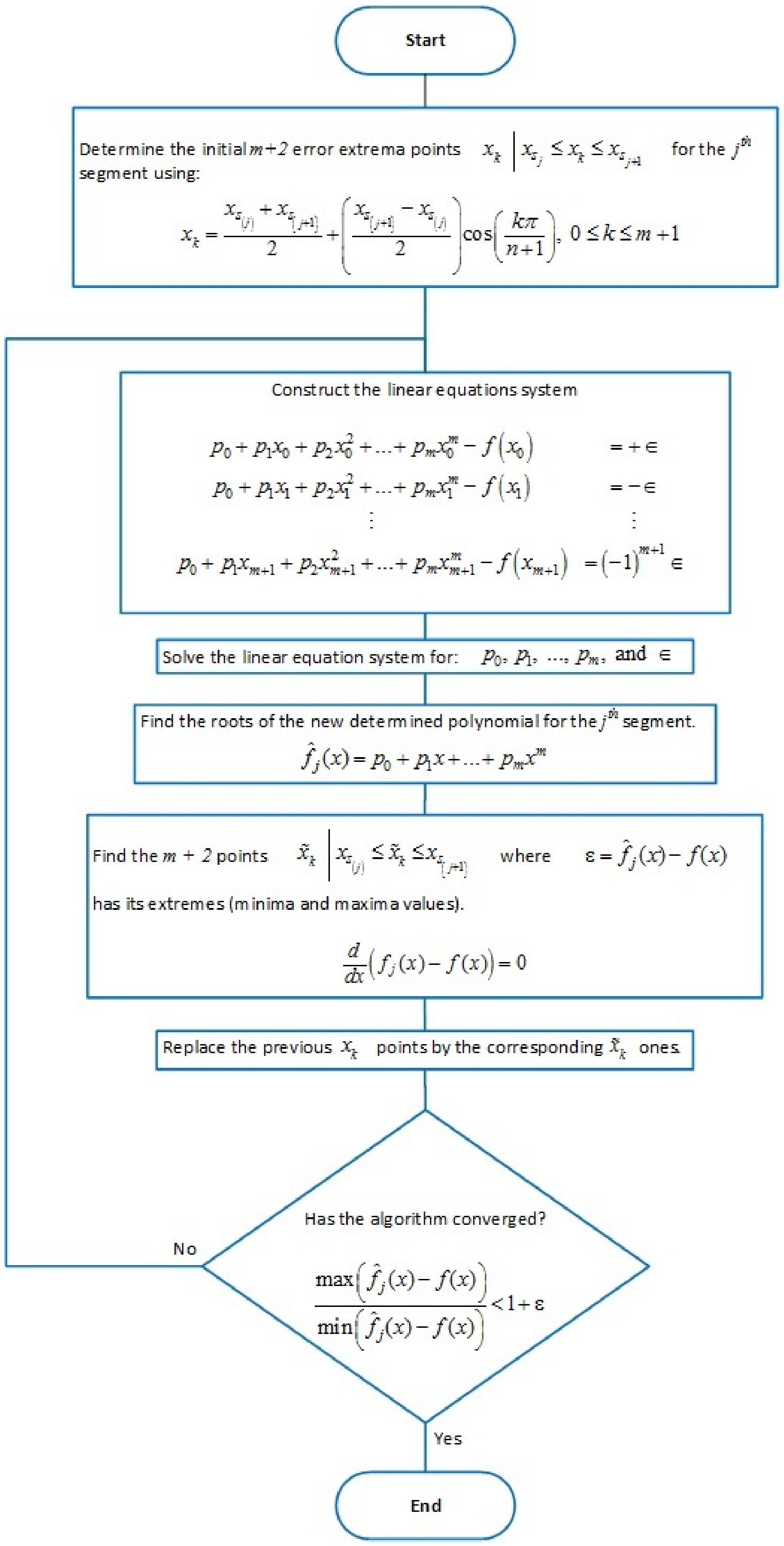

Start
Determine the initial $m+2$ error extrema points $x_k \,\big|\, x_{s_j} \le x_k \le x_{s_{j+1}}$ for the j^{th} segment using:
$x_k = \frac{x_{s_{(j)}} + x_{s_{[j+1]}}}{2} + \left(\frac{x_{s_{[j+1]}} - x_{s_{(j)}}}{2}\right)\cos\left(\frac{k\pi}{n+1}\right),\ 0 \le k \le m+1$
Construct the linear equations system
$p_0 + p_1x_0 + p_2x_0^2 + \ldots + p_mx_0^m - f(x_0) = +\in$
$p_0 + p_1x_1 + p_2x_1^2 + \ldots + p_mx_1^m - f(x_1) = -\in$
$p_0 + p_1x_{m+1} + p_2x_{m+1}^2 + \ldots + p_mx_{m+1}^m - f(x_{m+1}) = (-1)^{m+1}\in$
Solve the linear equation system for: $p_0, p_1, \ldots, p_m$, and $\in$
Find the roots of the new determined polynomial for the j^{th} segment.
$\hat{f}_j(x) = p_0 + p_1x + \ldots + p_mx^m$
Find the $m+2$ points $\tilde{x}_k \,\big|\, x_{s_{(j)}} \le \tilde{x}_k \le x_{s_{[j+1]}}$ where $\varepsilon = \hat{f}_j(x) - f(x)$
has its extremes (minima and maxima values).
$\frac{d}{dx}\left(f_j(x) - f(x)\right) = 0$
Replace the previous x_k points by the corresponding $\tilde{x}_k$ ones.
Has the algorithm converged?
$\frac{\max\left(\hat{f}_j(x) - f(x)\right)}{\min\left(\hat{f}_j(x) - f(x)\right)} < 1 + \varepsilon$
No
Yes
End

Fig. 2-5: Flow diagram of the Remez´s exchange algorithm.

In comparison to the least square method, for regular functions, the miniMax method yields a smaller error of approximation [15]; however, the miniMax method does not guarantee a lower SQNR than that achieved through least square.

2.4 Fixed-Point and SQNR Analysis

For this work, the SQNR is the metric employed for measuring the accuracy of the approximation to the reference function *f(x)* through a set of fixed-point low-degree polynomials. The SQNR, defined in (2-16), is an intuitive and widely used metric of accuracy, which is based on the ratio between the power of the signal of interest and the power of the quantization noise, as it was mentioned in Section 2.3.1. In other words, the SQNR expresses how well an analog signal is approximated through a digital fixed-point representation given the finite number of bits of the system's word length.

$$SQNR_{dB} = 10\log_{10}\left(\frac{\sum_{i=1}^{N} f(x_i)^2}{\sum_{j=1}^{J}\sum_{i=s_j}^{s_{j+1}} [f(x_i) - Q(\hat{f}_j(x_i)|_{\vec{p}_j})]^2} \right) \tag{2-16}$$

The term $Q(\cdot)$ in the denominator of (2-16) is the operator that quantizes the argument using a word length of WL bits, from which, QI bits are assigned to the integer part and QF bits are assigned to the fractional part [19]; the previous is expressed as follows:

$$WL = QI + QF \tag{2-17}$$

The first step to determine the most appropriated fixed-point representation as to avoid overflow or truncation is to compute the minimum number of bits assigned to QI. In this sense, the expression in (2-18) provides the minimum QI bits required to represent signed values in two's complement with a range that is symmetric around zero. The expression in (2-18) takes into account the magnitude of the entire set of polynomials coefficients for all segments, the magnitude of the values in the evaluation interval *X*, and the magnitude of the range of the function being approximated.

$$QI = \left\lceil \log_2 \left(\max \left(\alpha \right) + 1 \right) \right\rceil + 1 \qquad (2\text{-}18)$$

where,

$$\alpha = \left\{ \left| \vec{p}_j \right|, \left| x_i \right|, \left| f\left(x_i \right) \right| \right\} \ , \ \forall \ i, j$$

A fixed-point variable α for which the minimum number of QI and QF bits are determined through (2-17) and (2-18), can take values in the range $\left(-2^{QI-1} \right) \leq \alpha \leq \left(-2^{QI-1} - 1 \right)$, [19].

The proposed segmentation methodology relies on an iterative optimization algorithm to determine the best segmentation approach. Therefore, once the fixed-point analysis has been carried out for each segmentation iteration, the achieved SQNR is computed and fed back to the optimization algorithms' objective function to determine whether the SQNR requirement has been satisfied or further segmentation refinement is required.

2.5 Segmentation Optimization

The proposed AFSM implements an optimization algorithm that searches in the design space $\mathbb{R}^2$ of the γ and κ threshold parameters, looking for a suitable set of values that satisfy the SQNR requirement while minimizing the required number of segments. The implemented search algorithm solves the constrained non-linear optimization problem defined in (2-19), for a target SQNR requirement, which is provided by the user as a range with an upper d^{ub} and a lower limit d^{lb} according to application-specific needs.

$$\begin{gathered}\vec{d}^* = \arg\min_{\vec{d}\,\in\,\mathbb{R}^2} U\left(S\left(\vec{d}\right)\right)\\ \textit{subject to}\\ \vec{d}^{lb} \le \vec{d} \le \vec{d}^{ub}\end{gathered} \tag{2-19}$$

where,

- $\vec{d} \in \mathbb{R}^2, \vec{d} = [\gamma, \kappa]$; is the vector of design variables subject to optimization.
- $\vec{d}^* \in \mathbb{R}^2$, $\vec{d}^* = [\gamma^*, \kappa^*]$; is the vector of design variables after the optimization process has been completed.
- $\vec{d}^{lb}, \vec{d}^{ub} \in \mathbb{R}^2$; are the upper and lower design-feasibility restrictions for the design variables.
- $S(\vec{d}) \in \mathbb{R}^2 \to \mathbb{R}$; is the function that performs the segmentation process according to the input design variables in $\vec{d}$. The function returns the SQNR scalar value.
- $U : \mathbb{R} \to \mathbb{R}$; is the cost function that computes the error between the current design SQNR and the target SQNR requirement.

The solution of the constrained non-linear optimization problem is simplified if the boxed constraints ($\vec{d}^{lb} \le \vec{d} \le \vec{d}^{ub}$) are incorporated into an unconstrained optimization problem; refer to (2-20). For this, the design variables in $\vec{d}$ are transformed into $\vec{z}$ through (2-21). After applying the suggested transformation, the restrictions of the optimization problem are now embedded in the design variables because their range, due to the $\arcsin(\cdot)$ function (See Fig. 2-6), is now bounded within the interval $[0,\ 1.5708]$; for further reference, see [20].

$$\vec{z}^* = \arg\min_{\vec{z}} U\left(S(\vec{z})\right) \tag{2-20}$$

$$\vec{z}_i = arcsin\left(\sqrt{\frac{\vec{d_i} - \vec{d_i}^{lb}}{\vec{d_i}^{ub} - \vec{d_i}^{lb}}}\right) \tag{2-21}$$

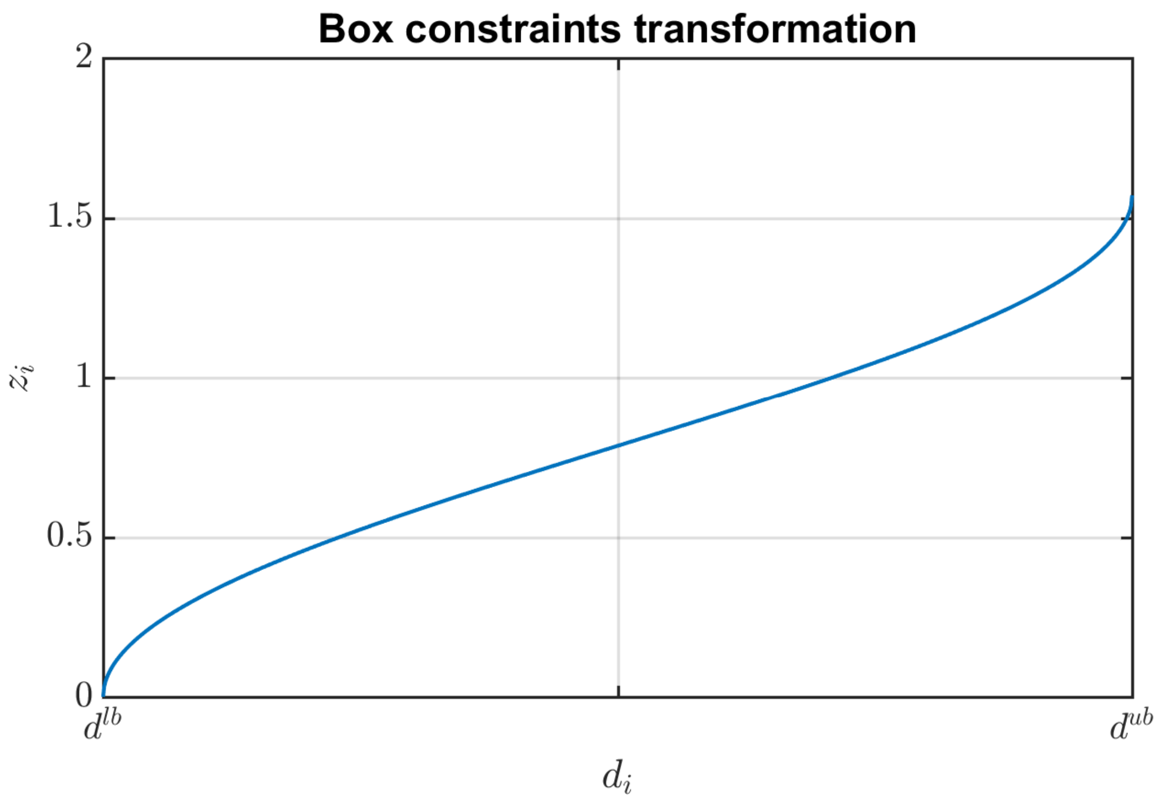


Fig. 2-6: Graphical representation of the boxed constraints transformation.

For this particular work, the solution to the unconstrained non-linear optimization problem for $\vec{z}$ is done through the Nelder-Mead algorithm [21]; however, many other local or global search methods can be employed as well.

3 Segmentation Methodology Implementation

The Algorithm 1 condenses the verbal methodology description provided in previous sections to facilitate the reproducibility of the proposed segmentation methodology. Given that the AFSM was implemented in MATLAB, the pseudocode employs sub-index notation to address the discrete elements of vectors and collections of objects. Further detail of the pseudocode variables and their usage is summarized in TABLE 1.

TABLE 1: DESCRIPTION OF VARIABLES EMPLOYED IN THE PSEUDO-CODE OF THE ADAPTIVE FUNCTION SEGMENTATION METHODOLOGY.

Variable name	Description
x	The vector of the evaluation interval X that is quantized from x_L to x_H .
y	The vector with the evaluation results of $f(x)$ within the interval X .
x_{FxdPt}	The x vector in fixed-point representation.
y_{FxdPt}	The y vector in fixed-point representation.
h , g	The vectors that store the first and second derivatives.
Dx	The discretization resolution for x , the default is $\Delta x = \frac{\lvert x_H - x_L \rvert}{2^{10}}$.
Dg	A temporary variable used to store the first derivative delta between the previous segment and a subsequent point x_i .
x_L	The lower limit of the evaluation interval X .
x_H	The upper limit of the evaluation interval X .
quantElmts	The number of quantization elements within the evaluation interval X .
mainSegmts	The collection to store the segment objects from the coarse segmentation process.
$allSegmts_{LSPA}$	The collection to store all the segments objects that delivered the largest SQNR through the LSPA.

$allSegmts_{MMPA}$	The collection to store all the segments objects that delivered the largest SQNR through the mMPA.
$fwdSegmts$	A temporary collection that stores the segments from the forward segmentation exploration.
$bwdSegmts$	A temporary collection that stores the segments from the backward segmentation exploration.
$fwdCoeffs_{LSPA}$	A collection of LSPA coefficients for the segments from the forward segmentation exploration.
$bwdCoeffs_{LSPA}$	A collection of LSPA coefficients for the segments from the backward segmentation exploration.
$fwdCoeffs_{MMPA}$	A collection of mMPA coefficients for the segments from the forward segmentation exploration.
$bwdCoeffs_{MMPA}$	A collection of mMPA coefficients for the segments from the backward segmentation exploration.
$fwdCoeffs_{LSPA\_FxdPt}$	A collection of LSPA coefficients in fixed-point representation for the segments from the forward segmentation exploration.
$bwdCoeffs_{LSPA\_FxdPt}$	A collection of LSPA coefficients for the segments from the backward segmentation exploration.
$fwdCoeffs_{MMPA\_FxdPt}$	A collection of mMPA coefficients in fixed-point representation for the segments from the forward segmentation exploration.
$bwdCoeffs_{MMPA\_FxdPt}$	A collection of mMPA coefficients in fixed-point representation for the segments from the backward segmentation exploration.
$allCoeffs_{LSPA}$	The collection of polynomial coefficients for the current segmentation realization through Least Square PPA method.
$allCoeffs_{LSPA\_FxdPt}$	A collection of polynomial coefficients in fixed-point representation for the current segmentation realization through Least Square PPA method.
$allCoeffs_{MMPA}$	The collection of polynomial coefficients for the current segmentation realization through miniMax PPA method.

$allCoeffs_{MMPA\_FxdPt}$	The collection of polynomial coefficients in fixed-point representation for the current segmentation realization through the miniMax PPA method.
$coeffsLUT_{LSPA}$	Stores the set of LSPA coefficients for the segmentation that satisfies the SQNR requirement.
$coeffsLUT_{MMPA}$	Stores the set of MMAP coefficients for the segmentation that satisfies the SQNR requirement.
$fwdSQNR_{LSPA}$	SQNR result from the forward segmentation exploration of the j^{th} main segment through the LSPA method.
$bwdSQNR_{LSPA}$	SQNR result from the backward segmentation exploration of the j^{th} main segment through the LSPA method.
$fwdSQNR_{MMPA}$	SQNR result from the forward segmentation exploration of the j^{th} main segment through the mMPA method.
$bwdSQNR_{MMPA}$	SQNR result from the backward segmentation exploration of the j^{th} main segment through the mMPA method.
$SQNR_{LSPA}$	The resulting SQNR responses from the last segmentation over the whole interval X with coefficients obtained through the LSPA method.
$SQNR_{MMPA}$	The resulting SQNR responses from the last segmentation over the whole interval X with coefficients obtained through the mMPA method.
γ_{Th}	The design parameter for optimization, first derivative threshold.
κ_{Th}	The design parameter for optimization, minimum chordal segment length threshold.
$SQNR^{lb}$	Lower bound of the target SQNR requirement.
$SQNR^{ub}$	Upper bound of the target SQNR requirement.
W_{Len}	The system word length.
m	The polynomial degree, the default is 2.
$accumLen$	A temporary variable that holds the accumulated chordal length.
i, j, k	The for-loop iteration count variables.
$contSearch$	The control flag for the optimization process stop condition.

3. Segmentation Methodology Implementation

Parameter Definitions and **Parameters Initialization** sections of the pseudocode, introduce and initialize the variables and constants that are used across the code to set up the algorithm functionality and to store the computation results. The main body of the segmentation algorithm is showed within the $do-while$ loop (lines 17 through 63) that resembles the optimization process, which iterates until the SQNR design requirement is met or the stop conditions of the optimization algorithm are reached.

Within the first $for-loop$ construct in the pseudo-code (lines 21 through 29), the coarse segmentation is performed based on the sign changes of the first and second order derivatives; the segments therein created are stored in the $mainSegmts$ collection. After this step, within the second $for-loop$ construct (lines 32 through 45), the segmentation tuning stage is performed according to the design parameters γ_{Th} and κ_{Th}. The following steps (lines 47 through 49) in the pseudocode are to compute the polynomial approximation coefficients through both, LSPA and mMPA methods, the fixed-point analysis, and the respective $SQNR_{LSPA}$ and $SQNR_{MMPA}$ responses. The ternary conditional construct on line 50 selects the higher SQNR response out of those obtained through the LSPA and the mMPA methods. The selected SQNR value is then provided to the cost function (line 51) to determine whether the target SQNR has been satisfied or further search should be carried out. The conditional constructs on lines 52 through 61 assess whether the SQNR requirement has been satisfied or the stop conditions have been reached; based on the result of these conditional evaluations, the optimization loop control flag is set or cleared for the search process to continue or stop, accordingly. Finally, the optimal set of polynomial coefficients from the optimized segmentation process is stored in the hardware LUT.

```
01:  Parameters definition: $x$, $y$, $x_{FxdPt}$, $y_{FxdPt}$, $h$, $g$, $Dx$, $Dg$, $x_L$, $x_H$, $quantElmts$,
     $mainSegmts$, $allSegmts_{LSPA}$, $allSegmts_{MMPA}$, $fwdSegmts$, $bwdSegmts$, $fwdCoeffs_{LSPA}$,
     $bwdCoeffs_{LSPA}$, $fwdCoeffs_{MMPA}$, $bwdCoeffs_{MMPA}$, $fwdCoeffs_{LSPA\_FxdPt}$, $bwdCoeffs_{LSPA\_FxdPt}$
     , $fwdCoeffs_{MMPA\_FxdPt}$, $bwdCoeffs_{MMPA\_FxdPt}$, $allCoeffs_{LSPA}$, $allCoeffs_{LSPA\_FxdPt}$,
        $allCoeffs_{MMPA}$, $allCoeffs_{MMPA\_FxdPt}$, $fwdSQNR_{LSPA}$, $bwdSQNR_{LSPA}$, $fwdSQNR_{MMPA}$,
        $bwdSQNR_{MMPA}$, $SQNR_{LSPA}$, $SQNR_{MMPA}$, $\gamma_{\mathrm{Th}}$, $\kappa_{\mathrm{Th}}$, $SQNR^{lb}$, $SQNR^{ub}$, $W_{Len}$, $m$,
        $accumLen$, $contSearch$, $i$, $j$, $k$
02:     Parameters initialization:
03:     Set $x_L \leftarrow \langle User\_Input \rangle$, default is 0
04:     Set $x_H \leftarrow \langle User\_Input \rangle$, default is 1
05:     $quantElmts \leftarrow \langle User\_Input \rangle$, default is $2^{10}$
06:     Set $Dx \leftarrow \frac{|x_H - x_L|}{quantElmts}$
07:     Set $x \leftarrow \mathrm{vector}(\mathrm{x_L} : Dx : \mathrm{x_H})$
08:     Set $\gamma_{\mathrm{Th}} \leftarrow \langle User\_Input \rangle$, default is 50%
09:     Set $\kappa_{\mathrm{Th}} \leftarrow \langle User\_Input \rangle$, default is 5%
10:     Set $[SQNR^{lb}, SQNR^{ub}] \leftarrow \langle User\_Input \rangle$, default is $[60\mathrm{dB}, 70\mathrm{dB}]$
11:     Set $W_{Len} \leftarrow \langle User\_Input \rangle$, default is 32bits
12:     Set $p_{degree} \leftarrow \langle User\_Input \rangle$, default is 2
13:     Set $i \leftarrow 1$, $j \leftarrow 1$, $k \leftarrow 1$
14:     Set $y \leftarrow \mathrm{funcEval}(f(x))$
15:     To load the initial $\gamma_{\mathrm{Th}}$ and $\kappa_{\mathrm{Th}}$ design parameters into optimization algorithm
16:     Do
17:             To clear required variables (Segments and Coefficients collections)
18:
19:             $mainSegmts$.createNewSegment()
20:             $mainSegmts(mainSegmts.\mathrm{count}).\mathrm{startIndex} \leftarrow 1$
21:             for loop $i \leftarrow 1 : \mathrm{length}(x)$
22:                     To compute $g_i \leftarrow \frac{df(x_i)}{dx}$ and $h_i \leftarrow \frac{d^2 f(x_i)}{dx^2}$
```

```
23:         Do coarse segmentation by finding sign changes in $g$ and $h$:
24:         **if** $\left(\text{sign}(g_i) \neq \text{sign}(g_{i+1})\right) \| \left(\text{sign}(h_i) \neq \text{sign}(h_{i+1})\right)$ **then**
25:             **Set** $mainSegmts(mainSegmts.\text{count}).\text{endIndex} \leftarrow (i-1)$
26:             $mainSegmts.\text{createNewSegment}()$
27:             **Set** $mainSegmts(mainSegmts.\text{count}).\text{startIndex} \leftarrow i$
28:         **end if**
29:     **end for**
30:     **Set** $mainSegmts(mainSegmts.\text{count}).\text{endIndex} \leftarrow \text{MaxIndexOf}(x)$
31:     **Set** $j \leftarrow 1$
32:     **for loop** $j \leftarrow 1 : mainSegmts.\text{count}$
33:         **Set** $fwdSegmts \leftarrow mainSegmts(j)$
34:         **Set** $bwdSegmts \leftarrow mainSegmts(j)$
35:         **Perform forward segmentation exploration:**
36:         **parfor loop** $i \leftarrow fwdSegmts(1).\text{startIndex} : fwdSegmts(1).\text{endIndex}$
37:             **Set** $accumLen \leftarrow fwdSegmts.\text{lastSegmt.lengthFromStartUpTo}(x_i)$
38:             To compute first derivative delta, $Dg \leftarrow \frac{|g_{fwdSegmts.\text{startIndex}} - g_i|}{|g_{fwdSegmts.\text{startIndex}}|} \times 100$
39:             **if** $(Dg \geq \gamma_{Th})$ and $(accumLen \geq \kappa_{Th})$ **then**
40:                 **To split current temporary main segment at $x_i$:**
41:                 $fwdSegmts.\text{lastSegmt.splitSegmtAt}(i)$
42:             **end if**
43:         **end parfor**
44:         **Perform backward segmentation exploration:**
45:         **parfor loop** $i \leftarrow bwdSegmts(1).\text{endIndex} : bwdSegmts(1).\text{startIndex}$
46:             **Set** $accumLen \leftarrow bwdSegmts.\text{firstSegmt.lengthFromEndUpTo}(x_i)$
47:             To compute first derivative delta, $Dg \leftarrow \frac{|g_{bwdSegmts.\text{endIndex}} - g_i|}{|g_{bwdSegmts.\text{endIndex}}|} \times 100$
```

```
48:             if $(Dg \geq \gamma_{Th})$ and $(\text{accumLen} \geq \kappa_{Th})$ then
49:                 To split first temporary main segment at $x_i$ :
50:                 $bwdSegmts$.firstSegmt.splitSegmtAt($i$)
51:             end if
52:         end parfor
53:         To compute the forward and backward segments coefficients:
54:         Set $fwdCoeffs_{LSPA} \leftarrow fwdSegmts$.computeLSPA($m$)
55:         Set $fwdCoeffs_{MMPA} \leftarrow fwdSegmts$.computeMMPA($m$)
56:         Set $bwdCoeffs_{LSPA} \leftarrow bwdSegmts$.computeLSPA($m$)
57:         Set $bwdCoeffs_{MMPA} \leftarrow bwdSegmts$.computeMMPA($m$)
58:         To compute fixed-point analysis for the given $W_{Len}$: …
              ( $fwdCoeffs_{LSPA\_FxdPt}$, $fwdCoeffs_{MMPA\_FxdPt}$, $bwdCoeffs_{LSPA\_FxdPt}$,
              $bwdCoeffs_{MMPA\_FxdPt}$, $x_{FxdPt}$, $y_{FxdPt}$ )
59:         To compute $j^{th}$ main segment SQNR for forward and
            backward segmentation…
60:           ( $fwdSQNR_{LSPA}$, $fwdSQNR_{MMPA}$, $bwdSQNR_{LSPA}$, $bwdSQNR_{MMPA}$ )
61:         To select the segmentation direction of higher SQNR:
62:         if $(fwdSQNR_{LSPA} > bwdSQNR_{LSPA})$
63:             $allSegmts_{LSPA}$.addSegments($fwdSegmts$)
64:             Set $allCoeffs_{LSPA} \leftarrow fwdCoeffs_{LSPA}$
65:             Set $allCoeffs_{LSPA\_FxdPt} \leftarrow fwdCoeffs_{LSPA\_FxdPt}$
66:         else
67:             $allSegmts_{LSPA}$.addSegments($bwdSegmts$)
68:             Set $allCoeffs_{LSPA} \leftarrow bwdCoeffs_{LSPA}$
69:             Set $allCoeffs_{LSPA\_FxdPt} \leftarrow bwdCoeffs_{LSPA\_FxdPt}$
```

```
70:                 end if
71:                 if $\left(fwdSQNR_{MMPA} > bwdSQNR_{MMPA}\right)$
72:                     $allSegmts_{MMPA}$.addSegments($fwdSegmts$)
73:                     Set $allCoeffs_{MMPA} \leftarrow fwdCoeffs_{MMPA}$
74:                     Set $allCoeffs_{MMPA\_FxdPt} \leftarrow fwdCoeffs_{MMPA\_FxdPt}$
75:                 else
76:                     $allSegmts_{MMPA}$.addSegments($bwdSegmts$)
77:                     Set $allCoeffs_{MMPA} \leftarrow bwdCoeffs_{MMPA}$
78:                     Set $allCoeffs_{MMPA\_FxdPt} \leftarrow bwdCoeffs_{MMPA\_FxdPt}$
79:                 end if
80:             end for
81:             To compute overall SQNR for the $k^{th}$ optimization iteration: …
                    ($SQNR_{LSPA}$, $SQNR_{MMPA}$)
82:             if stop conditions have been met? then
83:                 $continueSearch \leftarrow FALSE$
84:             Else
85:                 if $\left(SQNR^{lb} \le SQNR_{LSPA} \le SQNR^{ub}\right)$ and $\left(SQNR^{lb} \le SQNR_{MMPA} \le SQNR^{ub}\right)$ then
86:                     To search for alternative design parameters: ($\gamma_{Th}$, $\kappa_{Th}$)
87:                     $continueSearch \leftarrow TRUE$
88:                 else
89:                     $continueSearch \leftarrow FALSE$
90:                 end if
91:             end if
92:             To increment optimization iterations counter:
93:             Set $j \leftarrow j+1$
94:     while ($continueSearch$)
95:     To store the coefficients that deliver best overall SQNR:
```

```
96:     $coeffsLUT_{LSPA} \leftarrow allCoeffs_{LSPA\_FxdPt}$
97:     $coeffsLUT_{MMPA} \leftarrow allCoeffs_{MMPA\_FxdPt}$
98:     $LUTSegmts_{LSPA\_FxdPt} \leftarrow allSegmts_{LSPA\_FxdPt}$
99:     $LUTSegmts_{MMPA\_FxdPt} \leftarrow allSegmts_{MMPA\_FxdPt}$
```

Algorithm 1: Algorithmic description of the adaptive function segmentation methodology.

3. Segmentation Methodology Implementation

3.1 Segmentation Algorithm Software Architecture

The following diagram depicts the overall functional architecture of the implemented MATLAB code for the adaptive function segmentation methodology. Each square box represents a MATLAB function, and the hierarchical enclosing of boxes convey the actual dependencies across functions.

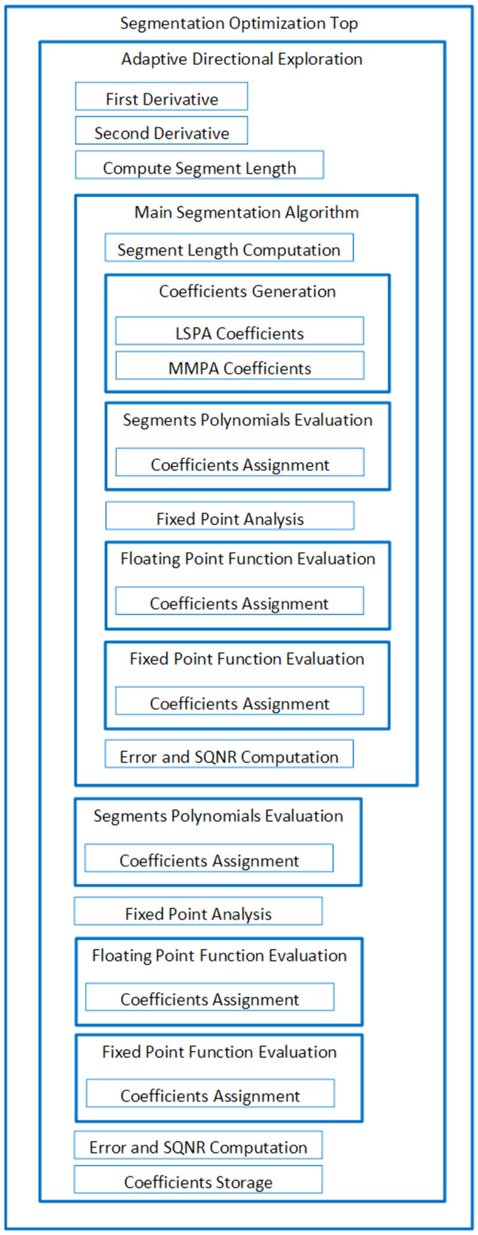

Segmentation Optimization Top
Adaptive Directional Exploration
First Derivative
Second Derivative
Compute Segment Length
Main Segmentation Algorithm
Segment Length Computation
Coefficients Generation
LSPA Coefficients
MMPA Coefficients
Segments Polynomials Evaluation
Coefficients Assignment
Fixed Point Analysis
Floating Point Function Evaluation
Coefficients Assignment
Fixed Point Function Evaluation
Coefficients Assignment
Error and SQNR Computation
Segments Polynomials Evaluation
Coefficients Assignment
Fixed Point Analysis
Floating Point Function Evaluation
Coefficients Assignment
Fixed Point Function Evaluation
Coefficients Assignment
Error and SQNR Computation
Coefficients Storage

Fig. 3-1: Software architecture of the adaptive function segmentation methodology.

3.2 Functional Description of the Software Architecture Modules

3.2.1 Adaptive Directional Exploration

This function is the driver of the optimization algorithm implemented for the bidirectional adaptive segmentation methodology. It takes the values of the design parameters subject to optimization $d_0 = [\gamma_{TH}, \kappa_{TH}]$, and a set of predefined design parameters through the vector d_P. Also, this function requires some global variables to be defined in the top file and set with the appropriate values for the correct functionality of the algorithm. The input parameters, the global variables, as well as the output parameters of the function are described in further detail in TABLE 2.

TABLE 2: LIST OF THE INPUT PARAMETERS, GLOBAL VARIABLES AND OUTPUT PARAMETERS OF THE ADAPTIVE DIRECTIONAL EXPLORATION FUNCTION.

Parameter definition	Description
$d_0 = […]$	Vector with the initial design parameters for the adaptive segmentation algorithm. **deriv_delta:** First derivative threshold γ_{TH} expressed as a percentage; it takes values greater than 0% up to values that make sense for the function at hand, let say X_0 = 500% for a 5-times derivative change from the previous segment endpoint. **min_seg_length:** The minimum chordal segment length threshold. This value is expressed as a percentage; valid values are those greater than 0% and smaller than 100%.
$d_P = […]$	Vector for the predefined design parameters, which are listed as follows: **word_length:** The system word length, the default value is 32 bits. **step_size:** Number of subsequent samples on the *x* vector to skip for the calculation of the chordal segment length. **samples_power:** Amount of samples in which the evaluation interval *X* is to be quantized.

	min_x: The lower value of the evaluation interval *X*, which was previously introduced as x_L. **max_x:** The upper value of the evaluation interval *X*, which was previously introduced as x_H. **poly_degree:** Polynomial degree to be employed for the polynomial approximation, which was previously introduced as *m*. **step_factor:** Amount of subsequent quantization samples of the *x* vector to skip throughout the sweep of the fine-tuning derivative exploration, the default is 1 (No samples are skipped, sample_index = sample_index + step_factor).
Global variables	**Description**
global funct	A global variable that stores the function handler to be segmented out. The signature of the function is as follows: `funct = @(x)function_name(parameters in terms of x)`
global exec_count	Global counter variable utilized to achieve the execution of certain initialization code within optimization procedure only for the first iteration of the segmentation algorithm. The user does not need to set this parameter.
global approx_method	The global variable used by the algorithm to select which approximation method should be utilized for the computation of the polynomials. 0: Least Square Polynomial Approximation (LSPA). 1: miniMax Polynomial Approximation (mMPA).
Output parameters	**Description**

seg_bounds	A vector containing the collection of segments endpoints/boundaries, (the values in the evaluation interval *X* where a segment ends and the following begins).
SQNR	The SQNR result from current segmentation realization.
Data	A vector that contains the following information about the current segmentation realization. **seg_bounds:** Vector that holds the collection of indexes of the vector *x* for the defined segments endpoints/boundaries. **boundaries**: Vector that holds the collection of values within the vector *x* for the defined segments endpoints/boundaries. **boundaries_fxp:** Collection that contains the values of the vector *x* in fixed-point representation for the endpoints of the defined segment. **vect_x:** Vector of the quantized evaluation interval *X*. **vect_eval_y:** Vector that contains the results of the evaluation of the function for each element in vect_x. **fltPnt_poly_vect_eval_y:** Vector that contains the results of the evaluation of the functions' polynomial approximation in floating-point representation for each element in vect_x. **fxdPnt_poly_vect_eval_y:** Vector that contains the results of the evaluation of the function approximated through the segments polynomials in fixed-point representation for each element in vect_x. **fixedPoint_vect_x_obj.data:** The vector of the quantized interval *X* in fixed-point representation. **Error_FltPntGolden_to_FltPntPoly:** Vector that contains the absolute errors of approximation between the original function and the polynomial approximation in floating-point representation. **Error_FltPntGolden_to_FxdPntPoly:** Vector that contains the absolute errors of approximation between the original function and the polynomial approximation in fixed-point representation.

	samples_power: The number of samples in which the evaluation interval *X* was quantized, ($2^{\text{samples\_power}}$). **QI_MaxCoeff:** Number of bits required to represent the integer part of the maximum number required. **QF_Xargument:** Number of bits remaining, from the predefined word length and the required QI bits for the representation of the floating portion of the numbers. **D1_collection:** Vector with the values of the functions' first order derivative at every point in vect_x. **D2_collection:** Vector with the values of the functions' second-order derivative at every point in vect_x.

.

3.2.2 First Derivative

This function computes the first order derivative of the function at the specified point in *x*. This function implements the centered differencing formula [22] to get a more accurate approximation of the first order derivative of *f(x)*. The details of the input and output parameters are given in TABLE 3.

TABLE 3: INPUT AND OUTPUT PARAMETERS OF THE FIRST DERIVATIVE FUNCTION.

Parameter definition	Description
Fun	Function handler with the signature: `funct = @(x)function_name(parameters in terms of x)`
x_0	Point in *x* where to evaluate the first order derivative of the function.
vect_x	The vector of the quantized evaluation interval *X*. In this case, this vector is employed to handle the computation of the derivative for those functions that are undefined outside of the evaluation interval.

Output parameters	**Description**
D	The value of the first order derivative at the x_0.

3.2.3 Second Derivative

This function computes the second order derivative of the function to be approximated, at the specified point within the evaluation interval *X*. This function implements the fifth stencil of the centered differencing formula [23] to get a more accurate and stable approximation of the second order derivative of *f(x)*. The details of the input and output parameters are given in TABLE 4.

TABLE 4: INPUT AND OUTPUT PARAMETERS OF THE SECOND DERIVATIVE FUNCTION.

Parameter definition	**Description**
Fun	Function handler with the following function signature: funct = @(x)function_name(parameters_in_terms_of_x)
x_0	The point within *X* where to evaluate the second order derivative of the function.
vect_x	The vector of the quantized evaluation interval *X*. In this case, this vector is employed to handle the computation of the derivative at the boundaries of the evaluation interval.
Output parameters	**Description**
D	The value of the second order derivative of the function at x_0.

3.2.4 Segment Length Computation

This function computes the chordal length of the function within a given interval. The details of the input and output parameters are given in TABLE 5.

TABLE 5: INPUT AND OUTPUT PARAMETERS OF THE SEGMENT LENGHT COMPUTATION FUNCTION.

Parameter definition	**Description**
vect_x	The vector of the quantized evaluation interval *X*.
vect_eval_y	The vector that contains the results of the function evaluation for every element in vect_x.
step_size	Number of samples to skip between subsequent iterations along the sweep of the interval of evaluation *X*. This parameter allows speeding up the computation of the chordal length at the expense of lost in accuracy.
Output parameters	**Description**
segment_length	The value of the chordal segment length for the interval of evaluation in vect_x.

3.2.5 Main Segmentation Algorithm

This function implements the actual fine-tuning bidirectional segmentation algorithm according to the parameters provided by the optimization process. The details of the input and output parameters are given in TABLE 6.

TABLE 6: INPUT AND OUTPUT PARAMETERS OF THE MAIN SEGMENTATION ALGORITHM FUNCTION.

Parameter definition	Description
startPoint	Index inside the vector vect_x where to start the fine-tuning bidirectional segmentation exploration.
approx_method	Polynomial approximation method that should be used to compute the segments polynomials.
derivative_criteria	An input parameter that is used to alter the behavior of derivative threshold design parameter. 0: The absolute derivative change between the current x_i point and the previous segment endpoint should be compared against the derivative threshold expressed as a percentage. 1: The absolute derivative change between the current x_i point and the previous segment endpoint should be compared against the derivative threshold expressed as a percentage of the absolute range of derivative values within the whole evaluation interval X. 2: The absolute derivative change between the current x_i point and the previous segment endpoint should be compared against the derivative threshold expressed as a percentage of the absolute range of derivative values within the interval of evaluation that has not yet been segmented out. 3: The absolute derivative change between the current x_i point and the previous segment endpoint should be compared against the derivative threshold expressed as a percentage of the average of the range of derivative values within the specified evaluation interval.

	4: The absolute derivative change between the current x_i point and the previous segment endpoint should be compared against the derivative threshold expressed as a percentage of the average of the range of derivative values within the interval of evaluation that has not yet been segmented out.
deriv_delta	The first order derivative threshold γ_{TH}.
vect_x	The vector of the quantized evaluation interval X.
vect_eval_y	The vector that contains the results from the evaluation of the function for every element in vect_x.
D1_collection	Vector with the first order derivative values of the function at every point in vect_x.
step_factor	Number of samples in vect_x to skip for each iteration of the segmentation exploration.
poly_degree	The polynomial degree to be employed for the polynomial approximation, which was previously introduced as m.
WordLength	The predefined word length of the system.
chunk_length	The input parameter for the minimum length allowed for the trailing segment. It controls whether the remaining of the evaluation interval which does not meet the design thresholds (γ_{TH} and κ_{TH}) is defined as a new segment or merged with the previous one.
step_size	Number of samples in vect_x to skip for each iteration of the chordal length calculation loop.
min_seg_length	The minimum segment chordal length threshold κ_{TH}.
Output parameters	**Description**
seg_bounds	A collection that contains the indexes of vect_x for the defined segments endpoints/boundaries.
SQNR	The SQNR result from current segmentation realization.
Data	A vector that contains information about the current segmentation realization as described in TABLE 2.

3.2.6 Coefficients Generation

This function computes the coefficients of the polynomial to approximate all the defined segments within a given evaluation interval. The details of the input and output parameters are given in TABLE 7.

TABLE 7: INPUT AND OUTPUT PARAMETERS OF THE COEFFICIENTS GENERATION FUNCTION.

Parameter definition	**Description**
segment_bounds	Collection that contains the indexes of vect_x for the defined segments endpoints/boundaries.
poly_degree	The polynomial degree to be employed for the polynomial approximation, which was previously introduced as *m*.
vect_x	The vector of the quantized evaluation interval *X*.
vect_eval_y	The vector that contains the results from the evaluation of the function for every element in vect_x.
approx_method	Polynomial approximation method that should be used to compute the segments polynomials.
Output parameters	**Description**
polynomial_coefficients	The vector that contains the collection of coefficients for all the defined segments in the evaluation interval.

3.2.7 LSPA Coefficients

This function computes the LSPA polynomials to approximate all the defined segments within the evaluation interval. The details of the input and output parameters are given in TABLE 8.

TABLE 8: INPUT AND OUTPUT PARAMETERS OF THE LSPA COEFFICIENTS FUNCTION.

Parameter definition	**Description**
segment_bounds	Collection that contains the indexes of vect_x for the defined segments endpoints/boundaries.
poly_degree	The polynomial degree to be employed for the polynomial approximation, which was previously introduced as *m*.
vect_x	The vector of the quantized evaluation interval *X*.
vect_eval_y	The vector that contains the results from the evaluation of the function for every element in vect_x.
Output parameters	**Description**
polynomial_coefficients	The vector that contains the collection of LSPA coefficients for all the defined segments in the evaluation interval.

3.2.8 MMPA Coefficients

This function computes the mMPA polynomials to approximate all the defined segments within the evaluation interval. The details of the input and output parameters are given in TABLE 9.

TABLE 9: INPUT AND OUTPUT PARAMETERS OF THE MMPA COEFFICIENTS FUNCTION.

Parameter definition	**Description**
segment_bounds	Collection that contains the indexes of vect_x for the defined segments endpoints/boundaries.
poly_degree	The polynomial degree to be employed for the polynomial approximation, which was previously introduced as *m*.
vect_x	The vector of the quantized evaluation interval *X*.
vect_eval_y	The vector that contains the results from the evaluation of the function for every element in vect_x.
Output parameters	**Description**
polynomial_coefficients	The vector that contains the collection of mMPA coefficients for all the defined segments in the evaluation interval.

3.2.9 Segments Polynomials Evaluation

This function performs the floating point evaluation of the function through approximated polynomials. The details of the input and output parameters are given in TABLE 10.

TABLE 10: INPUT AND OUTPUT PARAMETERS OF THE SEGMENTS POLYNOMIALS EVALUATION FUNCTION.

Parameter definition	**Description**
posx_values	Collection that contains the values of *x* for the defined segments endpoints/boundaries.
coef_ram	A vector that contains the collection of coefficients for all the defined segments in the evaluation interval.
vect_x	The vector of the quantized evaluation interval *X*.
Output parameters	**Description**
fltPnt_poly_vect_eval_y	A vector that contains the results of the evaluation of the function through the polynomial approximation in floating-point approximation.

3.2.10 Coefficients Assignment

This function assigns the polynomial coefficients to the corresponding segment. The details of the input and output parameters are given in TABLE 11.

TABLE 11: INPUT AND OUTPUT PARAMETERS OF THE COEFFICIENTS ASSIGNMENT FUNCTION.

Parameter definition	**Description**
posx_values	Collection that contains the values of *x* for the defined segments endpoints/boundaries.

coef_ram	A vector that contains the collection of coefficients for all the defined segments in the evaluation interval.
Xdata	The vector of the quantized evaluation interval *X*.
Output parameters	**Description**
Assigned	The matrix that contains the coefficients arranged correspondingly to each defined segment.

3.2.11 Fixed Point Analysis

This function performs the fixed-point analysis to determine the correct configuration to appropriately represent all the numbers within the evaluation interval, as well as the values of the domain of the function and the polynomial's coefficients. The details of the input and output parameters are given in TABLE 12.

TABLE 12: INPUT AND OUTPUT PARAMETERS OF THE FIXED POINT ANALYSIS FUNCTION.

Parameter definition	Description
coef_ram	The matrix that contains the coefficients of the defined segments.
vect_x	The vector of the quantized evaluation interval *X*.
vect_eval_y	A vector that contains the results of the evaluation of the function for each point in vect_x.
word_length	Predefined system word length.
Output parameters	**Description**
QI_MaxCoeff	The number of bits required to represent the integer part of the maximum number required.
QF_Xargument	The number of bits remaining, from the predefined word length and the required *QI* bits, for the representation of the fractional portion of the numbers.

range_min_limit	Minimum number that can be represented by the fixed-point configuration Q[QI_MaxCoeff, QF_Xagument].
range_max_limit	The maximum number that can be represented with the fixed-point configuration Q[QI_MaxCoeff, QF_Xagument].
fixed_point_resolution	Resolution provided by the fixed-point configuration Q[QI_MaxCoeff, QF_Xagument].
coef_ramA_fxp_obj	MATLAB fixed-point object that holds the fixed-point values of the p_0 coefficients for all the defined segments.
coef_ramB_fxp_obj	MATLAB fixed-point object that holds the fixed-point values of the p_1 coefficients for all the defined segments.
coef_ramC_fxp_obj	MATLAB fixed-point object that holds the fixed-point values of the p_2 coefficients for all the defined segments.
fixedPoint_vect_x_obj	MATLAB fixed-point object that holds the fixed-point values of the evaluation interval x.

3.2.12 Floating Point Function Evaluation

This function performs the evaluation of the function through the polynomials approximation using floating-point representation. The details of the input and output parameters are given in TABLE 13.

TABLE 13: INPUT AND OUTPUT PARAMETERS OF THE FLOATING POINT EVALUATION FUNCTION.

Parameter definition	Description
fltPnt_posx_values	Collection that contains the values within x for the defined segments endpoints/boundaries in floating-point representation.
coef_ram	The matrix that contains the coefficients of the defined segments.
fxdPnt_vect_x	A vector that contains the values of x that conform the evaluation interval in fixed-point representation.

vect_x	A vector that contains the values of *x* that conform the evaluation interval.
Output parameters	**Description**
fltPnt_poly_vect_eval_y	A vector that contains the results of the evaluation of the function through the polynomial approximation in floating-point approximation.

3.2.13 Fixed Point Function Evaluation

This function performs the evaluation of the function through the polynomials approximation using fixed-point representation. The details of the input and output parameters are given in TABLE 14.

TABLE 14: INPUT AND OUTPUT PARAMETERS OF THE FIXED POINT FUNCTION EVALUATION FUNCTION.

Parameter definition	Description
fxdPnt_posx_values	Collection that contains the values within *x* for the defined segments endpoints/boundaries in fixed-point representation.
coef_ramA_fxp	A vector that contains the values of the polynomial coefficient p_0 in fixed-point representation.
coef_ramB_fxp	A vector that contains the values of the polynomial coefficient p_1 in fixed-point representation.
coef_ramC_fxp	A vector that contains the values of the polynomial coefficient p_2 in fixed-point representation.
fxdPnt_vect_x	A vector that contains the values of *x* that conform the evaluation interval in fixed-point representation.
WordLength	Predefined system word length, the default is 32 bits.
QF_Xargument	The number of bits remaining from the predefined word length and the required *QI* bits, for the representation of the floating portion of the numbers.

Output parameters	Description
fxdPnt_poly_vect_eval_y	A vector that contains the results of the evaluation of the function through the polynomial approximation in fixed-point representation.

3.2.14 Error and SQNR Computation

This function computes the vector of absolute approximation error and the SQNR response from the performed segmentation realization. The details of the input and output parameters are given in TABLE 15.

TABLE 15: INPUT AND OUTPUT PARAMETERS OF THE ERROR AND SQNR COMPUTATION FUNCTION.

Parameter definition	Description
vect_eval_y	A vector that contains the results of the evaluation of the function for each element in vect_x.
fltPnt_poly_vect_eval_y	A vector that contains the results of the evaluation of the function for each element in vect_x using floating-point representation.
fxdPnt_poly_vect_eval_y	A vector that contains the results of the evaluation of the function for each element in vec_x using fixed-point representation.
Output parameters	**Description**
Error_FltPntGolden_to_FltPntPoly	A vector that contains the absolute errors of approximation between the original function and the polynomial approximation in floating-point representation.
Error_FltPntGolden_to_FxdPntPoly	A vector that contains the absolute errors of approximation between the original function and the polynomial approximation in fixed-point representation.

SQNR	The value of the SQNR response of the performed segmentation realization.

3.2.15 Coefficients Storage

This function creates the Verilog code for the ROM blocks that store the segments endpoints and the corresponding polynomial coefficients. The details of the input and output parameters are given in TABLE 16.

TABLE 16: INPUT AND OUTPUT PARAMETERS OF THE COEFFICIENTS STORAGE FUNCTION.

Parameter definition	Description
fixedPoint_vect_x_obj	Vector that contains the values of *x* that conform the evaluation interval in fixed-point representation.
fxdPnt_posx_values	Collection that contains the values of *x* for the defined segments endpoints/boundaries in fixed-point representation.
coef_ramA_fxp_ob	A vector that contains the values of the polynomial coefficient p_0 in fixed-point representation.
coef_ramB_fxp_obj	A vector that contains the values of the polynomial coefficient p_1 in fixed-point representation.
coef_ramC_fxp_obj	A vector that contains the values of the polynomial coefficient p_2 in fixed-point representation.

4 Function Segmentation Tests and Results

The segmentation performance and approximation accuracy of the proposed AFSM were evaluated for the set of test bench functions listed in TABLE 17. These functions are widely employed to construct hardware blocks with application in the fields of numerical analysis, digital signal processing, wireless channel emulation, artificial neural networks [24], amongst others.

For all the test bench functions, the optimization process of the segmentation algorithm was set up to maintain the output SQNR within the specified range, 60dB to 70dB. TABLE 17 summarizes the approximation results from the proposed AFSM employing both Least Squares and miniMax PPA methods. The columns "$\gamma_{Th}^{*}(\%)$" and "$\kappa_{Th}^{*}(\%)$" present the optimal design parameters (first order derivative and minimum chordal length thresholds) of the segmentation algorithm that satisfy the SQRN requirement. The column "$SQNR^{*}(dB)$" presents the achieved SQNR through the optimized design parameters in columns "$\gamma^{*}(\%)$" and "$\kappa_{Th}^{*}(\%)$". The column "Required Segments" shows the minimum number of segments needed to meet the SQNR requirement.

The columns "QI (bits)" and "QF (bits)" present the number of bits assigned to the integer and fractional parts of the fixed-point representation of the polynomial coefficients, the range, and the domain of the approximated function. The maximum absolute error of approximation between each function and its piecewise polynomial approximation is presented in the "Max |Error|" column. Finally, the column "ROM (Bytes)" shows the bytes of memory required by the LUT for the storage of the polynomial's coefficients of all the segments needed to achieve the SQNR requirement for each PPA method tested; the memory requirements are calculated as $ROM_{Bytes} = \frac{W_{Len}}{8} \times \text{Required\_Segments} \times (m+1)$.

4. Function Segmentation Tests and Results

Although the proposed AFSM can be employed to approximate transcendental functions using polynomials of any degree, to reduce the number of coefficients required for each segment, second-degree polynomials were used for both Least Square and miniMax methods. In this sense, the polynomial approximation tests were carried out with a uniform word length of $W_{\text{Len}} = 32$ bits. This decision is supported by the fact that most modern field programmable gate arrays (FPGA) or systems on a chip (SoC) have these or even greater bus width capabilities; therefore, no additional resources expenditure is required.

TABLE 17: SEGMENTATION AND APPROXIMATION ACCURACY RESULTS FROM THE PROPOSED AFSM FOR BOTH LS AND MINIMAX PPA METHODS. THESE RESULTS WHERE OBTAINED USING $W_{\text{Len}} = 32$ BITS AND POLYNOMIALS OF DEGREE $m = 2$.

	PPA Method	γ^*_{Th} (%)	κ^*_{Th} (%)	$SQNR^*$ dB	Required Segments	QI (bits)	QF (bits)	Max |Error|	ROM (Bytes)
$f_1(x) = \sqrt{(x)}$	LSPA	28.5	10	65.25	7	13	19	0.0099	84
	mMPA	41	10	62.48	6	12	20	0.0055	72
$f_2(x) = \frac{1}{\sqrt{x}}$	LSPA	30	30	66.85	14	16	16	0.0075	168
	mMPA	93.4	10	63.12	10	16	16	0.0096	120
$f_3(x) = \sin(x)$	LSPA	60	25	66.78	12	5	27	0.0013	144
	mMPA	91.8	25	64.61	12	5	27	0.0007	144
$f_4(x) = \frac{x}{2}\log_2(x)$	LSPA	72	20	64.71	9	6	26	0.0019	108
	mMPA	95.7	20	64.36	8	7	25	0.0005	96
$f_5(x) = \cos^{-1}(x)$	LSPA	4	10	64.52	9	11	21	0.0338	108
	mMPA	10	10	64.66	8	14	28	0.0078	96
$f_6(x) = \sqrt{-\ln(x)}$	LSPA	37	15	62.28	12	15	17	0.0146	108
	mMPA	20	15	60.24	12	15	17	0.007	96
$f_7(x) = \ln(1+x)$	LSPA	90	40	64.49	2	2	30	0.0008	144
	mMPA	40	40	63.22	2	2	30	0.0005	144
$f_8(x) = \frac{1}{1+x}$	LSPA	100	30	62.16	2	2	30	0.0019	24
	mMPA	100	30	60.68	2	2	30	0.0011	24

$f_9(x) = \dfrac{0.0004x + 0.0002x}{x^4 - 1.96x^3 + 1.348x^2 - 0.378x + 0.0373}$									
	LSPA	143	8	60.52	30	11	21	0.0031	360
	mMPA	143	8	60.32	30	11	21	0.0020	360
$f_{10}(x) = \tan\text{sig}(x)$	LSPA	50	25	62.02	8	33	29	0.0025	96
	mMPA	50	25	60.08	8	33	29	0.0014	96

One can observe in TABLE 17 that for the functions *$f_1(x)$*, *$f_4(x)$*, and *$f_5(x)$* one less segment is needed to reach the target SQNR when the polynomial approximation is carried out through the mMPA method than when it is performed through the LSPA method. Furthermore, given that the mMPA finds the polynomial coefficients that minimize the maximum error of approximation, for most of the test bench functions, the maximum absolute error achieved through the mMPA method was smaller in comparison to that obtained through the LSPA method. However, one can observe that for the functions *$f_3(x)$*, *$f_6(x)$*, *$f_7(x)$*, *$f_8(x)$*, and *$f_9(x)$* the achieved SQNR though the mMPA method was slightly lower in comparison to that obtained through LSPA method. The reason of this is that the objective of the LSPA method is to find a set of polynomial coefficients for each segment that minimize the sum of the squared residual between the original function and the approximating polynomial. Consequently, the denominator of the SQNR expression in (2-16) that accounts for the quantization noise is minimized explicitly.

TABLE 18: SEGMENTATION PERFORMANCE COMPARISON BETWEEN THE PROPOSED AFSM VERSUS THE UNIFORM AND THE NON-UNIFORM-BY-POWERS-OF-TWO SEGMENTATION METHODOLOGIES.

Function	Segmentation Methodology	Required Segments	QI (bits)	QF (bits)	SQNR (dB)
$f_5(x) = \cos^{-1}(x)$	AFTM	8	14	18	64.66
	Uniform	128	13	19	58.64
	Non-Uniform	8	13	19	66.53
$f_6(x) = \sqrt{-\ln(x)}$	AFSM	12	15	17	62.28
	Uniform	128	18	14	60.95
	Non-Uniform	16	15	17	65.74

$f_9(x) = \dfrac{0.0004x + 0.0002x}{x^4 - 1.96x^3 + 1.348x^2 - 0.378x + 0.0373}$	AFSM	30	11	21	60.52
	Uniform	64	11	21	61.96
	Non-Uniform	32	11	21	46.80
$f_{10}(x) = \tan\mathrm{sig}(x)$	AFSM	8	3	28	62.02
	Uniform	8	3	29	59.62
	Non-Uniform	16	3	29	61.19

For the functions $f_5(x)$, $f_6(x)$, $f_9(x)$, and $f_{10}(x)$, TABLE 18 shows a comparison of the approximation performance obtained through the proposed AFSM, the uniform, and non-uniform-by-powers-of-two segmentation methodologies for an SQNR requirement between 60 dB and 70 dB. These functions were selected for comparison because these present curvature features that are challenging to approximate through a basic segmentation methodology alone; prove of this is that for functions such as $f_9(x)$ and $f_{10}(x)$ the SQNR requirements was not satisfied employing the non-uniform and the uniform segmentation methodologies, respectively.

For example, given the specified SQNR, $f_5(x)$ can be approximated using only eight segments through both the proposed AFSM (plotted in Fig. 4-1) and the non-uniform-by-powers-of-two methodology (plotted in Fig. 4-2). On the other hand, the uniform segmentation methodology, plotted in Fig. 4-3, does not perform satisfactorily because an excessive number of 128 segments are required in an attempt to reduce the absolute approximation error shown in Fig. 4-4, which increases as the curvature of $f_5(x)$ increases. Similarly, the uniform segmentation methodology for the functions $f_6(x)$ and $f_9(x)$ (plotted in Fig. 4-7 and Fig. 4-11 respectively) requires a significantly greater amount of segments compared to the proposed AFSM. In this sense, for the functions $f_6(x)$ and $f_9(x)$, the uniform segmentation methodology requires 128 and 64 segments respectively, while the proposed AFSM requires only 12 and 30 segments, respectively.

The advantages of the proposed AFSM, over the previously discussed basic segmentation methodologies, are demonstrated through the more elaborated curvature shapes of the functions $f_6(x)$, $f_9(x)$, and $f_{10}(x)$, which are plotted in Fig. 4-5, Fig. 4-9, and Fig. 4-13, accordingly. For these test functions, the proposed AFSM meets the SQNR requirement with the minimum number of segments amongst the comparing segmentation methodologies. Also, and most importantly, through the proposed AFSM, the segmentation and approximation procedure was automatically performed and optimized according to the evolution of the curvature shape without intervention from the user.

In contrast, in order to apply the non-uniform-by-powers-of-two segmentation methodology on these functions, the user should intervene in the definition of a segmentation hierarchy within the sub-intervals in X. This segmentation hierarchy is needed to change the direction of segmentation to match the evolution of the function's shape and allocate more segments to the regions with increasing curvature [14]. An example of this is shown in Fig. 4-6, where the evaluation interval of $f_6(x)$ was first divided in half at $x=0.5$ using uniform segmentation. Then starting at $x=0.5$, the sub-interval (0, 0.5] was hierarchically segmented from right to left using the non-uniform-by-powers-of-two segmentation. Finally, the sub-interval (0.5, 1] was segmented out using the non-uniform-by-powers-of-two segmentation from left to right. Likewise, for $f_9(x)$ in Fig. 4-10, and for $f_{10}(x)$ in Fig. 4-14, the first level of the segmentation hierarchy divides the evaluation interval into four uniform sub-intervals. Then, for the second segmentation level of both $f_9(x)$ and $f_{10}(x)$, each of the uniformly divided sub-intervals is hierarchically segmented using the non-uniform-by-powers-of-two segmentation in the direction (left to right or vice versa) that allocates the maximum number of segments to the regions of higher curvature.

As it was already mentioned, an important drawback of the hierarchical segmentation methodology is that the user should determine the most appropriate direction of segmentation through visual inspection of the functions' shape. In this sense, one can observe on TABLE 18 that for $f_9(x)$, plotted in Fig. 4-10, the hierarchical segmentation does not meet the SQNR requirements because the endpoints of the uniformly spaced segments do not quite match with the regions where the function presents the higher curvature. As a consequence, the tightly spaced segments from the second level non-uniform segmentation are defined at inappropriate locations, causing the error of approximation to increase at the regions of the function that present the maximum curvature.

The uniform segmentation of $f_5(x)$, $f_6(x)$, and $f_9(x)$ is shown in Fig. 4-3, Fig. 4-7, and Fig. 4-11, respectively. One can observe that the uniform segmentation of these functions requires an excessive number of segments because this technique is not appropriate for functions with highly varying curvature shapes. On the contrary, since the evolution of the curvature of $f_{10}(x)$ in Fig. 4-15 is fairly smooth, the uniform segmentation delivers similar results to the proposed AFSM.

The plots of the absolute error of approximation for $f_5(x)$, $f_6(x)$, $f_9(x)$ and $f_{10}(x)$ are shown in Fig. 4-4, Fig. 4-8, Fig. 4-12, and Fig. 4-16 respectively. In these plots, one can clearly observe that the proposed AFSM does not deliver the minimum absolute approximation error at every point within the evaluation interval; instead of that, the approximation error is controlled and balanced according to the evolution of the curvature of an arbitrary function. The previous is an important effect that allows the proposed segmentation algorithm to adapt to functions of arbitrary shape and achieve a good balance between the number of segments and the accuracy requirements.

The advantages of the AFSM over the compared segmentation methodologies in term of the required number of segments is directly translated into a significant reduction memory resources required to store the LUT of polynomial coefficients. As an example, to achieve similar accuracy results for $f_5(x)$ and $f_6(x)$, the uniform segmentation requires a total of 128 segments, which translates to 1536 bytes of ROM. On the other hand, through the AFSM, for $f_5(x)$ only eight segments (96 bytes) are required, and for $f_6(x)$ only 12 segments (144 bytes) are required respectively. The previous calculations account for a 1600% and a 1066.66% reduction of the corresponding memory resources.

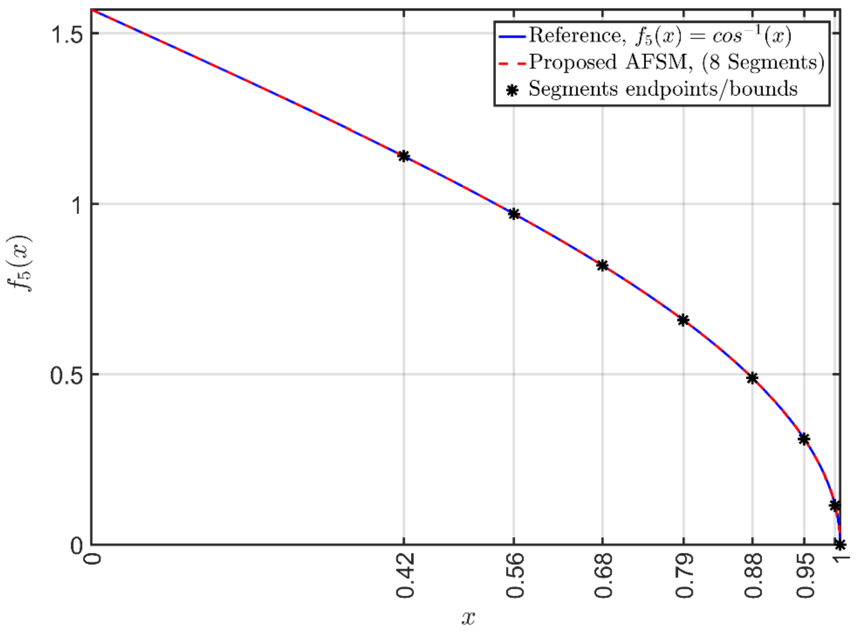


Fig. 4-1. Segmentation and approximation result for *f₅(x)* through the proposed AFSM.

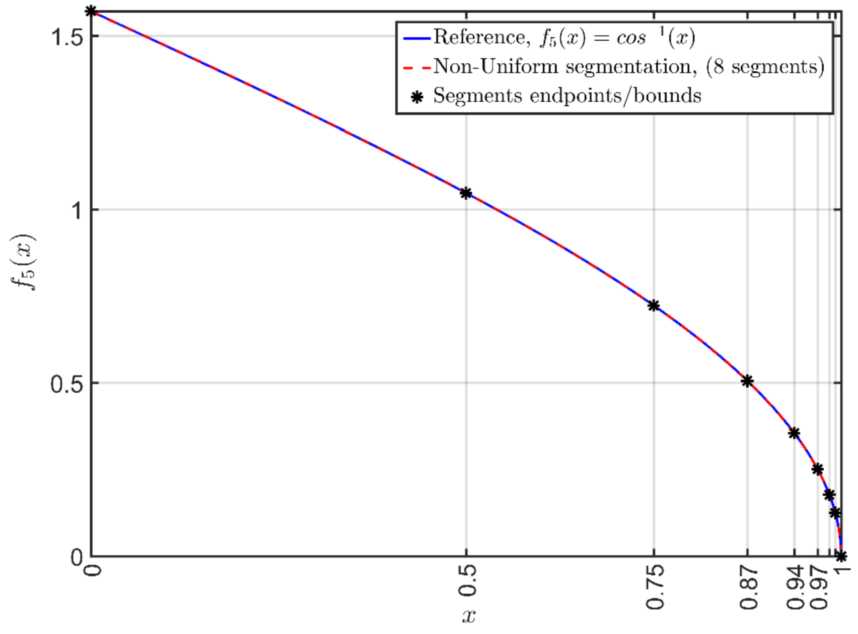


Fig. 4-2: Segmentation and approximation result for *f₅(x)* through the non-uniform methodology.

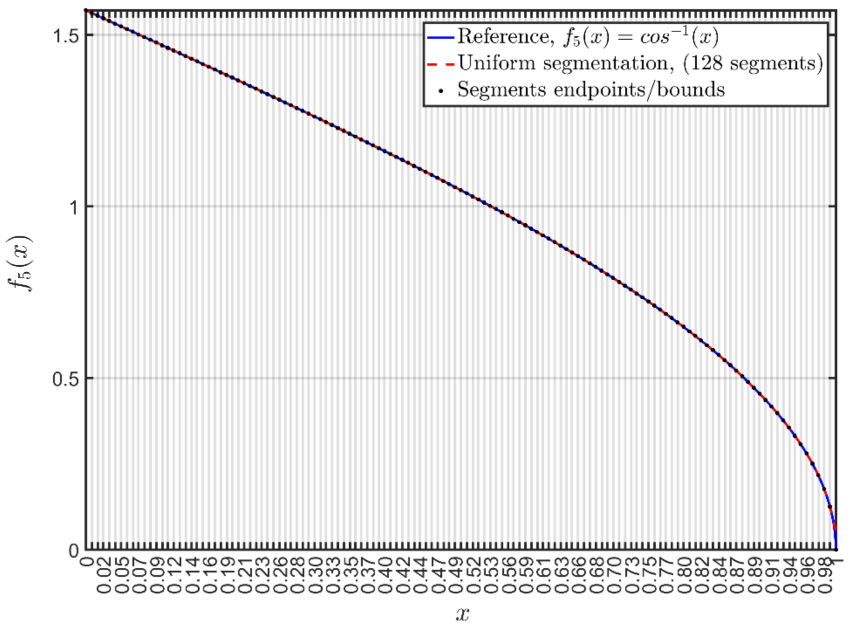


Fig. 4-3. Segmentation and approximation result for *f₅(x)* through the uniform segmentation methodology.

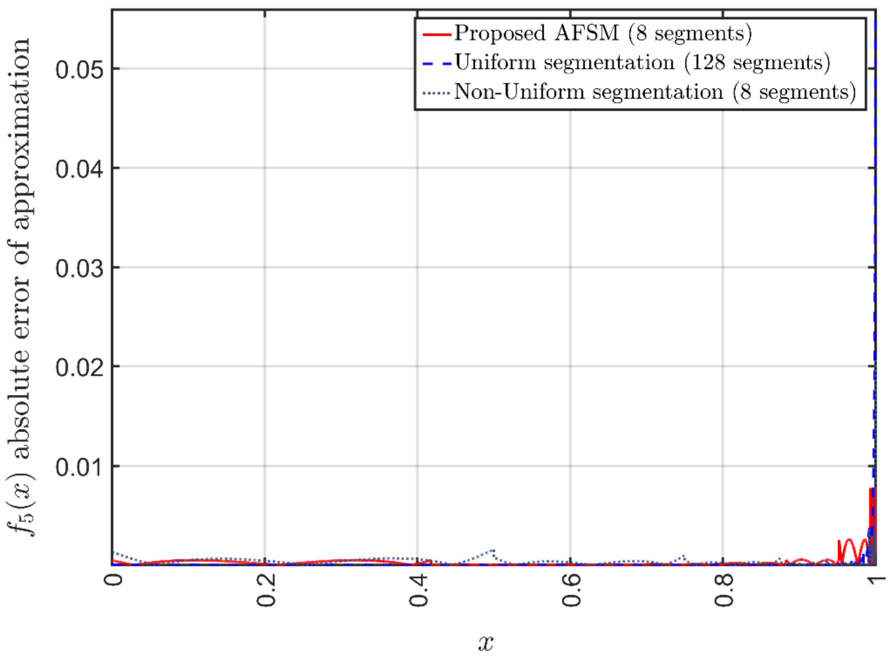


Fig. 4-4. Absolute error of approximation for *f₅(x)* from the proposed AFSM, non-uniform-by-powers-of-two, and uniform segmentation methodologies.

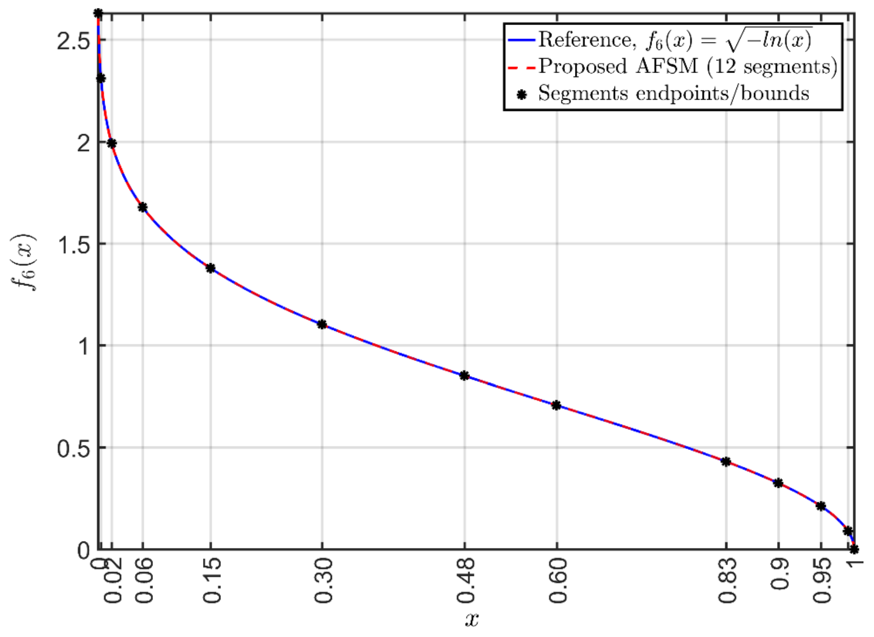


Fig. 4-5. Segmentation and approximation result for *$f_6(x)$* through the proposed AFSM.

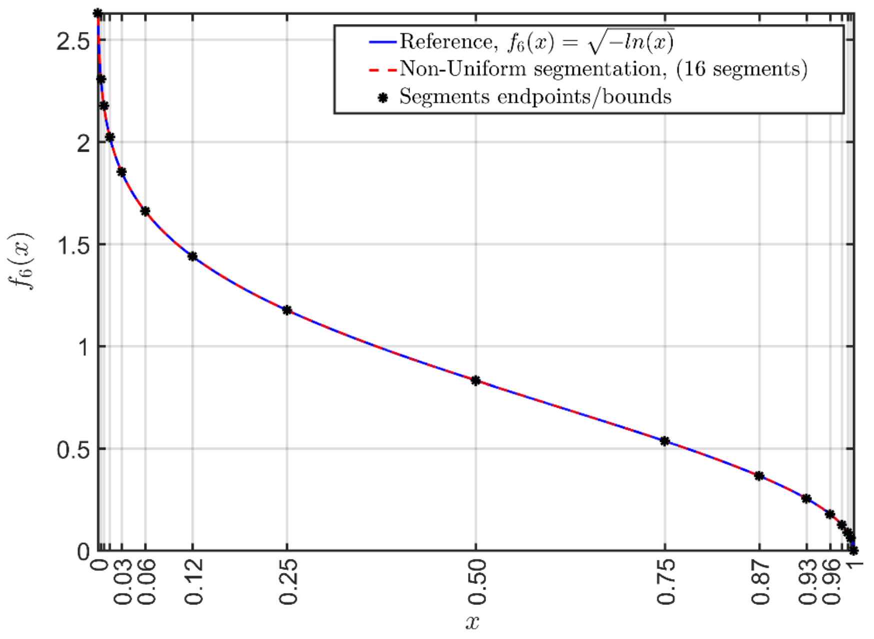


Fig. 4-6. Segmentation and approximation result for *$f_6(x)$* through the non-uniform segmentation methodology.

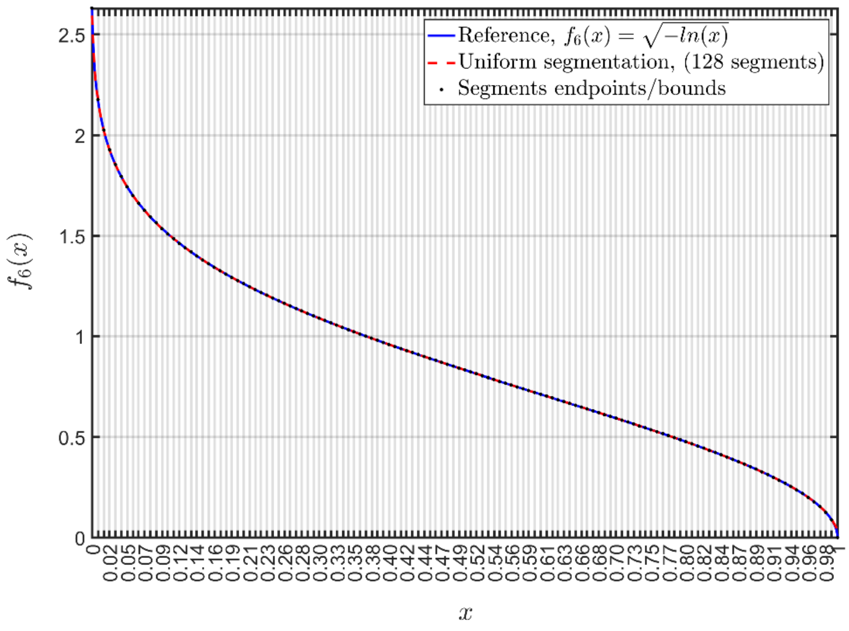


Fig. 4-7. Segmentation and approximation result for *f₆(x)* through the uniform segmentation methodology.

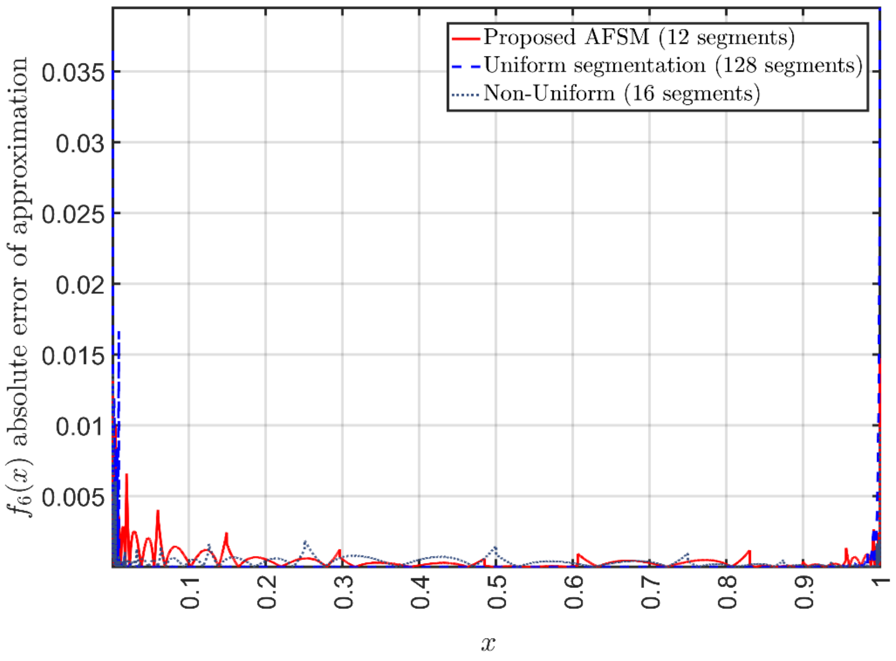


Fig. 4-8. Absolute error of approximation for *f₆(x)* from the proposed AFSM, non-uniform, and uniform segmentation methodologies.

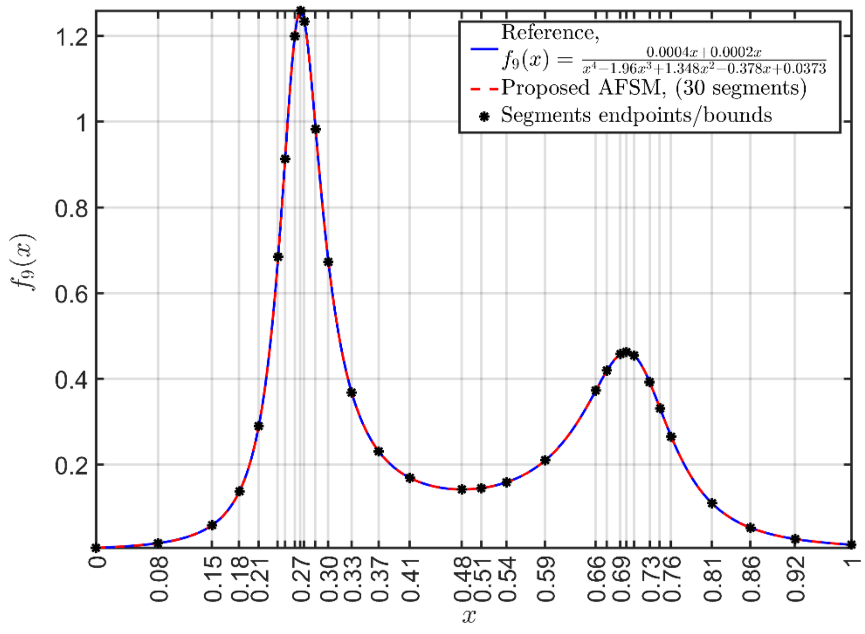


Fig. 4-9. Segmentation and approximation result for *f₉(x)* through the proposed AFSM.

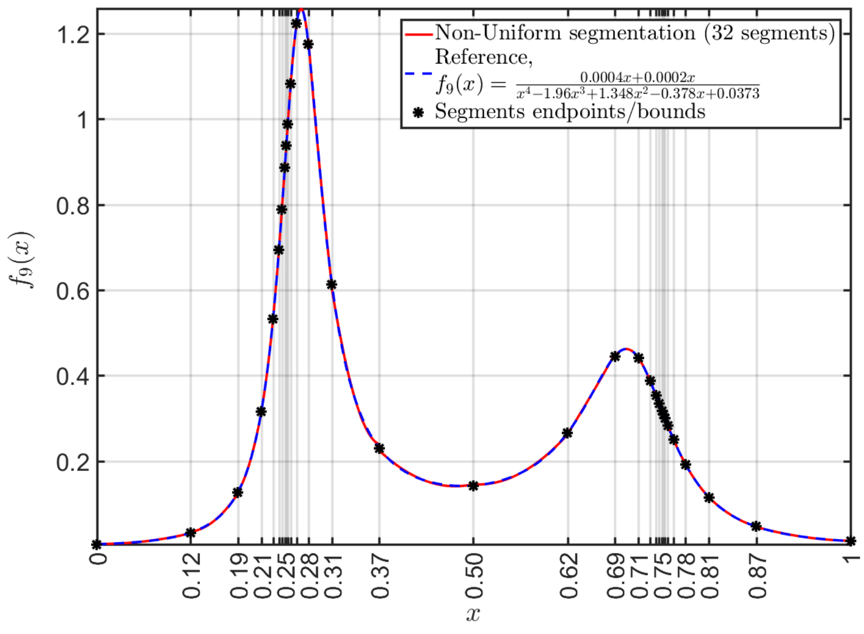


Fig. 4-10. Segmentation and approximation result for *f₉(x)* through the non-uniform segmentation methodology.

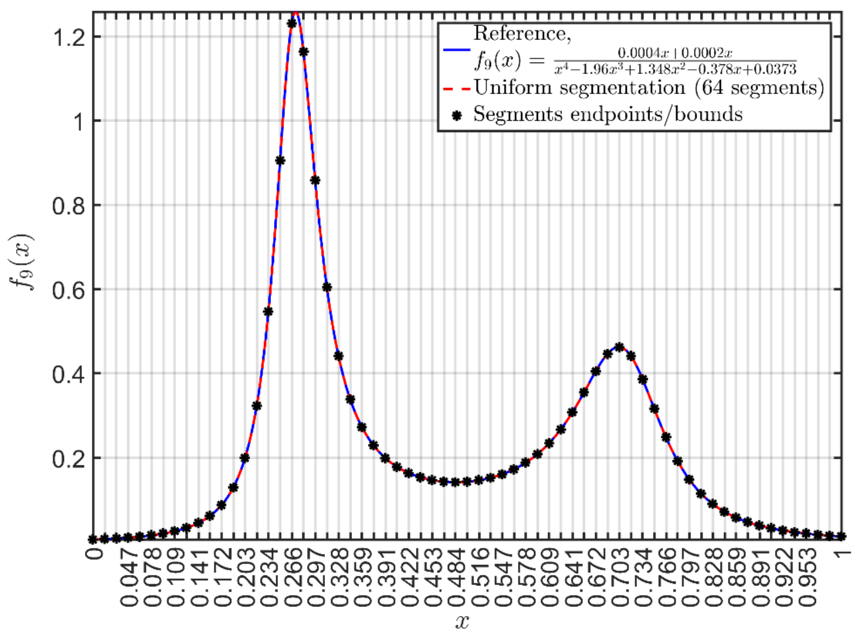


Fig. 4-11. Segmentation and approximation result for *f₉(x)* through the uniform segmentation methodology.

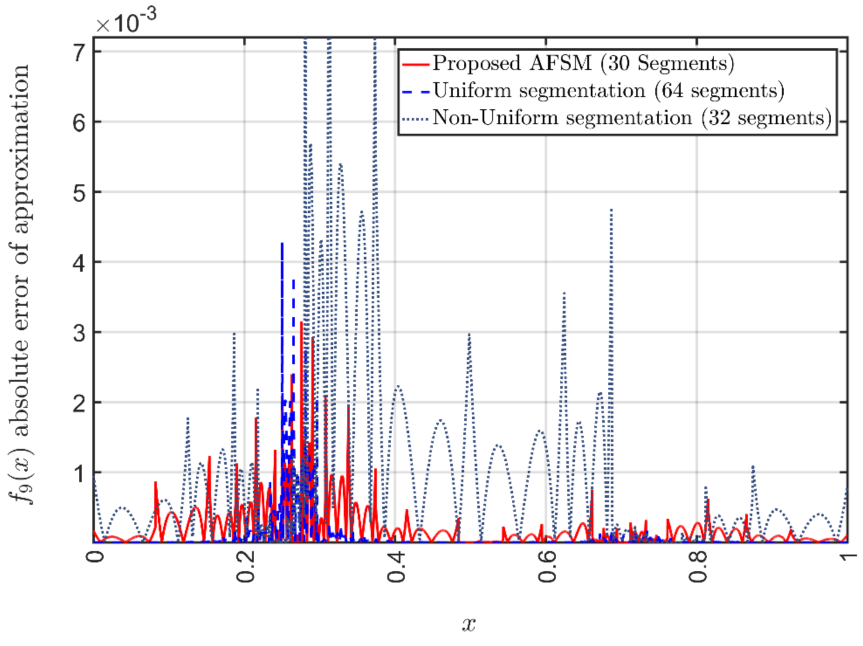


Fig. 4-12. Absolute error of approximation for *f₉(x)* from the proposed AFSM, non-uniform, and uniform segmentation methodologies.

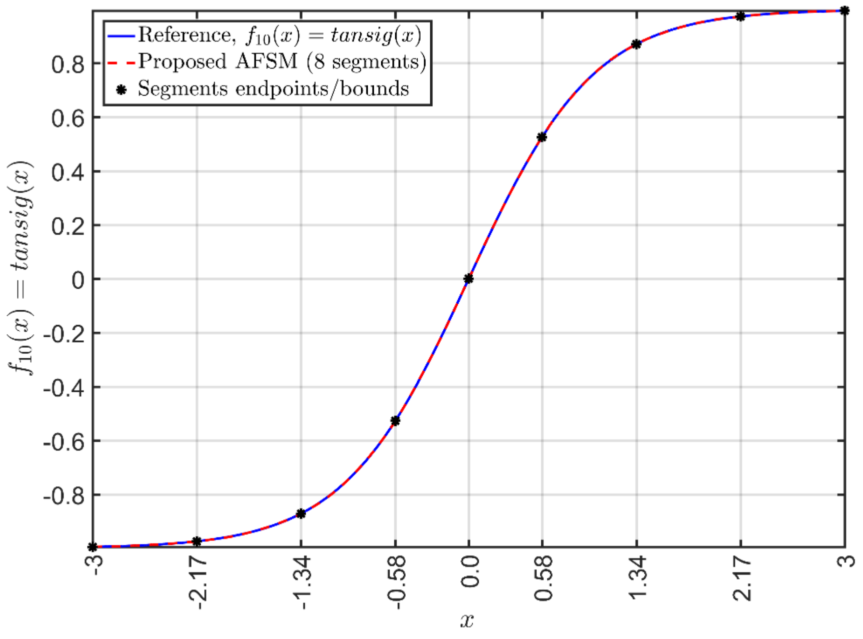


Fig. 4-13. Segmentation and approximation result for *$f_{10}(x)$* through the proposed AFSM.

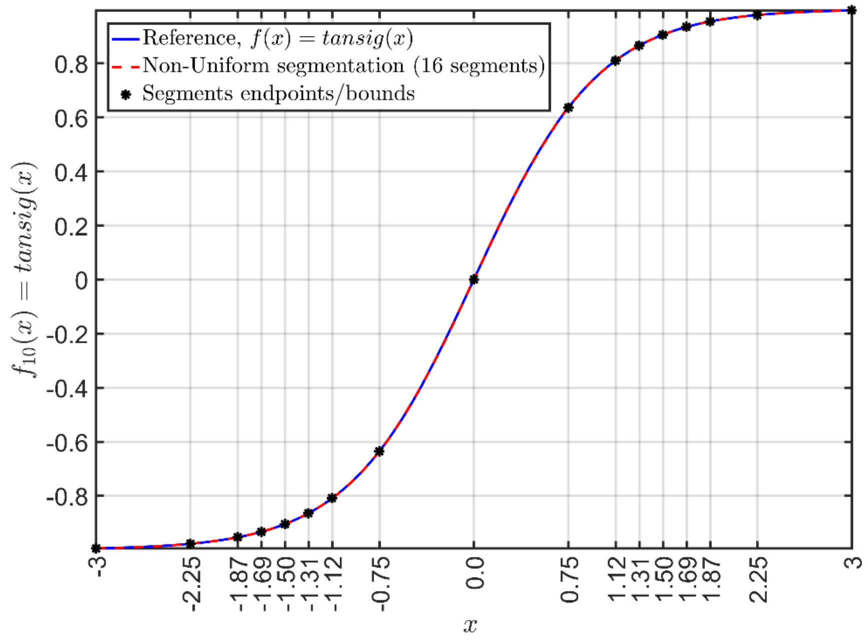


Fig. 4-14. Segmentation and approximation result for *$f_{10}(x)$* through the non-uniform segmentation methodology.

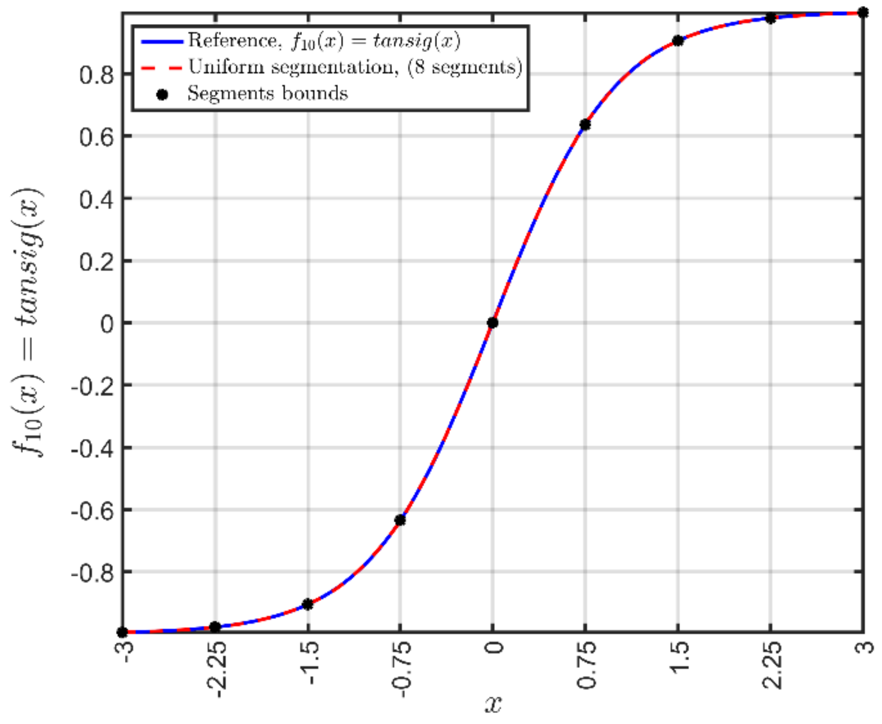


Fig. 4-15. Segmentation and approximation result for *$f_{10}(x)$* through the uniform segmentation methodology.

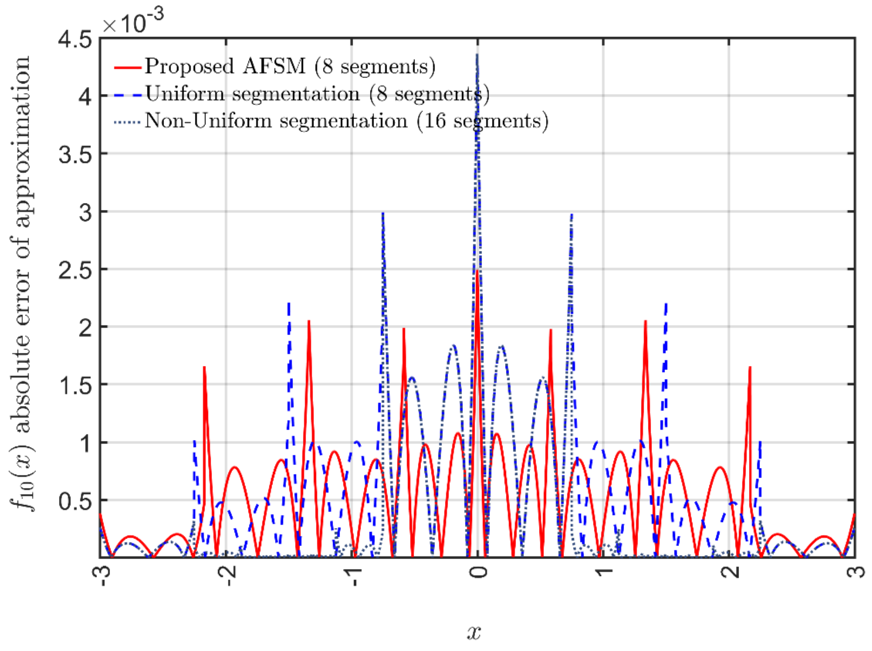


Fig. 4-16. Absolute error of approximation for *$f_{10}(x)$* from the proposed AFSM, non-uniform, and uniform segmentation methodologies.

Conclusions

This thesis presented a novel adaptive function segmentation methodology for the accurate approximation of transcendental functions through piecewise-polynomials for the efficient implementation of hardware-based functions evaluators. The proposed adaptive segmentation methodology is based on the analysis of the first and second order derivatives to perform the shape-aware segmentation of any continuous function and determine the size and location of the segments in such a way that the accuracy of the polynomial approximation is maximized. In this sense, the segmentation algorithm employs an automatic optimization algorithm that searches for the proper values of the segmentation design parameters to obtain the best balance between the number of segments and the accuracy requirements. Henceforth, the introduced algorithm can be used for implementing low area and efficient channel emulators for testing wireless communication systems.

The introduced segmentation method offers significant advantages over state-of-art segmentation methodologies such as the uniform and the non-uniform-by-powers-of-two because it can be flexibly employed for any arbitrarily-shaped continuous function, and the amount of memory required to store the coefficients of the polynomials is optimized in accord with the applications' SQNR requirements. Furthermore, the segment addressing and evaluation logic of the proposed segmentation methodology is simpler to implement than that required by the hierarchical segmentation method because it does not require the definition of addressing and evaluation hierarchies.

The presented approximation results emphasize the flexibility and accuracy offered by the proposed methodology for performing the approximation and evaluation of transcendental functions of diverse shapes. Additionally, the small hardware resourced required to make the proposed segmentation methodology an efficient and cost-effective option for implementing low area computing arithmetic blocks using PPA methodologies.

Future Work

The following are the activities planned for future work:

- The implementation of range reduction techniques to improve the approximation accuracy. However, range reduction techniques are applicable on a per function basis; therefore, the flexibility of applying the technique to any arbitrary continuous function without much intervention from the user is sacrificed.

- The implementation of a global search method such as particle swarm optimization or simulated annealing to find the global minimum amount of segments of the design space.

- The application of the polynomial coefficients into a hardware-based evaluator to obtain results of the accuracy from real hardware.

- The implementation of a case study where the proposed adaptive segmentation methodology is employed to develop a hardware channel emulator and tested to reproduce the characteristics of a real wireless transmission scenarios.

Appendix

A. PUBLISHED PAPER FOR THE IEEE MTT-S LATIN AMERICA MICROWAVE CONFERENCE

A novel function segmentation methodology for implementing affordable channel emulators

J. M. Trejo-Arellano *, J. Vázquez-Castillo †, O. Longoria-Gandara *, C. A. Gutiérrez ‡
R. Carrasco-Alvarez §, A. Castillo-Atoche ¶

*Dept. of Electronics, Systems and IT, ITESO, Guadalajara, Jal., 45604 Méx. e-mail: md687149, olongoria@iteso.mx
†Dept. of Engineering, Universidad de Quintana Roo, Chetumal, Q.R., 77019 Méx. e-mail: jvazquez@uqroo.edu.mx
‡Faculty of Science, Universidad Autónoma de San Luis Potosí, SLP, 78290 Méx. e-mail: cagutierrez@ieee.org
§Dept. of Electronic Engineering, UDG-CUCEI, Guadalajara, Jal., 44430 Méx. e-mail: r.carrasco@academicos.udg.mx
¶Dept. of Mechatronics, Universidad Autónoma de Yucatán, Mérida, Yuc., 97000 Méx. e-mail: acastill@correo.uady.mx

***Abstract*—AbstractNowadays, wireless channel emulators are designed for channel models that require the evaluation of logarithms, trigonometrics, exponentials, and other transcendental functions. These channel emulators are used for testing wireless communication standards associated to Weibull, Suzuki, Nakagami, Rayleigh and Gaussian distributions. Piecewise polynomial approximation (PPA) technique allows the evaluation of functions with an accuracy level that depends on the segmentation used, being the uniform and non-uniform by the power of two, the most commonly used. However, these segmentation techniques lack the required flexibilit to be effectively used for any function since the Signal to-Quantization-Noise-Ratio (SQNR) is highly dependent on the function at hand. A new function segmentation methodology is presented based on the firs and second derivative. Simulation results show significan SQNR advantages when the proposed methodology is compared with state-of-art segmentation techniques.**

***Index Terms*—Channel emulator, firs and second derivative, piecewise polynomial approximation.**

I. Introduction

Nowadays, the wireless communication system performance is evaluated by means of special devices such as channel emulators, which allow generating channel distortions in order to emulate the different scattering propagation conditions of a wireless communication environment.

Generally, the channel emulators implement models reported in the open literature [1], to generate variates associated to additive and multiplicative noise with different densities according to the wireless environment under test (Weibull, Suzuki, Nakagami, Rayleigh, Gaussian distributions, among others). In all cases, the channel emulators implement arithmetic computing blocks which carry out the evaluation of special functions such as logarithm, trigonometric functions (e.g. sines, cosines, etc.), exponential, and other transcendental functions. However, these special functions are not available off the shelf for hardware implementation and they need to be designed efficientl to avoid introducing additional distortions into the wireless communication channel under emulation. In this sense, several techniques such as Lookup tables (LUT), CORDIC (COordinate Rotation DIgital Computer) and Piecewise Polynomial Approximation (PAA) have been proposed for implementing complex functions evaluators.

LUT-based designs are easy to implement, however, the increment in memory is directly proportional to the accuracy required by the hardware (HW) architecture. The memory size could easily increase to several megabytes [2], depending on how high is the Signal-to-Quantization-Noise-Ratio (SQNR) requirement. CORDIC implementations have proven efficien for computing complex operations such as square roots, sine, and cosine, among other functions [2]. However, the accuracy of the CORDIC implementation heavily depends on a number of algorithm iterations. For this, the execution time requirement is an important drawback when designing high throughput systems, in which case, the accuracy might end up being sacrificed On the other hand, PPA techniques offer fl xible design trade-offs involving computation speed, memory, and accuracy. The input interval is able to be partitioned into multiple segments (uniform or non-uniform) and typically a low-degree polynomial is used to approximate each segment. Through PPA, the accuracy of the approximation can be controlled by modifying a number of segments, the segment length, the polynomial degree, or the data word length.

Currently, some works have been reported in the open literature related to the efficien implementation of channel emulators, where the hardware function evaluation is carried out following the paradigms of LUTs, CORDIC and PPA [2], [3]. As it was mentioned, PPA technique provides area and accuracy performance advantages; in addition, the hardware architecture does not change when a new function evaluation is required. However, this technique introduces approximation errors that cause distortions in the emulated channel model, when an inadequate segmentation strategy is employed. This inconvenient, results due to segmentation techniques such as uniform or non-uniform by the power of two are not sensitive to the function shape to be approximated, which causes an accuracy lost in the results and a degradation in the SQNR of the desired architecture.

In this sense, this paper presents a novel function segmentation methodology based on the firs order derivative and the second order derivative concepts. Likewise, the function segmentation is carried out according to a control parameter, provided by the firs order derivative, which define the degrees of freedom for approaching the desired function with a specifi SQNR. The proposed method is applicable to any function shape. Simulation results show the advantages of the introduced methodology considering an SQNR level and memory resources saved when it is compared with traditional segmentation techniques which are proposed in recent works.

II. Proposed function segmentation methodology

The hierarchical segmentation method (HSM) proposed in [4], is the most used function segmentation method. The main segmentation techniques embedded in HSM are the uniform and non-uniform by the power of two. Considering the function $\{y(x)|x \in x_{\mathrm{L}} \leq x \leq x_{\mathrm{H}}\}$, the uniform segmentation divides the desired function interval $X = [x_{\mathrm{L}}, x_{\mathrm{H}}]$ in equally sized segments, whereas in the non-uniform segmentation, the segment size decreases by power of two from the beginning to the end of the function interval X. Arbitrary function behaviors along the interval of interest are not correctly approximated by the non-uniform segmentation, whereas uniform segmentation uses a significan amount of segments increasing the memory resources for allocating the polynomial coefficients

Our approach is based on the firs and second derivatives of a fully define function $y(x)$ to be segmented out within the interval $X = \{x_{\mathrm{L}} \leq x \leq x_{\mathrm{H}}\}$:

$$a(x) = \frac{dy}{dx}, \tag{1}$$

$$b(x) = \frac{d^2y}{dx^2} = \frac{d}{dx}\left(\frac{dy}{dx}\right). \tag{2}$$

In this sense, the proposed methodology is carried out according to the following steps:

The firs step in the derivative segmentation algorithm is to split the overall X interval into the main sub-segments (MainSegments), afterward by identify the points where the function $y(x)$ changes direction (inflectio points, local minima or local maxima). This is at the locations where the firs or second order derivatives of $y(x)$ change of sign (i.e., $a_i(x)$ and $b_i(x)$ in (1) and (2) respectively). However, if the function does not present direction changes, then the whole X interval is taken as the single main segment. As the second step, for each of the identifie main sub-segments, an internal segmentation is performed by sweeping through and segmenting the function at the points where the absolute value of the firs order derivative has met or exceeded a given user-define threshold ($\Delta_{a_j} \geq a_{\mathrm{Th}}$). This threshold is relative to the derivative of the previous segment ($a_{\mathrm{LastEndPoint}}$). Thus, the segment length is indirectly controlled through the firs order derivative, giving shorter segments for the regions with greater derivative magnitude. The third step is to determine the coefficient of the 2-degree polynomials that best fi each of the segments. In the present work, the least square approximation algorithm is used to obtain the polynomial coefficient using

Algorithm 1 Introduced segmentation methodology.

```
1: Parameter definitions $a_{\mathrm{Th}}$, $Seg_{\mathrm{ML}}$ $SQNR_{\mathrm{level}}$, $W_{\mathrm{wl}}$, $p_{\mathrm{degree}}$.
2: for loop $i = 1 : length(X)$ do
3:     To compute $a_i = \frac{dy_i}{dx}$, $b_i = \frac{d^2y_i}{dx^2}$.
4:     To fin sign changes in $a(x)$ or $b(x)$ to defin main segments endpoints at $x_i$
5:     (MainSegments).
6: end for
7: for each MainSeg in MainSegments do
8:     for loop $j = \mathrm{MainSeg.startIndex} : \mathrm{MainSeg.endIndex}$ do
9:         To compute accumulated length AccumLength from $x_{\mathrm{lastSegEndPoint}}$
10:        to $x_j$.
11:        To calculate $\Delta_{a_j} = \frac{|a_{\mathrm{lastSegEndPoint}} - a_j|}{|a_{\mathrm{lastSegEndPoint}}|} \times 100$.
12:        if $\Delta_{a_j} \geq a_{\mathrm{Th}}$ and AccumLength $\geq Seg_{\mathrm{ML}}$ then
13:            To defin a new segment at $x_j$ ($x_{\mathrm{lastSegEndPoint}} = x_j$).
14:        end if
15:    end for
16: end for
17: To compute polynomial coeff cients of all segments.
18: To compute Fixed Point requirements of all segments coefficients
19: To compute SQNR over the whole $X$ interval.
```

the `polyfit` Matlab function, although other algorithms such as MiniMax could be used (see Algorithm 1).

Since LS method produces high magnitude coefficient for too short segments, a second user-define parameter is introduced to control the minimum length of any given segment Seg_{ML}. In this sense both criteria, the percentage of change of the firs order derivative and the minimum segment length requirement should be met to defin a new segment endpoint.

The appropriated amount of bits that should be allocated for the integer (QI) and fractional (QF) parts from the given word length (W_{wl}) is determined by.

$$QI = \lceil \log_2 (\max |p_2, p_1, p_0, X, y(x)|) \rceil, \tag{3}$$

$$QF = W_{\mathrm{wl}} - QI. \tag{4}$$

Through (3) and (4), the fi ed point analysis is carried out taking into account all coefficient (p_2, p_1, p_0) and the minimum and maximum values of x and $y(x)$.

In order to achieve the required SQNR with the minimum amount of segments for certain functions, the user might need to test different derivative thresholds and minimum segment lengths configurations This process could be automated by implementing a local or global search algorithm where the objective function is formulated in terms of the SQNR, minimum segment length, the maximum amount of segments required, polynomial degree, among others.

III. Results

The proposed algorithm has been fully implemented in Matlab for segmenting several functions in order to show the accuracy of the piecewise polynomial approximation. For each segment, 2-degree polynomials are considered; however, the algorithm could use different degree polynomials. Table I summarizes the segmentation results using the proposed algorithm for several functions that were also used as test bench in [2].

The segmentation tests were carried out assuming a word length of $W_{\mathrm{wl}} = 32$ bits; this decision is supported by the fact that modern FPGAs or SoCs systems possess such native

TABLE I: Comparative segmentation results using the proposed method for different functions with $W_{\mathrm{wl}} = 32$ bits.

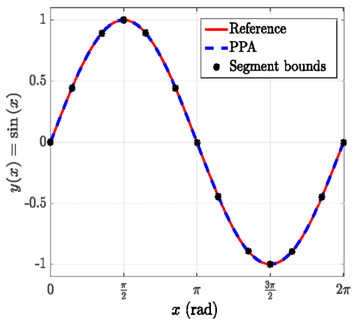
$y(x)$	a_{Th} %	Seg_{ML} %	*# Segments*	QI bits	QF bits	$SQNR$ dBs	Error	ROM Bytes
$\sqrt{x}$	28.5	10	7	13	19	65.25	.0099	84
$\frac{1}{\sqrt{x}}$	30	30	14	16	16	66.85	.0075	168
$\sin(x)$	60	25	12	5	27	66.78	.0013	144
$-\frac{x}{2}\log_2(x)$	72	20	9	6	26	64.71	.0019	108
$\cos^{-1}(x)$	4	10	9	11	21	64.52	.0338	108
$\sqrt{-\ln(x)}$	37	15	12	15	17	62.28	.0146	144
$\ln(1+x)$	90	40	2	2	30	64.49	.0008	24
$\frac{1}{1+x}$	100	30	2	2	30	62.16	.0019	24

bus widths or even greater capabilities, thus no additional resources expenditure is required. In addition, the reported a_{Th}, and Seg_{ML} values allow us achieving $SQNR$ values greater than 60 dBs with the aim of obtaining the minimum number of segments. It is highlighted that a_{Th} is represented as a percentage of the maximum magnitude value achieved for the firs order derivative. On the other hand, Seg_{ML} is expressed as a percentage of the length $x_{\mathrm{H}} - x_{\mathrm{L}}$. The ROM column represents a number of memory resources (in bytes) for allocating the polynomial coefficients which is calculated as ROM$= \frac{W_{\mathrm{wl}}}{8} \times \#Segments \times (p_{\mathrm{degree}} + 1)$, and Error column shows the accumulated absolute error between the reference functions and the approximated functions.

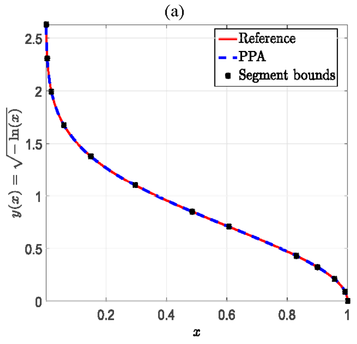
Table II shows the segmentation performance comparison between the proposed methodology and the uniform segmentation method. The segmentation was carried out in such a way to ensure reaching a $SQNR$ between 60 and 70 dBs with both methods. For comparison the $\sin(x)$ and $\sqrt{-\ln(x)}$ functions were selected. It is important to highlight that the selected functions can be used for implementing blocks in channel emulators; e.g. Rayleigh fading generators based on sum-of-cissoids, variates generation via inversive methods [2]. The proposed segmentation technique provides a significan reduction in the amount of segments needed to reach the target $SQNR$ relative to the uniform segmentation technique, which requires 38 and 128 segments for the functions $\sin(x)$ and $\sqrt{-\ln(x)}$ respectively. Such advantage can be directly translated into significan reduction of the required memory to achieve similar accuracy, given that through uniform segmentation $\sin(x)$ requires 456 bytes, while $\sqrt{-\ln(x)}$ requires 1536 bytes; this is 316.7% and 1066.6% of reduction in memory respectively.

TABLE II: Segmentation performance comparison between the proposed methodology and the uniform method.

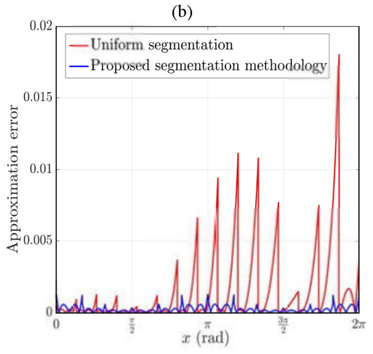
$y(x)$	Segmentation	# Segments	QI bits	QF bits	$SQNR$ dBs
$\sin(x)$	Proposed	12	5	27	66.78
	Uniform	38	5	27	66.24
$\sqrt{-\ln(x)}$	Proposed	12	15	17	62.28
	Uniform	128	18	14	60.95



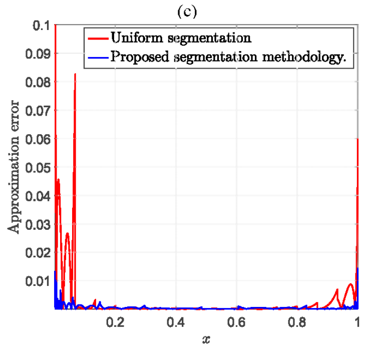
(a)



(b)



(c)



(d)

Fig. 1: Segmentation results and approximation error comparison for $y(x) = \sin(x)$ and $y(x) = \sqrt{-\ln(x)}$ functions.

Finally, in Fig. 1a and Fig. 1b, the $\sin(x)$ and $\sqrt{-\ln(x)}$ reference functions and their approximations are plotted using solid lines and dashed lines respectively. Likewise, in Fig. 1c and Fig. 1d the approximation errors using the proposed methodology and uniform segmentation method are compared for both reference functions.

IV. Conclusions

In this paper, a novel function segmentation methodology applicable to PPA techniques was presented. It is based on the firs and second derivative concepts to achieve fl xibility and segmentation performance given that the method is sensitive to the shape of the function at hand. Comparison results show that the proposed methodology can be used for evaluating arbitrary functions with excellent SQNR performances.

ACKNOWLEDGEMENT

This work was finance by the Mexican Ministry of Education (SEP-PRODEP-2014) and by the Mexican Council for Science and Technology (CONACYT) through the SEP-CONACYT Basic Research Program: Project reference #241272.

References

[1] M. Pätzold, *Mobile Radio Channels*, 2nd ed. Wiley, Nov. 2011.

[2] J. Vázquez Castillo, L. Vela-Garcia, C. A. Gutiérrez, and R. Parra-Michel, "A reconfigurabl hardware architecture for the simulation of Rayleigh fading channels under arbitrary scattering conditions," *International Journal of Electronics and Communications*, pp. 1–13, 2014.

[3] L. Pizano-Escalante, R. Parra-Michel, J. Vázquez Castillo, and O. Longoria-Gandara, "Fast bit-accurate reciprocal square root," *Microprocessors and Microsystems*, vol. 39, no. 2, pp. 74 – 82, 2015.

[4] D.-U. Lee, R. Cheung, W. Luk, and J. Villasenor, "Hierarchical segmentation for hardware function evaluation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 17, no. 1, pp. 103 –116, Jan. 2009.

B. SUBMITTED PAPER FOR THE IEEE TRANSACTIONS ON MICROWAVE THEORY AND TECHNIQUES

An adaptive function segmentation methodology based on first and second order derivatives for hardware optimization of function evaluators

Journal:	*Transactions on Microwave Theory and Techniques*
Manuscript ID	Draft
Manuscript Type:	LAMC2016 Mini-Special Issue
Date Submitted by the Author:	n/a
Complete List of Authors:	Trejo-Arellano, Juan Martin; ITESO, Jesuit University of Guadalajara, Electronics, Systems and IT Vazquez-Castillo, Javier; Universidad de Quintana Roo, Engineering Longoria-Gandara, Omar; ITESO, Jesuit University of Guadalajara, Electronics, Systems and IT Carrasco-Alvarez, Roberto; Universidad de Guadalajara, CUCEI Electronic Engineering Gutierrez, Carlos; Universidad Autonoma de San Luis Potosi, Faculty of Science Castillo-Atoche, Alejandro; Universidad Autonoma de Yucatan, Mechatronics
Keywords:	Function approximation, piecewise-polynomial, hardware optimization, segmentation, hardware evaluation, mathematical functions, surrogate modeling, wireless channel emulator

An adaptive function segmentation methodology based on first and second order derivatives for hardware optimization of function evaluators

J. M. Trejo-Arellano, J. Vázquez-Castillo, O. Longoria-Gandara, R. Carrasco-Alvarez, C. A. Gutiérrez, A. Castillo-Atoche

***Abstract*—The evaluation of mathematical functions is fundamental in surrogate models such as wireless channel emulators and other signal processing applications. This paper presents a new adaptive function segmentation methodology for the evaluation of mathematical functions through piecewise-polynomial approximation methods. In contrast to state-of-art segmentation methodologies, which applicability is highly dependent on the function shape and require significant intervention from the user to setup the algorithm, the proposed segmentation methodology is flexible and applicable to any continuous function within an evaluation interval. Through the analysis of first and second order derivatives, the methodology becomes aware of the function shape and adapts the algorithm behavior accordingly. The proposed segmentation methodology is aimed towards hardware architectures of limited resources that resort to fixed-point numeric representation where the function evaluation unit designer should make a compromise between resources consumption and output accuracy. An optimization algorithm is implemented for searching the best segmentation parameters that maximize the outcome of the design trade-offs for a given signal-to-quantization-noise ratio specification. In comparison to state-of-the-art segmentation methodologies, the proposed segmentation methodology delivers better performance of approximation for the in-hardware evaluation of transcendental functions; through a flexible and automated process, the consumption of hardware resources is minimized.**

***Index Terms*—Function approximation, piecewise-polynomial, hardware optimization, segmentation, hardware evaluation, mathematical functions, surrogate modeling, wireless channel emulator.**

I. Introduction

Nowadays, digital signal processing algorithms use high complexity blocks, which are associated with the evaluation of transcendental functions. In wireless communication channel modeling, the channel emulation can be considered as a surrogate model of the wireless channel and is carried out using various models such as ray tracing [1], [2], sum-of-cissoids (complex exponentials) [3], [4], and others. Channel models based on sum-of-cissoids, the accuracy of evaluation of the $\sin(\cdot)$ and $\cos(\cdot)$ functions within the models is a primary concern. As an example, in Weibull fading channel emulators, which are widely used for modeling vehicle-vehicle (V2V) channels [5], the hardware implementation is significantly complex due to the evaluation of $\ln(\cdot)$, $\sqrt{\cdot}$, $1/x$, and $\exp(\cdot)$ functions [6], [7]. Likewise, the efficient hardware implementation of algorithms based on algebraic matrix operations such as QR decomposition (QRD), commonly used for matrix inversion, is highly sensitive to the accuracy of evaluation of the function $\sqrt{\cdot}$ and $1/\sqrt{\cdot}$ [8]. Additionally, function evaluation is implemented for dealing with some high-complexity blocks when communication systems are developed (e.g. predistorters implemented in [9], [10], [11]), as well as for hardware accelerator blocks in general-purpose computing and graphic processor unit (GPU) applications [12].

Currently, there are several methods for the evaluation of transcendental functions. Although some of them offer certain advantages, they are also subject to disadvantages that make them unsuitable for applications that require high accuracy and substantial computing throughput. Iterative methods such as CORDIC (COordinate Rotation DIgital Computer) allow the efficient evaluation of transcendental functions [13], [14], [15], [16]. However, the output accuracy is highly dependent on the number of iteration that the algorithm is executed and represents a significant drawback that limits the development of hardware architectures for real-time computing applications. An alternative methodology for evaluating transcendental functions is via look-up tables (LUT) [7], [17]; this is arguably the simplest and easiest way to implement function evaluation blocks; however, the memory size needed for allocating the function values increases along with the output accuracy requirement.

On the other hand, piecewise-polynomial approximation (PPA) is an alternative method for evaluating transcendental functions. It offers flexible design trade-offs between computing speed, area, output accuracy, and hardware architecture utilization because the design of the polynomials evaluator does not change across functions. Approximating a functions using PPA methods requires the input evaluation interval to be partitioned into multiple segments. Each of these segments is approximated using a low-degree polynomial, which is addressed through the hardware polynomial evaluator according to the input value to the function. In this sense, the accuracy achieved using the PPA approximation methodology signif-

J. M. Trejo-Arellano and O. Longoria-Gandara are whith Dept. of Electronics, Systems and IT, ITESO, Guadalajara, Jal., 45604 Méx. e-mail: md687149, olongoria@iteso.mx

J. Vázquez-Castillo is with Dept. of Engineering, Universidad de Quintana Roo, Chetumal, Q.R., 77019 Méx. e-mail: jvazquez@uqroo.edu.mx

R. Carrasco-Alvarez is with Dept. of Electronic Engineering, UDG-CUCEI, Guadalajara, Jal., 44430 Méx. e-mail: r.carrasco@academicos.udg.mx

C. A. Gutiérrez is with Faculty of Science, Universidad Autónoma de San Luis Potosí, SLP, 78290 Méx. e-mail: cagutierrez@ieee.org

A. Castillo-Atoche is with Dept. of Mechatronics, Universidad Autónoma de Yucatán, Mérida, Yuc., 97000 Méx. e-mail: acastill@correo.uady.mx

icantly depends on the segmentation methodology utilized; i.e., sizable approximation errors might be introduced when an inadequate segmentation strategy is employed, resulting in reduced signal-to-quantization-noise ratio (SQNR) performance of the function evaluation block.

Today, the most popular segmentation methodology for PPA is called hierarchical segmentation method (HSM) [18], which combines the more basic segmentation methodologies known as uniform and non-uniform by the power of two. In principle, any function could be segmented out through all these methodologies; however, the downside is that these methods are not sensitive to the shape of the function, therefore, causing substantial accuracy loss and SQNR degradation of the desired hardware architecture. In this sense, this paper presents a new segmentation methodology for arbitrary transcendental functions, which addresses the segmentation process as a constraint optimization problem where the goal is to determine the minimum number of segments according to design objectives such as SQNR and hardware area. The latter is achieved through an automated function shape analysis using the first and second order derivatives within the given evaluation interval. This paper is an expanded version of the IEEE MTT-S Latin America Microwave Conference, Dec. 12-14, 2016, Puerto Vallarta, Mexico. The extended version of this work adds an automatic search algorithm for the optimization of the segmentation strategy and minimization of hardware resources. Also, a cost function is defined in terms of the segmentation design variables to assess the compliance of the segmentation strategy with respect to the SQNR requirements. Furthermore, a broader set of test functions is presented to emphasize the benefits regarding accuracy and reduced hardware resources delivered by the proposed function segmentation methodology.

The main contributions of this paper are summarized as follows:

- A new adaptive function segmentation methodology (AFSM), for the evaluation of arbitrary mathematical functions via PPA.
- A shape analysis procedure for arbitrary functions based on the first and second order derivatives.
- The introduction of a cost function for finding the best segmentation scheme according to specific design objectives for hardware resources optimization.

The simulation results show that the AFSM provides better performance in processing time and higher SQNR with lower hardware resources consumption in comparison to state-of-art segmentation methodologies; therefore, the AFSM represents an excellent alternative for implementing high accuracy PPA based transcendental functions evaluators embedded in sophisticated digital signal processing algorithms.

The rest of the paper is divided as follows: state-of-art approximation methods for implementing function evaluators in hardware are described in Section II. The proposed adaptive function segmentation method for hardware resource optimization is presented in Section III. Correspondingly, the performance results of the introduced segmentation methodology are presented in Section IV. Finally, the conclusions are provided in Section V.

II. Background

The approximation accuracy to a mathematical function through piecewise-polynomial approximation (PPA) methods highly depends on the function, the system word length, the number of segments, and the segmentation methodology employed. The segmentation methodologies most commonly used for hardware-based function evaluators are the uniform, non-uniform by powers of two, and the HSM proposed in [18]. Consider a continuous function $f(x)$, with first and second order derivatives, where $x \in X$ and $X = [x_{\mathrm{L}}, x_{\mathrm{H}}]$. The uniform segmentation methodology divides the function interval X, in equally sized segments; whereas, the non-uniform by power of two segmentation methodology, decreases the size of subsequent segments within X, according to the geometric progression with a common ratio of $1/2$; the segmentation can be started either from x_{L} to x_{H} or vice-versa.

In Fig. 1a and Fig. 1b, the basic segmentation methodologies show bad performance when dealing with functions that present non-monotonic curvature features. For example, the uniform segmentation methodology is only suitable for functions that present a mostly constant or slightly changing curvature within the evaluation interval. Otherwise, if the function exhibits both fast-changing and slow-changing curvature features, an excessive amount of small segments are also created around the regions of slow-changing curvature. The reason of this is that the high density of segments that is needed to appropriately approximate the fast-changing curvature features is kept uniform along the whole evaluation interval. On the other hand, the non-uniform by power of two segmentation methodology is only useful for functions that present a curvature that either increases or decreases in the same direction. As a result, the direction in which the segments decrease in size is of utmost importance because to appropriately approximate the function, the density of segments should increase as the functions' curvature increases.

The HSM is a hybrid segmentation methodology that employs both uniform and non-uniform by power of two segmentation methodologies to improve the approximation accuracy to functions with non-monotonic curvature behaviors; however, the control logic required for addressing the hierarchy of segments is it too complex and requires a significant amount of hardware resources.

To employ the previously discussed segmentation methodologies, the user should properly select the segmentation strategy (or a combination of these), and the minimum numbers of segments based on the shape of the function at hand. In many cases, this is an iterative trial and error process carried out by the user until the SQNR requirement (accuracy) is satisfied. Employing the inappropriate segmentation strategy results in a suboptimal trade-off between hardware resource consumption and SQNR degradation. In contrast, the proposed AFSM, through the analysis of the functions first and second order derivatives, tackles these issues given that the algorithm automatically adapts the segmentation strategy and the density of segments to the shape of the function at hand.

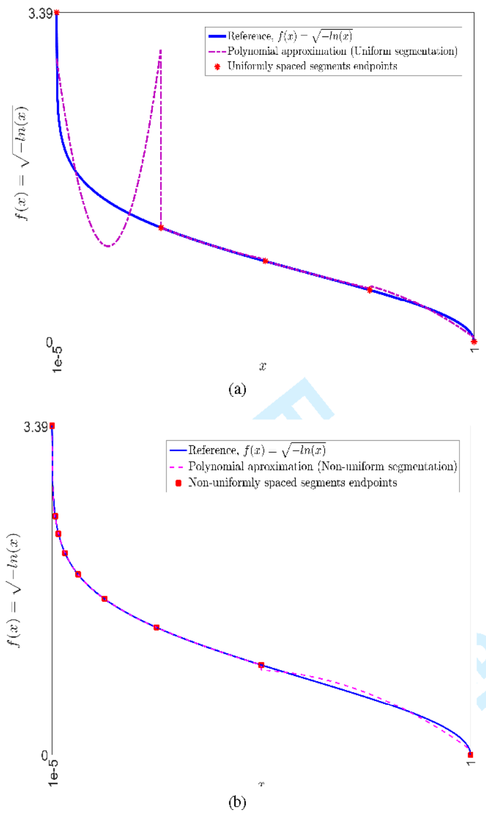


Fig. 1: (a) Poor approximation accuracy to $f(x) = \sqrt{-\ln(x)})$ when it is segmented out through the uniform segmentation methodology. (b) Insufficient approximation accuracy to $f(x) = \sqrt{-\ln(x)})$ when it is segmented out using the non-uniform by the power of two segmentation methodology.

III. Proposed Adaptive Function Segmentation for Hardware Resource Optimization

The implementation of an algorithm that automatically adapts the segmentation strategy requires precise knowledge about the shape of the function under analysis and how fast it evolves within the evaluation interval. A convenient way to get such an insight is through the implementation of an exploratory algorithm that analyzes the first and second order derivatives of the function and identifies the points in X where to split the function into segments. In this sense, the density of segments along X is automatically balanced according to the progression of the curvature shape; consequently, the algorithm automatically allocates a greater amount of segments around the regions that present a more pronounced curvature.

It is important to mention that the calculations carried out by the algorithm are solved numerically; therefore, the following sections utilize a discrete nomenclature for referring to the equations, functions, and procedures used to describe the proposed methodology.

A. Function shape analysis through fist and second order derivatives

The shape of $f(x)$ and its curvature speed of change are analyzed through the first and second order derivatives in a simple but yet powerful manner. To simplify the segmentation process and to achieve improved approximation accuracy, the first step is to perform a coarse segmentation by splitting the evaluation interval X at the critical points where the function presents a local minimum, a local maximum or an inflection point; the segments defined through this segmentation stage are called main segments.

The computation of the first and second order derivative is performed numerically through (1) and (2). Therefore, the interval X is quantized into N points addressed as x_i, where $1 \leq i \leq N$.

$$g(x_i) = \left.\frac{df(x)}{dx}\right|_{x=x_i} \approx \frac{f(x_{i+1}) - f(x_{i-1})}{2\Delta x} \tag{1}$$

$$h(x_i) = \left.\frac{d^2f(x)}{dx^2}\right|_{x=x_i} \approx \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{\Delta x^2} \tag{2}$$

where $\Delta x = |x_{i+1} - x_i| \ \forall \ i$.

The local minimum, maximum or inflection points are found at the x_i where $g(x)$ or $h(x)$ change sign, i.e., $sign(g(x_{i+1})) \neq sign(g(x_i))$ OR $sign(h(x_{i+1})) \neq sign(h(x_i))$. Thus, the set of endpoints S encompass the points x_L, x_H and at any other x_i identified through the coarse segmentation process. However, if no critical points are identified, then the entire interval delimited by the segment endpoints x_L and x_H is passed to the second step for segmentation tuning.

To exemplify the previous point, let us think on $f(x) = \sin(x)$ in Fig. 2a, which is to be segmented out within an interval that stretches along a full cycle. In this sense, the limiting points x_L and x_H of the interval are called the evaluation interval endpoints (circle marks), which are automatically created by the segmentation algorithm. The square marks in Fig. 2a, at $\pi/2$, π and $3\pi/2$ correspond to a local maximum, an inflection point, and a local minimum in $f(x)$. These locations are identified during the coarse segmentation stage by the sign changes in either $g(x)$ or $h(x)$ and represent the main segment's endpoints in $f(x)$ where its curvature changes direction.

The second step, as depicted in Fig. 2b, has the purpose of further splitting the previously defined main segments to achieve the SQNR requirement. This fine tuning segmentation process defines internal endpoints inside a coarse segment, which is bounded by the consecutive coarse endpoints $[\,x_s, x_{s+1}\,)$. A new internal endpoint is defined at an x_i where the relative change of value on the first order derivative between the previously defined endpoint at x_S and the nearest subsequent point $x_i \ \forall \ x_i < x_{s+1}$ exceeds a given γ threshold. The next expression synthesizes the previous description.

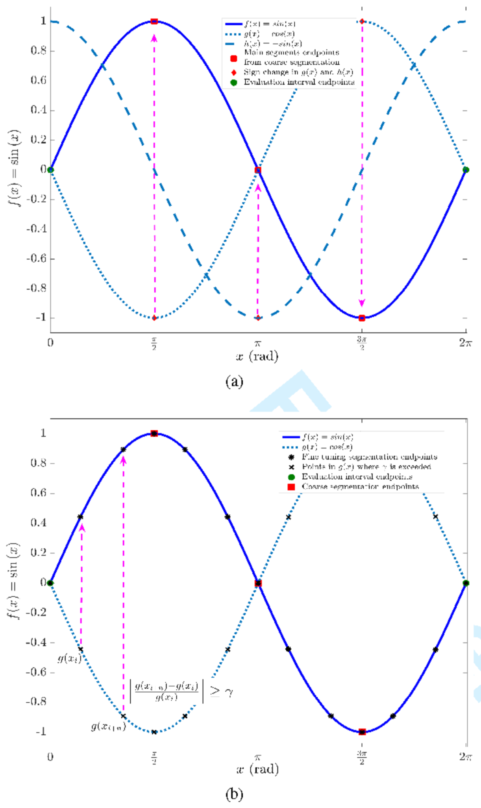


Fig. 2: (a) Coarse segmentation on $f(x) = \sin(x)$ at the points (square marks) of local maximum, inflection point, and local minimum. (b) Fine segmentation (asterisk marks) within the main segments; it is controlled through first order derivative threshold (γ) parameter.

$$\left|\frac{g(x_i) - g(x_s)}{g(x_s)}\right| \geq \gamma. \tag{3}$$

If the condition in (3) is satisfied, x_i defines a new segment endpoint in the set S and the search for the next internal endpoint starts over from $x_s = x_i$ to x_{s+1}.

B. Chordal segment length tuning

In addition to the shape analysis based on the γ threshold, the proposed algorithm also implements a minimum chordal length control that serves as a design knob for the optimization process through the κ threshold, where $\kappa > 0$.

The κ threshold serves two purposes; the first one is to achieve a better balance in the density of segments allocated when dealing with functions that present both regions of pronounced curvature as well as regions of subtle curvature. In this sense, it is possible to avoid having an excessive amount

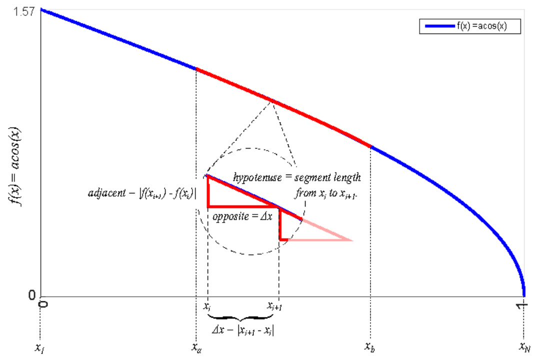

f(x) = acos(x)
$f(x) = acos(x)$
1.57
0
hypotenuse = segment length from x_i to x_{i+1}
adjacent = $|f(x_{i+1}) - f(x_i)|$
opposite = Δx
$\Delta x = |x_{i+1} - x_i|$
x_i
x_{i+1}
x_a
x_b
x_N

Fig. 3: Chordal segment length approximation through the accumulation of small triangle's hypotenuse length that fit within the interval $[x_a,\ x_b]$. x_N is the N^{th} abscissa value from the quantized interval X into N elements.

of tightly spaced segments around areas with pronounced curvature when γ is too small as a result of a poor user's selection of the seed value or because the optimization process itself has taken γ towards the design space of small values.

The second purpose of the κ threshold is to prevent having too small segments that would cause the PPA algorithm to become unstable and fail in finding a suitable set of coefficients. This failure manifests itself when the integer part of the generated coefficients is too big that its fixed point representation requires most available bits from the word length. A consequence of this is a severe loss of accuracy given that only a few bits remain for the fractional part of the coefficients.

The minimum chordal segment length threshold κ, is defined as a percentage of the total chordal length of the function within the evaluation interval. Consequently, to define a new segment endpoint at a given x_i, it is required to meet both γ and κ threshold.

The chordal length of a function within the interval limited by the points x_a and x_b is approximated by summing up the length of the hypotenuse of the many small triangles that fit such interval (see Fig. 3). The length of the triangles hypotenuse is computed using the Pythagoras theorem as:

$$length\,[x_a,\ x_b] = \sum_{i=a}^{b} \sqrt{(\Delta x)^2 + \left(f(x_{i+1}) - f(x_i)\right)^2} \tag{4}$$

where the length of the triangles opposite and adjacent sides is defined as $|x_{i+1} - x_i|$ and $|f(x_{i+1}) - f(x_i)|$, respectively.

C. Polynomials coefficient generation

After each iteration of the AFSM splitting the function interval X into a set of J segments such that $X = \bigcup_{j=1}^{J}[x_{s_j}, x_{s_{j+1}}]$, where $x_{s_1} < \ldots < x_{s_j} < \ldots < x_{s_{(J+1)}}$ are the endpoints computed according to the γ and κ thresholds, the m^{th} order of polynomial coefficients that best fit each segment are computed. The polynomials employed to approximate the function segments can be of any order $m \geq 1$ for

$m \in \mathbb{Z}$; however, it is advised to use low-even-order polynomials given that in this work, through the coarse segmentation step, we are ensuring that the curvature of the function evolves monotonically within each segment.

For the proposed AFSM, two polynomial approximation methods were tested for the computation of the best fit polynomials coefficients; the Polynomial Least Square Approximation method (LSPA) [19, p. 28], and the MiniMax Polynomials Approximation method (MMPA) [19, p. 32], which is based on the Remez algorithm [20].

Each of the tested methods treats the approximation error differently, thus, providing different levels of SQNR and accuracy from the polynomial-based approximation function $\hat{f}_j(x_i)|_{\mathbf{p}_j}$, where j represents the segment number and $\mathbf{p}_j = [p_{j_1}, p_{j_2}, ..., p_{j_{m+1}}]$ are the polynomial coefficients that correspond to the j^{th} segment. The objective of the LSPA is to find the $m+1$ polynomial coefficients for each segment that minimize the sum of the squared residual between the original function and the approximating polynomial within the segment delimited by $[x_{s_j},\ x_{s_{j+1}})$. Consequently, the squared residuals of the j^{th} segment are minimum when the following condition is satisfied:

$$\frac{\partial(R_j)}{\partial \mathbf{p}_j} = 0, \quad for\ j = 1, ..., J \tag{5}$$

where,

$$R_j \equiv \sum_{i=s_j}^{s_j+1} \left[f(x_i) - \hat{f}_j(x_i)|_{\mathbf{p}_j} \right]^2. \tag{6}$$

The error treatment strategy of the LSPA algorithm, provides direct benefit to the improvement of the SQNR because it explicitly minimizes the sum of squared residuals expression, which in fact represent the quantization noise energy, as follows:

$$E[R] = \sum_{j=1}^{J} R_j. \tag{7}$$

On the other hand, the objective of the MMPA algorithm is to minimize the maximum error or discrepancy between the approximation response $\hat{f}_j(x_i)|_{\mathbf{p}_j}$ and the original function $f(x_i)$. In general, the MiniMax algorithm yields a smaller error of approximation although the SQNR achieved is not assured to be lower than that obtained through the LSPA Method. Thus, the coefficients $\mathbf{p}_j$, are calculated as follows:

$$\mathbf{p}_j = \arg\min_{\mathbf{p}_j} \left\{ \|\mathbf{e}_j(\mathbf{p}_j)\|_\infty \right\}, \quad \text{for } j = 1, ..., J \tag{8}$$

where,

$$\mathbf{e}_j(\mathbf{p}_j) = [e_{s_j}, ..., e_i, ..., e_{s_{j+1}}] \tag{9}$$

$$e_i = \left| f(x_i) - \hat{f}_j(x_i)|_{\mathbf{p}_j} \right|, \quad \text{for} \quad s_j \le i < s_{j+1} \tag{10}$$

D. Fixed-point and SQNR analysis

For this work, the SQNR is the metric employed for measuring the accuracy of the approximation to a reference function through a set of fixed-point low-degree polynomials. The SQNR is an intuitive and widely used metric of the ratio between the power of the signal of interest and the power of the quantization noise; in other words, how well approximated is an analog signal through a digital fixed-point representation given the finite number of bits of the system word length. Therefore, the SQNR in decibels could be calculated as:

$$SQNR_{dB} = 10\log_{10}\left(\frac{\sum_{i=1}^{N} f(x_i)^2}{\sum_{j=1}^{J}\sum_{i=s_j}^{s_j+1} [f(x_i) - Q(\hat{f}_j(x_i)|_{\mathbf{p}_j})]^2} \right) \tag{11}$$

where the term $Q(\cdot)$ is the operator that quantizes the argument using a word length of WL bits with QI bits allocated to the integer part and QF bits assigned to the fractional part, such $WL = QI + QF$. In this work, the number of bits for representing the integer part are calculated as follows:

$$QI = \left\lceil \log_2\Big(\max\left(\{|\mathbf{p}_j|, |x_i|, |f(x_i)|\} \right) + 1 \Big) \right\rceil + 1, \forall i, j \tag{12}$$

The proposed AFSM relies on an iterative optimization algorithm to determine the best segmentation approach. For each segmentation realization, once the fixed point analysis has been carried out, the achieved SQNR is computed and fed back to the optimization algorithm for the objective function to determine whether the SQNR requirement has been satisfied or further segmentation refinement is required.

E. Segmentation Optimization

The proposed AFSM implements an optimization algorithm that searches in the design space for a suitable set of γ and κ threshold values that satisfy the SQNR requirement while minimizing the required number of segments. The implemented search algorithm solves the constrained non-linear optimization problem for a target SQNR requirement, which is provided by the user according to application-specific needs. The latter is mathematically expressed as:

$$\begin{gathered} \mathbf{d}^* = \arg\min_{\mathbf{d} \in \mathbb{R}} U\left(R\left(\mathbf{d}\right)\right) \\ subject\ to \\ \mathbf{d}^{lb} \le \mathbf{d} \le \mathbf{d}^{ub} \end{gathered} \tag{13}$$

where:

- $\mathbf{d} \in \mathbb{R}^2$, $\mathbf{d} = [\gamma, \kappa]$; is the vector of design variables subject to optimization.
- $\mathbf{d}^{lb}, \mathbf{d}^{ub} \in \mathbb{R}^n$; are the upper and lower design-feasibility restrictions for the design variables.
- $R(\mathbf{d}) \in \mathbb{R}^n \to \mathbb{R}$; is the function that performs the segmentation process according to the input design variables in $\mathbf{d}$. The function returns the SQNR value.

- $U : \mathbb{R} \to \mathbb{R}$; is the objective function that computes the error between the current design SQNR and the target SQNR requirement.

The solution of the constrained non-linear optimization problem is simplified if the boxed constraints ($\mathrm{d}^{lb} \leq \mathrm{d} \leq \mathrm{d}^{ub}$) are incorporated into an unconstrained optimization problem [21, p. 428], thus:

$$\mathbf{z}^* = \arg\min_{\mathbf{z}} U\left(R(\mathbf{z})\right) \tag{14}$$

where the design variables contained in $\mathbf{z}$ come from the transformation of d using:

$$\mathrm{z_i} = arcsin\left(\sqrt{\frac{\mathrm{d_i} - \mathrm{d_i}^{lb}}{\mathrm{d_i}^{ub} - \mathrm{d_i}^{lb}}}\right). \tag{15}$$

For this particular work, the solution to the unconstrained optimization problem for $\mathbf{z}$ is done through the Nelder-Mead algorithm [22]; however, many other local or global search methods could be employed as well.

F. Segmentation Technique Implementation

The pseudocode in Algorithm 1 condenses the verbal methodology description provided in previous sections to facilitate the reproducibility of the proposed segmentation methodology. Given that the proposed segmentation methodology was implemented in MATLAB, the pseudocode employs sub-index notation to address the discrete elements of vectors and collections of objects.

The Parameter Definitions and the Parameters Initialization sections of the pseudocode, introduce and initialize the variables and constants that are used across the code to set up the algorithm functionality and to store computation results. The main body of the segmentation algorithm is showed within the $do-while$ loop (lines 17 through 64) that resembles the optimization process, which iterates until the SQNR design requirement is met or the stop conditions of the optimization algorithm are reached.

Within the first $for-loop$ construct in the pseudo-code (lines 22 through 30), the coarse segmentation is performed based on the sign changes of the first and second derivatives; the segments therein created are stored in the $mainSegmts$ collection. After this step, within the second $for-loop$ construct (lines 30 through 46), the segmentation tuning stage is performed according to the design parameters γ_{Th} and κ_{Th}. The following steps (lines 48 through 50) in the pseudocode are to compute the polynomial approximation coefficients through both LSPA and MMPA methods, the fixed point analysis, and the respective $SQNR_{LSPA}$ and $SQNR_{MMPA}$ responses. The ternary conditional construct (line 51) selects the higher SQNR response, which is provided to the objective function (line 52) to determine whether the target SQNR has been satisfied or further search should be carried out. If the SQNR requirement has not been yet satisfied, the optimization algorithm iterates until the stop conditions are met (lines 53 through 62). Finally, the optimal set of polynomial coefficients from the optimized segmentation process are stored in the hardware LUT. Further detail of the pseudocode variables and their usage is summarized in Table I .

Variable name	**Usage description**
$\mathbf{x}$	The vector of the evaluation interval X that is quantized from x_L to x_H.
h, g	The vector that stores the first and second derivatives.
Δx	The discretization resolution of the vector $\mathbf{x}$, the default is $\Delta x = \frac{\lvert x_H - x_L \rvert}{2^{10}}$.
Δg	A temporary variable used to store the first derivative delta between the previous segment and a subsequent point x_i.
x_L	The lower limit of the evaluation interval X.
x_H	The upper limit of the evaluation interval X.
$Quant_{elements}$	The number of quantization elements within the evaluation interval X.
$mainSegmts$	The collection to store the segment objects from the coarse segmentation process.
$allSegmts$	The collection to store all the segments defined after the fine segmentation process.
$SQNR_{Resps}$	The collection of the resulting SQNR responses from each segmentation realization through the optimization process.
$Coeffs_{LSPA}$	The collection of polynomial coefficients for the current segmentation realization through Least Squares PPA method.
$Coeffs_{MMPA}$	The collection of polynomial coefficients for the current segmentation realization through MiniMax PPA method.
γ_{Th}	The design parameter for optimization, first derivative threshold.
κ_{Th}	The design parameter for optimization, minimum chordal segment length threshold.
$\mathrm{SQNR_{spec}}$	The target SQNR specification.
$\mathrm{W_{Len}}$	The system word length.
m	The polynomial degree, the default is 2.
segmt	The temporary iteration control segment object.
$\mathrm{Accum_{Len}}$	The temporary variable that holds the accumulated chordal length.
i, j, k	The for-loop iteration count variables.
LSPA_SQNR	The resulting SQNR for the current segmentation iteration using the coefficients from Least Squares PPA method.
MMPA_SQNR	The resulting SQNR for the current segmentation iteration using the coefficients from MiniMax PPA method.
ε	The error to the optimization objective per the SQNR design requirements.
$\mathrm{Continue}_{Search}$	The control flag of optimization process stop condition.

TABLE I: Description of variables and constants of Algorithm 1.

IV. Results

The segmentation performance and approximation accuracy of the proposed AFSM were evaluated for the set of test bench functions listed in Table II. These functions are widely employed to construct hardware blocks with application in the fields of numerical analysis, digital signal processing, wireless channel emulation, artificial neural networks, amongst others.

For all the test bench functions, the optimization process of the segmentation algorithm was set up to maintain the output SQNR within the specified range, 60dB to 70dB. Table II summarizes the approximation results from the proposed AFSM employing both LS and MiniMax PPA methods. The columns "γ^*_{Th} (%)" and "κ^*_{Th} (%)" present the optimal design parameters (first derivative and minimum chordal length thresholds) of the segmentation algorithm that satisfy the SQRN specification. The column "$SQNR^*(dB)$" presents the achieved SQNR through the optimized design parameters in columns "γ^* (%)" and "κ^*_{Th} (%)". The column "Required Segments" shows the minimum number of segments needed to meet the SQNR specification. The columns "QI (bits)" and "QF (bits)" present the number of bits assigned to the

Algorithm 1 Adaptive Function Segmentation Technique

```
1: Parameter definitions: x, g, h, $x_L$, $x_H$, $Quant_{elements}$, $\Delta x$, $\Delta g$, $mainSegmts$, $allSegmts$, $SQNR_{responses}$, $Coeffs_{LSPA}$, $Coeffs_{MMPA}$, $\gamma_{Th}$, $\kappa_{Th}$, $\mathrm{SQNR_{spec}}$, $\mathrm{W_{Len}}$, m, segmt, $Accum_{Len}$, $LSPA_{SQNR}$, $MMPA_{SQNR}$, $\varepsilon$, $\varepsilon_{Target}$, $Continue_{Optim}$,i, j, $k$
2: Parameter initialization:
3: $x_L \leftarrow \langle User\_Input \rangle$ ,default is 0
4: $x_H \leftarrow \langle User\_Input \rangle$ ,default is 1
5: $Quant_{elements} \leftarrow \langle User\_Input \rangle$ ,default is $2^{10}$
6: $\Delta x \leftarrow \frac{|x_H - x_L|}{Quant_{elements}}$
7: x $\leftarrow$ vector($\mathrm{x_L}$ : $\Delta x$ : $\mathrm{x_H}$)
8: $\gamma_{\mathrm{Th}} \leftarrow \langle User\_Input \rangle$ , default is 50%
9: $\kappa_{\mathrm{Th}} \leftarrow \langle User\_Input \rangle$ , default is 5%
10: $\mathrm{SQNR_{spec}} \leftarrow \langle User\_Input \rangle$ , default is [60dB, 70dB]
11: $\mathrm{W_{Len}} \leftarrow \langle User\_Input \rangle$ , default is 32bits
12: m $\leftarrow \langle User\_Input \rangle$ , default is 2
13: $i \leftarrow 1$, $j \leftarrow 1$, $k \leftarrow 1$
14: $\varepsilon_{\mathrm{Target}} \leftarrow 0$
15: do
16:     To load the initial design parameters $\gamma_{\mathrm{Th}}$ and $\kappa_{\mathrm{Th}}$into optimization algorithm.
17:     mainSegmts.createNewSegment()
18:     mainSegmts(mainSegmts.count).startIndex $\leftarrow 1$
19:     for loop $i \leftarrow 1 : length(\mathbf{x})$ do
20:         To compute $\mathrm{g}_i \leftarrow \frac{df(x_i)}{dx}$ and $\mathrm{h}_i \leftarrow \frac{d^2 f(x_i)}{dx^2}$
21:         Do coarse segmentation by finding sign changes in g and h:
22:         if $(sign\ (g_i) \neq sign\ (g_{i+1})) \parallel (sign\ (h_i) \neq sign\ (h_{i+1}))$ then
23:             mainSegmts(mainSegmts.count).endIndex $\leftarrow (i-1)$
24:             mainSegmts.createNewSegment()
25:             mainSegmts(mainSegmts.count).startIndex $\leftarrow i$
26:         end if
27:     end for
28:
29:     mainSegmts(mainSegmts.count).endIndex$\leftarrow$ $MaxIndexOf(\mathbf{x})$
30:     $j \leftarrow 1$
31:     for  loop $j \leftarrow 1$ : mainSegmts.count do
32:         segmt $\leftarrow$ mainSegmts($j$)
33:         for loop $i \leftarrow$ segmt.startIndex : segmt.endIndex do
34:             $\mathrm{Accum_{Len}}$ $\leftarrow$Compute length from segmt.startIndex to $x_i$
35:             To compute first derivative delta, $\Delta \mathrm{g} \leftarrow \frac{|\mathrm{g_{segmt.startIndex}} - \mathrm{g}_i|}{|\mathrm{g_{segmt.startIndex}}|} \times 100.$
36:             if $(\Delta g \geq \gamma_{Th})$ and $(\mathrm{Accum_{Len}} \geq \kappa_{Th})$ then
37:                 To split current main segment at $x_i$:
38:                 allSegmts.createNewSegment()
39:                 allSegmts.startIndex $\leftarrow$ segmt.startIndex
40:                 allSegmts(allSegmts.count).endIndex $\leftarrow (i-1)$
41:                 segmt.startIndex $\leftarrow i$
42:             end if
43:         end for
44:     end for
45:     allSegmts(allSegmts.count).endIndex $\leftarrow$ mainSegmts(mainSegmt.count).endIndex
46:     To compute the segments coefficients: ($\mathrm{Coeffs_{LSPA}}$, $\mathrm{Coeffs_{MMPA}}$)
47:     To compute Fixed-Point analysis
48:     Compute SQNR for the $k^{th}$ optimization iteration: ($\mathrm{SQNR}_{LSPA}$, $\mathrm{SQNR}_{MMPA}$)
49:     $SQNR_{Resps(k)} \leftarrow SQNR_{LSPA} \geq SQNR_{MMPA} : SQNR_{LSPA}\ ?\ SQNR_{MMPA}$
50:     To compute error ($\varepsilon$) to optimization objective
51:     if Stop conditions have been meet?  then
52:         $Continue_{Search} \leftarrow FALSE$
53:     else
54:         if  then$\varepsilon \geq \varepsilon_{\mathrm{Target}}$then
55:             To search for alternative design parameters ($\gamma_{Th}$ and $\kappa_{Th}$)
56:             $Continue_{Search} \leftarrow TRUE$
57:         else
58:             $Continue_{Search} \leftarrow FALSE$
59:         end if
60:     end if
61:     Increment optimization iterations counter: Set$j \leftarrow j + 1$
62: while continueOptimization
63: To Store the coefficients that deliver best SQNR:
64: $Coeffs_{LUT} \leftarrow (SQNR_{LSPA} \geq SQNR_{MMPA}) : Coeffs_{LSPA}\ ?\ Coeffs_{MMPA}$
```

integer and fractional parts of the fixed point representation of the values of the polynomial coefficients, the range, and the domain of the approximated function. The maximum absolute error of approximation for each function and PPA method is presented in the "Max |Error|" column. Finally, the column "ROM (Bytes)" shows the bytes of memory required by the look-up table (LUT) with the polynomials coefficients of all the segments needed to achieve the SQNR specification for each PPA method; the memory requirements are calculated as $ROM_{Bytes} = \frac{W_{\mathrm{len}}}{8} \times \#Seg \times (\mathrm{m}+1)$.

Although the proposed AFSM can be employed to approximate transcendental functions using polynomials of any degree, to reduce the number of coefficients required for each segment, second-degree polynomials were used for both Least Squares and MiniMax methods. In this sense, the polynomial approximation tests were carried out with a uniform word length of $W_{\mathrm{Len}} = 32$ bits. This decision is supported by the fact that most modern field programmable gate arrays (FPGA) or systems on a chip (SoC) have these or even greater bus width capabilities; therefore, no additional resources expenditure is required.

It is possible to observe in Table II that when the polynomial approximation is carried out through the MiniMax method for the functions $f_1(x)$, $f_2(x)$, $f_4(x)$, and $f_5(x)$ one less segment is needed to reach the target SQNR than when the segmentation is performed through the LSPA method. Furthermore, given that MiniMax finds the polynomial coefficients that minimize the maximum error of approximation, for most of the test bench functions, the maximum absolute value achieved through the MiniMax method was smaller in comparison to that obtained through the LSPA method. However, it is possible to observe that for the functions $f_3(x)$, $f_6(x)$, $f_7(x)$, $f_8(x)$, and $f_9(x)$ the achieved SQNR though the MinMax method was slightly lower in comparison to that obtained through LSPA method. The reason of this is that the objective of the LSPA method is to find a set of polynomial coefficients for each segment that minimize the sum of the squared residual between the original function and the approximating polynomial. Consequently, the denominator of the SQNR expression in (11) that accounts for the quantization noise is minimized.

For the functions $f_5(x)$, $f_6(x)$, $f_9(x)$, and $f_{10}(x)$, Table III shows the segmentation performance comparison between the proposed AFSM versus the uniform and non-uniform by the power of two methodologies for an SQNR specification between 60 dB and 70 dB. These functions were selected for comparison because these present curvature features that are challenging to approximate trough a basic segmentation methodology alone. For example, given the specified SQNR, $f_5(x)$ can be approximated using only eight segments through both the proposed AFSM, plotted in Fig. 4a and the non-uniform by the power of two methodology, plotted in Fig. 4b. On the other hand, the uniform segmentation methodology that is plotted in Fig. 4c, does not perform satisfactorily because an excessive number of 128 segments are required in an attempt to reduce the approximation error shown in Fig. 4d, which increases as the curvature of $f_5(x)$ increases. Similarly, for the functions $f_6(x)$ and $f_9(x)$, which are plotted in Fig. 5c and Fig. 6c, the uniform segmentation methodology requires a significantly greater amount of segments compared to the proposed AFSM; for $f_6(x)$ and $f_9(x)$ the uniform segmentation methodology requires 128 and 64 segments while the proposed AFSM requires only 12 and 30 segments, respectively.

The advantages of the proposed AFSM, over the previously discussed basic segmentation methodologies, are demonstrated through the more elaborated curvature shapes of the functions $f_6(x)$, $f_9(x)$, and $f_{10}(x)$, which are plotted in Fig. 5a, Fig.

	PPA Meth.	γ^* %	κ^* %	$SQNR^*$ dB	# seg.	QI bits	QF bits	Max $\|Error\|$	ROM Bytes
$f_1(x) = \sqrt{x}$	LS	28.5	10	65.25	7	13	19	0.0099	84
	MM	41	10	62.48	6	12	20	0.0055	72
$f_2(x) = \frac{1}{\sqrt{x}}$	LS	30	30	66.85	14	16	16	0.0075	168
	MM	93.4	10	63.12	10	16	16	0.0096	120
$f_3(x) = \sin(x)$	LS	60	25	66.78	12	5	27	0.0013	144
	MM	91.8	25	64.61	12	5	27	0.0007	144
$f_4(x) = -\frac{x}{2}\log_2(x)$	LS	72	20	64.71	9	6	26	0.0019	108
	MM	95.7	20	64.36	8	7	25	0.0005	96
$f_5(x) = \cos^{-1}(x)$	LS	4	10	64.52	9	11	21	0.0338	108
	MM	10	10	64.66	8	14	18	0.0078	96
$f_6(x) = \sqrt{-\ln(x)}$	LS	37	15	62.28	12	15	17	0.0146	144
	MM	20	15	60.24	12	15	17	0.007	144
$f_7(x) = \ln(1+x)$	LS	90	40	64.49	2	2	30	0.0008	24
	MM	40	40	63.22	2	2	30	0.0005	24
$f_8(x) = \frac{1}{1+x}$	LS	100	30	62.16	2	2	30	0.0019	24
	MM	100	30	60.86	2	2	30	0.0011	24
$f_9(x) = \frac{0.0004x+0.0002x}{x^4-1.96x^3+1.348x^2-0.378x+0.0373}$									
	LS	143	8	60.52	30	11	21	0.0031	360
	MM	113	8	60.32	30	11	21	0.0020	360
$f_{10}(x) = tansig(x)$	LS	50	25	62.02	8	3	29	0.0025	96
	MM	50	25	60.80	8	3	29	0.0014	96

TABLE II: Segmentation and approximation accuracy results from the proposed AFSM for both LS and MiniMax (MM) PPA methods. To obtain these results it was used a $WL = 32$ bits and polynomials of degree, m = 2.

$6a$, and Fig. $7a$. For these test functions, the proposed AFSM meets the SQNR specification with the minimum number of segments amongst the comparing segmentation methodologies. Also, and most importantly, through the AFSM, the segmentation and approximation procedure was automatically performed and optimized according to the evolution of the curvature shape without intervention from the user.

In contrast, to apply the non-uniform methodology on these functions, the user should intervene in the definition of a segmentation hierarchy within the sub-intervals in X where

$f(x)$	Segmentation Technique	# Segments	QI bits	QF bits	$SQNR$ dB
$f_5(x) = \cos^{-1}(x)$	AFSM	8	14	18	64.66
	Uniform	128	13	19	58.64
	Non-Uniform	8	13	19	66.53
$f_6(x) = \sqrt{-\ln(x)}$	AFSM	12	15	17	62.28
	Uniform	128	18	14	60.95
	Non-Uniform	16	15	17	65.74
$f_9(x) = \frac{0.0004x+0.0002x}{x^4-1.96x^3+1.348x^2-0.378x+0.0373}$					
	AFSM	30	11	21	60.52
	Uniform	64	11	21	61.96
	Non-Uniform	32	11	21	50.51
$f_{10}(x) = tansig(x)$	AFSM	8	3	28	62.02
	Uniform	8	3	29	59.62
	Non-Uniform	16	3	29	61.19

TABLE III: Segmentation performance comparison between the proposed AFSM versus the uniform and the non-uniform by power of two methodologies

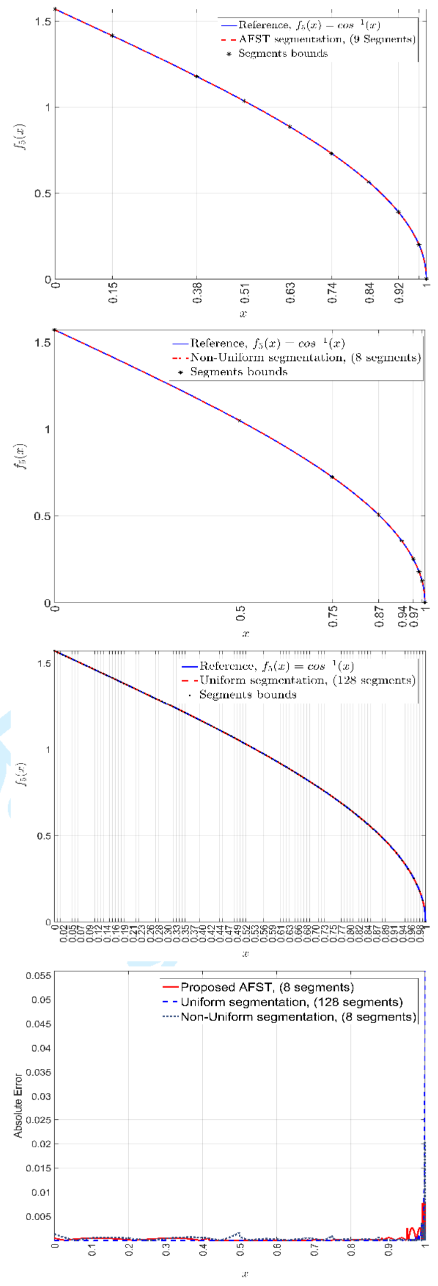
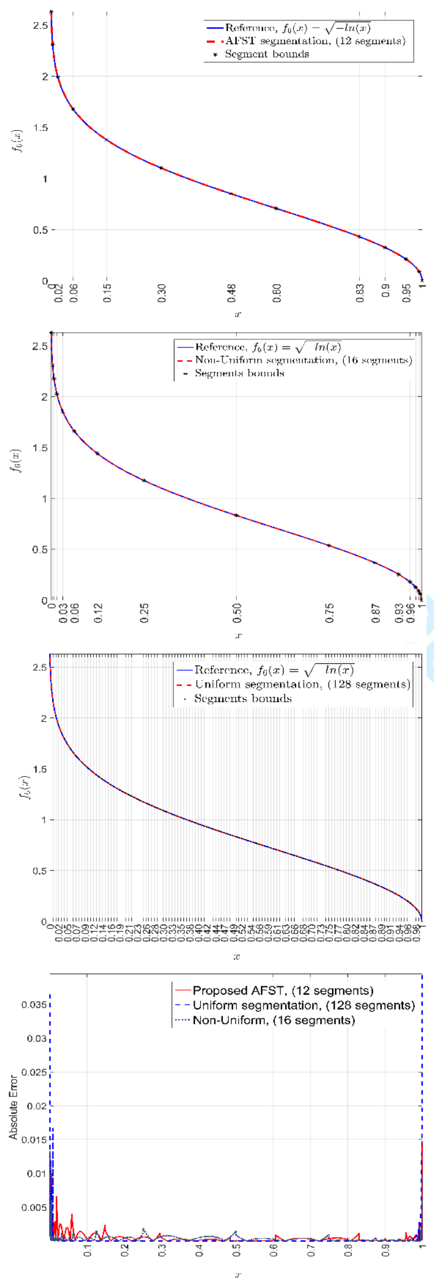
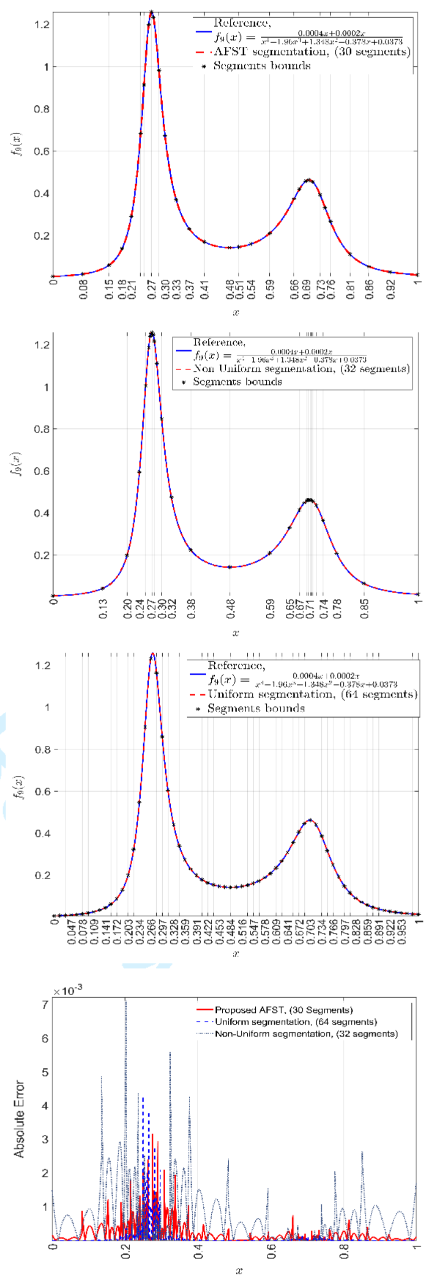


Fig. 4: Segmentation and approximation results for $f_5(x)$. (a) AFSM. (b) Uniform segmentation. (c) Non-uniform by power of two. (d) Absolute error of approximation.



(d)

Fig. 5: Segmentation and approximation results for $f_6(x)$. (a) AFSM. (b) Uniform segmentation. (c) Non-Uniform by power of two. (d) Absolute error of approximation.



(d)

Fig. 6: Segmentation and approximation results for $f_9(x)$. (a) AFSM. (b) Uniform segmentation. (c) Non-Uniform by power of two. (d) Absolute error of approximation.

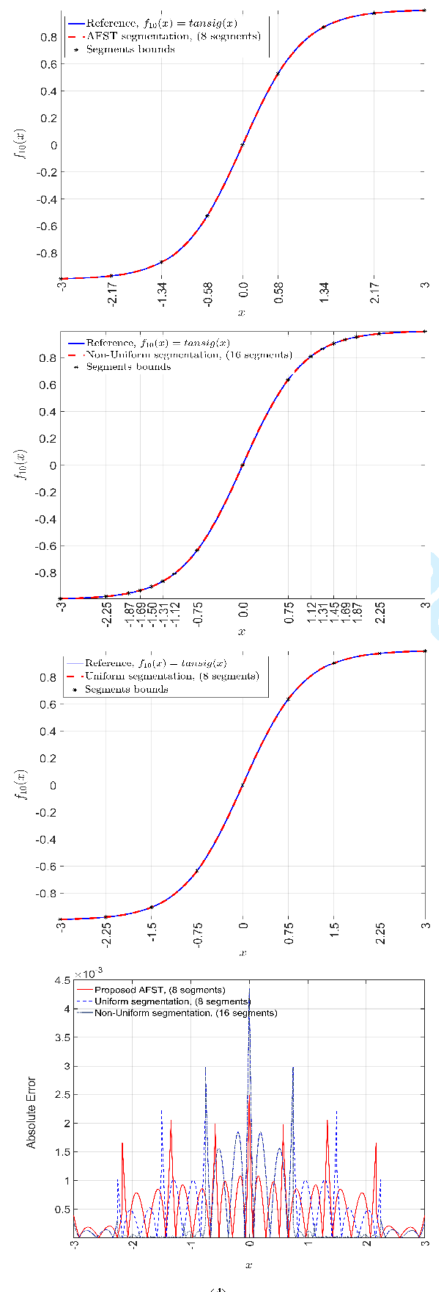


Fig. 7: Segmentation and approximation results for $f_{10}(x)$. (a) AFSM. (b) Uniform segmentation. (c) Non-Uniform by power of two. (d) Absolute error of approximation.

V. Conclusions

The efficient and accurate evaluation of transcendental functions in digital signal processors is a factor of utmost importance that drives the performance of the developed algorithms. In this sense, this paper presented a novel adaptive function segmentation methodology for the accurate approximation of transcendental functions through piecewise-polynomials for the efficient implementation of hardware-based functions evaluators. The proposed adaptive segmentation method is based on the analysis of the first and second order derivatives to perform the shape-aware segmentation of any continuous function and determine the size and location of the segments in such a way that the accuracy of the polynomial approximation is maximized. In this sense, the segmentation algorithm employs an automatic optimization algorithm that searches for the proper values of the segmentation design parameters to obtain the best balance between the number of segments and the accuracy requirements.

The introduced segmentation method offers significant advantages over state-of-art segmentation methodologies such as the uniform and the non-uniform by power of two because it can be flexibly employed for any arbitrarily-shaped continuous function, and the amount of memory required to store the coefficients of the polynomials is optimized in accord with the applications SQNR requirements. Furthermore, the segment addressing and evaluation logic of the proposed segmentation methodology is simpler to implement than that required by the hierarchical segmentation method because it does not require de definition of addressing and evaluation hierarchies.

The presented approximation results emphasize the flexibility and accuracy offered by the proposed methodology for performing the approximation and evaluation of transcendental functions of diverse shapes. Additionally, the small hardware resources required make the proposed segmentation method an efficient and cost-effective option for implementing low area computing arithmetic blocks using piecewise-polynomial approximation methodologies.

Acknowledgement

This work was supported by Consejo Nacional de Ciencia y Tecnologia (CONACYT), México, under Grant CB-241272.

References

[1] M. K. Samimi and T. S. Rappaport, “3-D millimeter-wave statistical channel model for 5G wireless system design,” *IEEE Transactions on Microwave Theory and Techniques*, vol. 64, no. 7, pp. 2207–2225, July 2016.

[2] J. He, A. A. Verstak, L. T. Watson, C. A. Stinson, N. Ramakrishnan, C. A. Shaffer, T. S. Rappaport, C. R. Anderson, K. K. Bae, J. Jiang, and W. H. Tranter, “Globally optimal transmitter placement for indoor wireless communication systems,” *IEEE Transactions on Wireless Communications*, vol. 3, no. 6, pp. 1906–1911, Nov 2004.

[3] J. Vázquez Castillo, L. Vela-Garcia, C. A. Gutiérrez, and R. Parra-Michel, “A reconfigurable hardware architecture for the simulation of Rayleigh fading channels under arbitrary scattering conditions,” *International Journal of Electronics and Communications*, pp. 1–13, 2014.

[4] P. Huang, Y. Du, and Y. Li, “Stability analysis and hardware resource optimization in channel emulator design,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 7, pp. 1089–1100, July 2016.

[5] Y. Li, B. Ai, X. Cheng, S. Lin, and Z. Zhong, “A TDL Based Non-WSSUS Vehicle-to-Vehicle Channel Model,” *International Journal of Antennas and Propagation*, vol. 2013, p. e103461, Nov. 2013.

[6] P. Huang, D. Rajan, and J. Camp, "Weibull and Suzuki fading channel generator design to reduce hardware resources," in *2013 IEEE Wireless Communications and Networking Conference (WCNC)*, Apr. 2013, pp. 3443–3448.

[7] A. Alimohammad, S. Fard, and B. Cockburn, "Hardware Implementation of Nakagami and Weibull Variate Generators," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 7, pp. 1276–1284, Jul. 2012.

[8] L. Canche Santos, A. Castillo Atoche, J. Vázquez Castillo, O. Longoria-Gándara, R. Carrasco Alvarez, and J. Ortegon Aguilar, "An improved hardware design for matrix inverse based on systolic array QR decomposition and piecewise polynomial approximation." IEEE, Dec. 2015, pp. 1–6.

[9] Y. Liu, G. Liu, and P. M. Asbeck, "High-order modulation transmission through frequency quadrupler using digital predistortion," *IEEE Transactions on Microwave Theory and Techniques*, vol. 64, no. 6, pp. 1896–1910, June 2016.

[10] Y. Ma, Y. Yamao, Y. Akaiwa, and C. Yu, "FPGA implementation of adaptive digital predistorter with fast convergence rate and low complexity for multi-channel transmitters," *IEEE Transactions on Microwave Theory and Techniques*, vol. 61, no. 11, pp. 3961–3973, Nov 2013.

[11] S. Boumaiza, J. Li, M. Jaidane-Saidane, and F. M. Ghannouchi, "Adaptive digital/RF predistortion using a nonuniform LUT indexing function with built-in dependence on the amplifier nonlinearity," *IEEE Transactions on Microwave Theory and Techniques*, vol. 52, no. 12, pp. 2670–2677, Dec 2004.

[12] S. Eldridge, F. Raudies, D. Zou, and A. Joshi, "Neural network-based accelerators for transcendental function approximation," in *Proceedings of the 24th Edition of the Great Lakes Symposium on VLSI*. New York, NY, USA: ACM, 2014, pp. 169–174.

[13] S. D. Munoz and J. Hormigo, "High-throughput fpga implementation of qr decomposition," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 62, no. 9, pp. 861–865, Sept 2015.

[14] A. Vázquez and J. D. Bruguera, "Iterative algorithm and architecture for exponential, logarithm, powering, and root extraction," *IEEE Transactions on Computers*, vol. 62, no. 9, pp. 1721–1731, Sept 2013.

[15] M. Shabany, D. Patel, and P. Gulak, "A Low-Latency Low-Power QR-Decomposition ASIC Implementation in 0.13 CMOS," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 2, pp. 327–340, Feb. 2013.

[16] V. Tiwari and N. Khare, "Hardware implementation of neural network with Sigmoidal activation functions using CORDIC," *Microprocessors and Microsystems*, vol. 39, no. 6, pp. 373–381, Aug. 2015.

[17] F. Ren and Y. Zheng, "A novel emulator for discrete-time MIMO triply selective fading channels," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 9, pp. 2542 –2551, Sep. 2010.

[18] D.-U. Lee, R. Cheung, W. Luk, and J. Villasenor, "Hierarchical segmentation for hardware function evaluation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 17, no. 1, pp. 103 –116, Jan. 2009.

[19] J.-M. Muller, *Elementary Functions: Algorithms and Implementation*, 2nd ed. Birkhäuser, Oct. 2005.

[20] E. W. Cheney, *Introduction to Approximation Theory*, 2nd ed. Amer Mathematical Society.

[21] S. S. Rao, *Engineering Optimization: Theory and Practice*, 4th ed. Wiley.

[22] J. A. Nelder and R. Mead, "A simplex method for function minimization," vol. 7, no. 4, pp. 308–313. [Online]. Available: http://comjnl.oxfordjournals.org/content/7/4/308

J. M. Trejo-Arellano was born in Atotonilco, Jalisco, Mexico, in 1986. He received a B.Sc. degree in electronics and communications from the Universidad de Guadalajara (U de G) in 2009; he is currently pursuing a Masters in Electronic Design from ITESO, The Jesuit University of Guadalajara, Mexico. From 2010 to 2016 he worked for Intel in the areas of CPU pre-silicon verification, server platform hardware design, and server platform signal integrity. His research interests include signal integrity analysis of high-speed interconnects, development and implementation of digital signal processing algorithms, design and implementation of software and hardware for microcontrollers based embedded systems, and the application of optimization and design of experiments methods for the improvement of electronics systems design.

J. Vázquez Castillo received the Ph.D. (2014) and the M.Sc. (2002) degrees in Electrical Engineering from CINVESTAV Guadalajara, Mexico and the Electronic Engineer of ITM (Mérida Institute of Technology, 2000) in Mexico. He is Professor of the University of Quintana Roo. His research work is in applications of intelligent signal processing, channel emulation, computer arithmetic, and hardware design.

O. Longoria-Gandara was born in Delicias, Mexico, in 1971. He received a B.Sc. degree in electronics and communications from the Instituto Tecnologico y de Estudios Superiores de Monterrey (ITESM-Mty.) in 1993; the M.Sc. degree in electrical engineering specialized in communications from CINVESTAV-IPN, Mexico, in 1998. From 1998 to 2006 he was with the Electrical Engineering Department (EED) of ITESM Guadalajara as full time Professor and from 2004 to 2006 he was in charge of this department. In the summer of 2001 and 2002 he received a Project Orient Learning seminar at Twente University in Netherlands which result was the publishing of a Hardware Computer Design Handbook for the ITESM. Since 2001 he has been a committee member of the General Electric Foundation scholarships for Mexico. He received the PhD degree in 2010 at CINVESTAV-IPN, Guadalajara, Mexico, where he has done research in time invariant/variant MIMO channel estimation, radio channel modeling and spacetime block codes. His research interests include MIMO channel estimation, performance and analysis of high-speed interconnect circuits, MIMO digital precoding and implementation of embedded communications algorithms using hardware description languages. Currently, he is with the Electronics, Systems and Computer Department at ITESO University, Mexico, and he is cooperating with CINVESTAV, Gdl. and Intel-Labs Guadalajara, Mexico.

R. Carrasco Alvarez received the B.Sc. degree in electronics (2004) from Instituto Tecnologico de Morelia, Mexico, the M.Sc. and Ph.D. degrees in electrical engineering, specializing in telecommunications, from CINVESTAV Guadalajara, Mexico, in 2006 and 2010 respecively. He currently is professor and researcher in CUCEI, Universidad de Guadalajara. His research interests include signal processing and digital communications.

C. A. Gutiérrez received the B.E. degree in electronics and digital communication systems from the Autonomous University of Aguascalientes, Aguascalientes, Mexico, in 2002; the Advanced Studies Diploma in signal processing and communication theory from the Polytechnic University of Catalonia, Barcelona, Spain, in 2005; the M.S. degree in electronics and telecommunications from Ensenada Center for Scientific Research and Higher Education (CICESE), Ensenada, Mexico, in 2006; and the Ph.D. degree in mobile communication systems from the University of Agder, Kristiansand, Norway, in 2009. From 2009 to 2011, he was with the School of Engineering, Universidad Panamericana Campus Aguascalientes, Aguascalientes. Since January 2012, he has been with the Faculty of Science, Universidad Autonoma de San Luis Potosi, San Luis Potosi, Mexico. His research interests include modeling and simulation of mobile fading channels for wireless communication systems, and physical layer aspects of orthogonal frequency-division multiplexing and multicarrier code-division multiple access systems. Dr. Gutierrez has served as a Guest Editor for Modelling and Simulation in Engineering, Procedia Technology, and Research in Computing Science, and as Technical Program Committee member for various international conferences. He is a member of the Mexican National System of Researchers (SNI).

A. Castillo Atoche received the Ph.D. (2010) and the M.Sc. (2002) degrees in Electrical Engineering from CINVESTAV Guadalajara, Mexico and the Electronic Engineer (2000) by Mérida Institute of Technology, Mexico. He is Professor of the Universidad Autónoma de Yucatán and Mérida Institute of Technology. His research work is in applications of intelligent signal processing to remote sensing imagery, real time systems applications, embedded systems, MPSoC and FPGA hw/sw co-design.

References

[1] A. Alimohammad, S. F. Fard and B. F. Cockburn, "Reconfigurable performance measurement system-on-a-chip for baseband wireless algorithm design and verification," *IEEE Wireless Communications,* vol. 19, pp. 84-91, 12 2012.

[2] D. W. Matolak, "V2V Communication Channels: State of Knowledge, New Results, and What's Next," in *Communication Technologies for Vehicles*, 2013.

[3] A. Alimohammad, S. F. Fard and B. F. Cockburn, "Hardware Implementation of Nakagami and Weibull Variate Generators," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 20, pp. 1276-1284, Jul 2012.

[4] N. M. a. E. D. Wirastuti and N. P. Sastra, "APPLICATION OF THE SUZUKI DISTRIBUTION TO SIMULATION OF SHADOWING/FADING EFFECTS IN MOBILE COMMUNICATION," in *ICST*, Surabaya Indonesia, 2007.

[5] S. Chatterjee, A. S. Hadi and B. Price, Regression Analysis by Example, 3rd ed., New, York: Wiley-Interscience, 1999.

[6] J. Vázquez Castillo, L. Vela-Garcia, C. A. Gutiérrez and R. Parra-Michel, "A reconfigurable hardware architecture for the simulation of Rayleigh fading channels under arbitrary scattering conditions," *International Journal of Electronics and Communications,* pp. 1-13, 2014.

[7] P. Huang, D. Rajan and J. Camp, "Weibull and Suzuki fading channel generator design to reduce hardware resources," in *2013 IEEE Wireless Communications and Networking Conference (WCNC)*, 2013.

[8] L. Canche Santos, A. Castillo Atoche, J. Vázquez Castillo, O. Longoria-Gándara, R. Carrasco Alvarez and J. Ortegon Aguilar, "An improved hardware design for matrix inverse based on systolic array QR decomposition and piecewise polynomial approximation," 2015.

[9] S. D. Munoz and J. Hormigo, "High-Throughput FPGA Implementation of QR Decomposition," *IEEE Transactions on Circuits and Systems II: Express Briefs,* vol. 62, pp. 861-865, Sept 2015.

[10] A. Vázquez and J. D. Bruguera, "Iterative Algorithm and Architecture for Exponential, Logarithm, Powering, and Root Extraction," *IEEE Transactions on Computers,* vol. 62, pp. 1721-1731, Sept 2013.

[11] M. Shabany, D. Patel and P. G. Gulak, "A Low-Latency Low-Power QR-Decomposition ASIC Implementation in 0.13 CMOS," *IEEE Transactions on Circuits and Systems I: Regular Papers,* vol. 60, pp. 327-340, Feb 2013.

[12] V. Tiwari and N. Khare, "Hardware implementation of neural network with Sigmoidal activation functions using CORDIC," *Microprocessors and Microsystems,* vol. 39, pp. 373-381, Aug 2015.

[13] F. Ren and Y. R. Zheng, "A novel emulator for discrete-time MIMO triply selective fading channels," *IEEE Transactions on Circuits and Systems I: Regular Papers,* vol. 57, pp. 2542-2551, Sep 2010.

[14] D.-U. Lee, R. C. C. Cheung, W. Luk and J. D. Villasenor, "Hierarchical Segmentation for Hardware Function Evaluation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 17, pp. 103-116, Jan 2009.

[15] J.-M. Muller, Elementary Functions: Algorithms and Implementation, 2nd ed., Birkhäuser, 2005.

[16] E. W. Cheney, Introduction to Approximation Theory, 2 edition ed., New York, N.Y.: Amer Mathematical Society, 2000.

[17] G. Strang, Linear Algebra and Its Applications, 4th Edition, 4th edition ed., Belmont, CA: Cengage Learning, 2006.

[18] P. Borwein and T. Erdélyi, Polynomials and Polynomial Inequalities. Graduate Texts in Mathematics, vol. 161, Springer-Verlag, New York, NY, 1995.

[19] R. Yates, "Computer Science and Engineering. University of Washington. CSE 467: Advanced Digital Design," 23 08 2007. [Online]. Available: http://www.cs.washington.edu/education/courses/. [Accessed 15 06 2016].

[20] S. S. Rao, Engineering Optimization: Theory and Practice, 4 edition ed., Hoboken, N.J: Wiley, 2009.

[21] J. A. Nelder and R. Mead, "A Simplex Method for Function Minimization," *The Computer Journal,* vol. 7, pp. 308-313, 1 1965.

[22] R. Hamming, Numerical Methods for Scientists and Engineers, Courier Corporation, 2012.

[23] M. Abramowitz and I. A. Stegun, Handbook of Mathematical Functions: with Formulas, Graphs, and Mathematical Tables, Courier Corporation, 2012.

[24] P. M. a. E. Fiesler, "Neural network adaptations to hardware implementations," *Handbook of Neural Computation,* vol. 1, p. 2, 1997.

[25] Y. Li, B. Ai, X. Cheng, S. Lin and Z. Zhong, "A TDL Based Non-WSSUS Vehicle-to-Vehicle Channel Model," *International Journal of Antennas and Propagation,* vol. 2013, p. e103461, #nov# 2013.

List of Figures

List of Tables

Index