DETERMINAÇÃO DE APROXIMAÇÃO DE FUNÇÕES NÃO LINEARES PARA SISTEMAS EMBARCADOS UTILIZANDO ALGORITMOS GENÉTICOS HÍBRIDO

Juan M. Mauricio
juan@dee.ufma.br

Sebastian Y.C. Catunda
catunda@dee.ufma.br

Osvaldo R. Saavedra
osvaldo@dee.ufma.br

João V. FonsecaNeto
jviana@dee.ufma.br

Departamento de Engenharia Elétrica DEE-UFMA
Av. Dos Portugueses s/n, Bacanga
CEP 65080-040, São Luis, Maranhão, Brasil

ABSTRACT

In this work a procedure for determining piecewise linear approximation of nonlinear functions for low-cost embedded systems with fixed-point calculations capacity. An evolutionary hierarchical algorithm has been developed for determining the locations and minimal number of breakpoints and the minimal size of a look-up table for storing these breakpoints, in order to generate the approximate function. The hierarchical algorithm consists of two levels and, for the lower level, a hybrid algorithm consisting of a standard genetic algorithm together with a local search mathematical programming algorithm have been developed. The hybrid and the standard genetic algorithms are tested for the approximation of a first-quadrant sine function and the obtained results are presented for several input and output signal generation resolutions. A comparison between the proposed and standard genetic algorithms is presented from the results obtained.

KEYWORDS: Nonlinear functions approximation, Piecewise linear functions, Uncertainty propagation, Embedded systems, Look-up Table, Genetic algorithms, Hybrid algorithms.

RESUMO

Neste trabalho apresenta-se um procedimento para a determinação de aproximação linear por partes de funções não lineares para sistemas embarcados de baixo custo com capacidade de cálculo em ponto fixo. Para resolver este problema, desenvolveu-se um algoritmo hierárquico evolutivo que determinará as posições e número mínimo de pontos de quebra e tamanho mínimo da tabela de equivalência para armazenar esse pontos de quebra, para gerar os valores da função aproximada. O algoritmo hierárquico consiste de dois níveis e, para o nível inferior, foi desenvolvido um algoritmo híbrido composto por um algoritmo genético padrão junto com um algoritmo de programação matemática de busca local. Os algoritmos genético padrão e o híbrido são testados para o caso de aproximação da função seno no primeiro quadrante e os resultados obtidos são apresentados para diversas resoluções de entrada e de geração dos valores de saída. Apresenta-se uma comparação do algoritmo proposto com algoritmo genético padrão a partir dos resultados obtidos.

PALAVRAS-CHAVES: Aproximação de funções não lineares, Funções lineares por partes, Propagação de incertezas, Tabelas de equivalência, Algoritmos genéticos, Algoritmos Híbridos.

1 INTRODUÇÃO

Em diversas aplicações em sistemas embarcados existe o problema de aproximar ou gerar valores de funções não lineares. Exemplos destes sistemas são descritos a seguir: Em instrumentação, diversas aplicações necessitam gerar funções não lineares para reconstrução dos valores de medição, que podem ser utilizados para compensação das não linearidades do sensor ou para compensação da influência de grandezas que causem interferências (Catunda, 2002). Em controle, sinais de referência são gerados digitalmente utilizando funções aproximadas (Julian, 1998). Em sistemas digitais de posicionamento, geralmente necessita-se converter a informação de ângulo em posição e vice-versa, utilizando funções trigonométricas (Lygouras, 1999).

Essas funções não lineares podem ser implementadas diretamente em sistemas embarcados com capacidade de processamento de cálculo em ponto flutuante. Entretanto, em diversas aplicações se faz necessário a utilização de sistemas embarcados em que a implementação destas funções não lineares não é direta devido às restrições de cálculo em ponto fixo e resolução limitada, características da arquitetura do processador empregado. Esses sistemas embarcados baseados em hardware de baixo custo podem ser compostos de: microcontroladores de resolução

limitada, FPGA, ASIC, DSP em ponto fixo (Catunda, 2002).

Para a implementação de aproximação de funções não lineares em sistemas embarcados de baixo custo com capacidade de cálculo em ponto fixo, devem-se considerar aspectos como: incertezas associadas as variáveis livres, saturação na representação de números e efeitos de quantização devido à resolução de armazenamento dos pontos de quebra. Assim, um procedimento automatizado deve ser implementado tal que procure as soluções de funções aproximadas, considerando os aspectos anteriormente descritos. Neste panorama pode-se implementar um algoritmo genético para encontrar soluções, com um processo de evolução robusto de uma representação das soluções (Ahmed, 1997).

Funções aproximadas podem ser implementadas utilizando-se aproximação por funções lineares por partes, determinando-se o tamanho da tabela de equivalência (LUT), que será utilizada, levando em consideração aspectos como restrições e limites para aproximação de funções em sistemas embarcados. Neste sentido a implementação do algoritmo genético deve atingir os objetivos seguintes: calcular o número e as posições dos pontos de quebra necessários para reproduzir os valores aproximados da função não linear a partir dos pontos de quebra.

2 FORMULAÇÃO DO PROBLEMA

Considere-se um sistema digital embarcado em que se necessita gerar valores de uma função não linear, utilizando aproximação de funções lineares por partes. Utiliza-se um sistema embarcado de baixo custo, em que os cálculos sejam realizados em ponto fixo, com resolução limitada e com limitações de espaço de memória devido às restrições da arquitetura do projeto.

A função não linear é definida por $y = f(x)$, $x \in [x_{\min}, x_{\max}]$ e $y \in [y_{\min}, y_{\max}]$, em que cada elemento do conjunto de entrada x é quantizado com uma resolução equivalente a N bits e os elementos do conjunto de valores de y são gerados com uma exatidão equivalente a uma resolução de N_Y bits. Supõe-se que os valores de x e y são positivos e seus valores mínimos são zeros, $x_{\min} = 0$ e $y_{\min} = 0$, que podem ser obtidos adicionando-se constantes à função f. Essa constante pode ser subtraída após a geração dos valores de y (Catunda, 2002).

A função não linear é aproximada por uma função linear por partes considerando que os cálculos podem ser realizados com uma precisão equivalente a N_T bits de resolução, com $N_T \geq N_Y$. Os valores dos pontos de quebra para reproduzir a função aproximada são armazenados em uma tabela de equivalência (LUT - *Look-up Table*) embutido em um sistema embarcado, como é mostrado na Figura 1.

A tabela de equivalência (LUT), contém m pontos de quebra $p_x = \{p_{x1}, ..., p_{xm}\}$, $p_x \subseteq x$, e $p_y = \{p_{y1}, ..., p_{ym}\}$, $p_y \subseteq y$ com $m \leq n$ (Catunda, 2002). Os pontos de quebra são armazenados com uma resolução de N_T bits. A função aproximada $s = \tilde{f}(x)$ é composta por m funções $\tilde{f}_j$ definidas nos intervalos $[p_{xj}, p_{xj+1}]$, com $j = 1,...,m$, $p_{x1} = x_1$ e $p_{xm} = x_n$, é definida em geral como:

$$s = \tilde{f}(x) = a_j + b_j\left(x - p_{xj}\right) \tag{1}$$

com:

$$a_j = p_{yj} \text{ e } b_j = \frac{p_{yj+1} - p_{yj}}{p_{xj+1} - p_{xj}} \tag{2}$$

para

$$p_{xj} \leq x < p_{xj+1} \text{ e } \tilde{f}_{m+1}(x_n) = \tilde{f}_m(x_n) \tag{3}$$

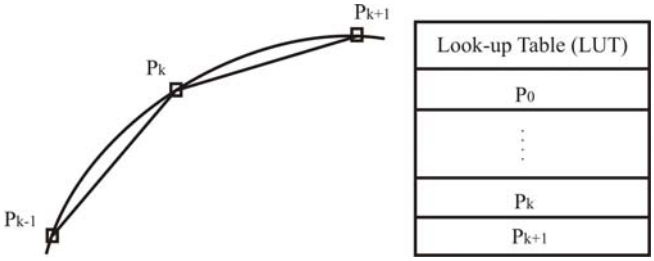

Figura 1. Aproximação linear por partes de uma função não linear.

O erro de aproximação é definido como $\varepsilon_A = s - y$, limitado por $[\varepsilon_{Amin}, \varepsilon_{Amax}]$. O problema de aproximação consiste em determinar valores de s tal que o erro de aproximação se encontre dentro deste limites. Para todo x, os limites do erro de aproximação dependem da precisão especificada na faze de projeto e da incerteza propagada associada com a variável livre (Catunda, 2002).

$$\varepsilon_{A\min} = -\hat{\varepsilon}_y - \varepsilon_{p\min} \text{ e } \varepsilon_{A\max} = \hat{\varepsilon}_y - \varepsilon_{p\max} \tag{4}$$

sendo $\hat{\varepsilon}_y = 2^{N_Y}$, $\varepsilon_{p\min}$ e $\varepsilon_{p\max}$ os limites da incerteza propagada.

O melhor valor aproximado, definido como y_q, corresponde à quantização de y utilizando N_T bits de resolução e é o inteiro mais próximo de y (considerando uma representação inteira). Então, para normalizar o problema desconsiderando a forma da função, a seguinte transformação é aplicada: $S = s - y_q$, $Y = y - y_q$, $E_{Amin} = \varepsilon_{Amin} + Y$, $E_{Amax} = \varepsilon_{Amax} + Y$ e $P_y = p_y - y_q$ ($P_x = p_x$). Esta transformação desloca as variáveis para em torno de zero, desta maneira elas assumem valores 0, ±1, ±2, ... e $\varepsilon_{Amin} = Y - S$ (Catunda, 2002), (Catunda, 2003).

3 PROCEDIMENTO PARA SOLUÇÃO

Na metodologia proposta desenvolve-se um algoritmo hierárquico evolutivo cujo objetivo é de procurar uma solução ótima do problema de aproximação de funções. O procedimento proposto possui dois níveis hierárquicos, como representado na Figura 2. No primeiro nível define-se o tamanho da tabela de equivalência ($m \times N_T$), com m pontos de quebra e uma resolução equivalente de N_T bits. O algoritmo para este nível procura as soluções de aproximação a partir de um par de resoluções de entrada (N e N_Y) e número de pontos de quebra inicial (m). Encontrando a solução para o tamanho da LUT em bits

($m \times N_T$), seguidamente o algoritmo decresce um desses parâmetros e tenta achar a solução.

No segundo nível é implementado um algoritmo evolutivo, que procura uma solução para a determinação das posições dos pontos de quebra para a condição especificada no primeiro nível. Se a solução é encontrada, esta é repassada ao primeiro nível, que procedera reduzindo o tamanho da tabela de equivalência. Com esta nova condição, o segundo nível é novamente ativado. Se uma solução factível não é encontrada, considera-se como resultado a última solução encontrada (solução incumbente) (Mauricio, 2004).

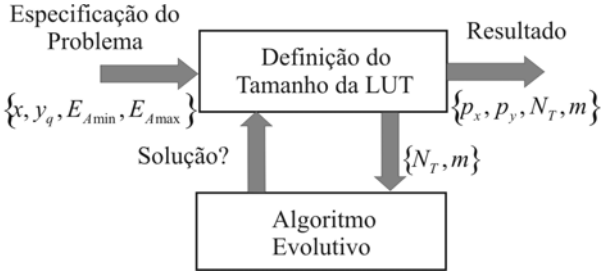

Figura 2. Diagrama do procedimento hierárquico de busca.

O segundo nível foi desenvolvido em duas partes. Inicialmente, foi desenvolvido um algoritmo genético padrão e foi avaliado em seu desempenho através de estudo de caso. Os resultados apresentados com o uso desse algoritmo foram satisfatórios para casos com pequenas resoluções de N e N_Y até 10 bits. Entretanto, esse algoritmo apresentou problemas de convergência para resoluções maiores a 10 bits. De forma a melhorar a convergência do algoritmo evolutivo, em seguida, ao algoritmo genético padrão foi incorporado um bloco de programação matemática de busca local, como representado na Figura 3. Desta forma utiliza-se um algoritmo genético padrão para determinar uma estrutura apropriada da solução, enquanto que o algoritmo de programação matemática identifica a estrutura de super-indivíduos usando como espaço de busca a população atual.

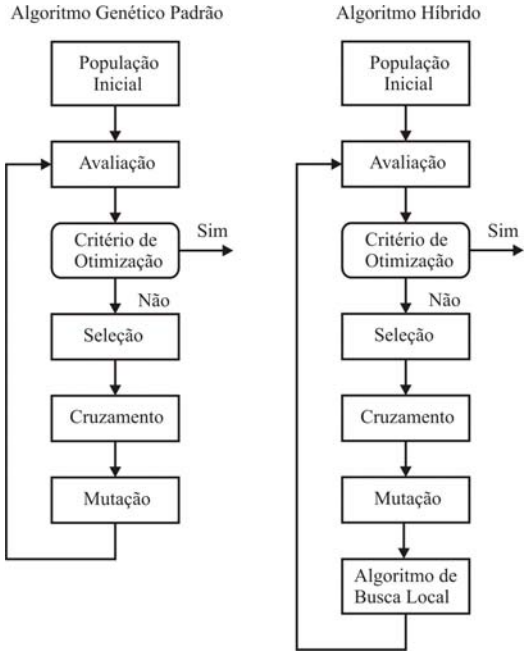

Figura 3. Estrutura dos algoritmos evolutivos propostos para o segundo nível de hierarquia.

A seguir , desenvolvem-se aspectos que envolvem a implementação dos algoritmos propostos apresentados na Figura 3, como por exemplo: representação dos indivíduos, avaliação da função de aptidão, método de seleção, operadores de reprodução genética e o algoritmo de programação matemática.

3.1 População Inicial

No algoritmo implementado utiliza-se uma representação dos cromossomos com codificação inteira, em que cada cromossomo tem um comprimento de m pontos de quebra (p_{xi}, p_{yi}), $i = 1, \ldots, m$, representados por valores inteiros com tamanhos de palavras de N e N_T bits respectivamente. O cromossomo é inicializado aleatoriamente, porém, projetado dentro de uma região viável de variação dos pontos de quebra: $p_{xi} \in [0, 2^N\text{-}1]$ e $p_{yi} \in [0, 2^{N_T}\text{-}1]$. O tamanho da população inicial é definido por *Pop* indivíduos, conseqüentemente a população inicial se encontra constituída por *Pop* cromossomos. Na Figura 4 apresenta-se a estrutura da população de indivíduos a ser otimizada pelo algoritmo genético.

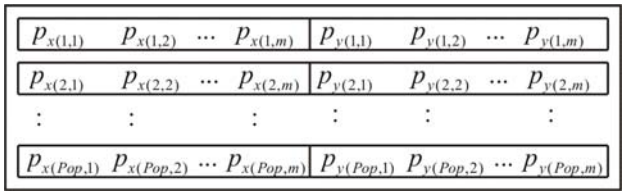

Figura 4. Estrutura da população total de indivíduos.

3.2 Avaliação da Função de Aptidão

O erro de aproximação de cada indivíduo da população é dado por $\varepsilon_i = s_i - y_q$, em que s_i é a função de aproximação do indivíduo i, y_q é o valor quantizado da função y. O algoritmo genético tem que minimizar o erro de aproximação considerando sus limites. Para cada indivíduo, a função e o erro de aproximação são calculados utilizando funções lineares por partes (Catunda, 2003). Como os limites do erro de aproximação não são constantes para a faixa de variação de x (devido à própria quantização e à propagação da incerteza na variável independente) o erro de aproximação normalizado é determinado por:

$$\varepsilon_{AN}(x) = \left\{ \frac{\varepsilon_A(x)}{E_{A\max}(x)} \middle| \varepsilon_A(x) > 0 \;\; ; \;\; \frac{\varepsilon_A(x)}{E_{A\min}(x)} \middle| \varepsilon_A(x) < 0 \right\} \tag{5}$$

Assim, cada indivíduo da população tem associado uma função de aptidão, que reflete quão boa ela é, comparada com outras da população. Define-se a função de aptidão como *fit*, para cada indivíduo i, determinada como uma combinação de três parâmetros:

$$fit(i) = f_1(i) + \sqrt{f_2^2(i) + f_3^2(i)} / w_f \tag{6}$$

sendo w_f um fator de ponderação, que foi ajustado para 40, e f_1, f_2 e f_3, são respectivamente valores máximo, médio e desvio padrão, dados por:

$$\begin{aligned} f_1(i) &= \max\left(\varepsilon_{AN}^2\right) \\ f_2(i) &= mean\left(\varepsilon_{AN}^2\right) \\ f_3(i) &= std\left(\varepsilon_{AN}^2\right) \end{aligned} \tag{7}$$

A função de aptidão, definida em (6), força o algoritmo a minimizar o máximo do erro de aproximação normalizado considerando a minimização também dos valores médio e desvio padrão deste erro. O fator de ponderação tem por objetivo evitar o mascaramento do primeiro parâmetro pelos outros dois (Catunda, 2003).

3.3 Método de Seleção

A partir da população principal, n_{ind} indivíduos são selecionados para gerar novos cromossomos, utilizando uma pontuação baseada em atributos não lineares (tipo *rank*) e seleção através de roleta.

3.4 Cruzamento e Mutação

A operação de cruzamento troca parte da informação genética entre dois indivíduos, produzindo novas soluções potenciais com algumas características dos pais. Neste algoritmo aplica-se o operador de cruzamento binário simples (de um ponto).

Depois da operação de cruzamento os indivíduos da população atual sofrem um processo de mutação, com o objetivo de introduzir indivíduos com nova codificação na população, tendo como conseqüência uma maior exploração do espaço de busca.

3.5 Algoritmo de Programação Matemática

O algoritmo de programação matemática é responsável pela criação de super-indivíduos a partir da população atual de indivíduos. Uma vez criados os super-indivíduos, estes competirão por sua sobrevivência com os indivíduos da população atual introduzindo novas soluções à população atual com uma melhora do processo de exploração (Yen, 1997). Para cada geração, super-indivíduos são gerados a parir da população $P(t)$ alterado pelo processo de seleção e reprodução. As novas soluções serão inseridas e substituirão aos piores indivíduos da população $P(t)$. Cada super-indivíduos é construído através de um algoritmo de busca local que avalia o conjunto atual de dados para determinar uma promissória direção de busca (Yen, 1997).

O procedimento do algoritmo de busca local consiste em ajustar as posições dos pontos de quebra das soluções da população atual $P(t)$. Considera-se o ajuste de uma solução de m pontos de quebra ($p_1, p_2, ..., p_i, ..., p_m$), define-se uma nova solução obtida pelo algoritmo de busca local denotada por ($p_1, p_2, ..., p'_i, ..., p_m$), sendo p'_i o novo ponto de quebra ajustado pelo algoritmo de busca local que substituirá p_i depois do resultado da avaliação da função de aptidão (Yen, 1998). Para realizar o ajuste de um ponto de quebra p_i, o seguinte procedimento de busca é descrito sumariamente:

Passo 1: Especificar um ponto de quebra p_i para ajuste;

Passo 2: Gerar dois novos pontos a partir de p_i utilizando-se as equações seguintes:

$$p'_i = p_i + \gamma (p_{i+1} - p_i)$$
$$p''_i = p_i - \beta (p_i - p_{i-1})$$

sendo γ o coeficiente de expansão ($0 \le \gamma \le 1$) e β o coeficiente de contração ($0 \le \beta \le 1$).

Passo 3: Avaliar a função de aptidão para cada ponto de quebra p'_i e p''_i.

Passo 4: Substituir p_i pelo ponto de quebra p'_i ou p''_i que apresente menor valor da função de aptidão.

Passo 5: Voltar ao Passo 1 e selecionar o seguinte ponto de quebra para ajuste.

sendo γ e β variáveis aleatórias que toma valores no intervalo [0 , 1].

O novo ponto de quebra gerado ao redor do ponto de quebra p_i, que pode ser criado a partir: da operação de expansão ($p'_i \in < p_i , p_{i+1} >$) ou da operação de contração ($p''_i \in < p_{i-1} , p_i >$). Esta flexibilidade permite ao algoritmo de busca explorar o espaço de busca com uma maior liberdade. Isto também facilita uma sintonização das soluções ao redor de um ponto ótimo

4 RESULTADOS UTILIZANDO OS ALGORITMOS PROPOSTOS

Na atualidade, geradores digitais de funções são empregados em diversas áreas, como aplicações em radar, sistemas de comunicações, aplicações de controle digital, em instrumentação. Neste sentido existe o interesse de viabilizar o problema de gerar valores de funções digitais utilizando-se sistemas embarcados de baixo custo com o objetivo de substituir geradores digitais convencionais e caros (Yeary, 2004).

Para verificação dos algoritmos propostos, considera-se o estudo de caso de aproximar a função seno no primeiro quadrante utilizando aproximação linear por partes. Considera-se a função seno definida como $y = \text{sen}(x)$, com limites $x \in [0, \pi/2]$ e $y \in [0, 1]$. Analisa-se o caso que não existe incerteza associada à variável livre, e para isto faz-se $N_Y = N$.

Para cada caso de aproximação, variando-se as exigencias de resolução de entrada e saída, utiliza-se uma probabilidade de cruzamento $P_c = 0{,}9$ e uma probabilidade de mutação $P_m = 0{,}04$. As simulações foram realizadas utilizando o programa MATLAB para um computador Pentium IV, 2,4 GHz e memória de 512MB.

4.1 Resultados Utilizando o Algoritmo Genético Padrão

Os resultados apresentados com o uso desse algoritmo foram satisfatórios para casos de pequenas resoluções de N e N_Y até 10 bits, respectivamente. Entretanto, esse algoritmo apresentou problemas de convergência para resoluções maiores de 10 bit. Na Tabela 1 apresentam-se os parâmetros e resultados utilizados nas simulações e o tamanho mínimo da LUT (em bits) obtido para cada aproximação especificada.

Tabela 1. Parâmetros e resultados da aproximação utilizando o algoritmo genético padrão.

Parâmetros de entrada			Resultados			
N	N_Y	Pop	N_T	m	LUT (bits)	Tempo computacional por época (s)
8	8	40	10	9	90	0,188
9	9	40	11	13	143	0,280
10	10	50	12	17	204	0,485

4.2 Resultados Utilizando o Algoritmo Híbrido

Os resultados apresentados com o uso do algoritmo híbrido foram satisfatórios, conseguindo encontrar soluções para resoluções de N e N_Y até 12 bits, respectivamente. Na Tabela 2 contém os parâmetros e resultados utilizados nas simulações e o tamanho mínimo da LUT (em bits) obtido para cada aproximação especificada.

Tabela 2. Parâmetros e resultados da aproximação utilizando o algoritmo híbrido.

Parâmetros de entrada			Resultados			
N	N_Y	Pop	N_T	m	LUT (bits)	Tempo computacional por época (s)
8	8	40	10	9	90	0,188
9	9	40	11	13	143	0,280
10	10	50	12	17	204	0,485
11	11	60	14	22	308	1,93
12	12	80	15	31	465	6,15

4.3 Comparação computacional entre o AG padrão e o Algoritmo Híbrido

A seguir, comparam-se as taxas de convergência do AG padão e o algoritmo híbrido. Para cada algortimo foram feitas 100 simulações independentemente, determinando-se os valores da média final de épocas e desvio médio padrão em que são encontradas as soluções. Para cada caso utiliza-se uma taxa de cruzamento $P_c = 0{,}9$ e uma taxa de mutação $P_m = 0{,}04$. Os dois algoritmos são inicializados com o mesmo valor da semente aleatória, isto é, a geração da seqüência de números aleatórios é a mesma para cada caso analisado, sendo os resultados apresentados na Tabela 3.

Tabela 3. Comparação de taxa de convergência do AG padrão e o Algoritmo híbrido.

Parâmetros				Algoritmo Genético Padrão		Algoritmo Híbrido	
N	N_Y	N_T	m	Média época final	Desvio padrão médio	Média época final	Desvio padrão médio
8	8	10	9	383	56	62	35
9	9	11	13	769	42	40	1,8
10	10	12	17	1522	123	341	65
11	11	14	22	-	-	850	83
12	12	15	31	-	-	1522	194

Na Figura 5 apresenta-se um gráfico típico de convergência para uma simulação dos algoritmos propostos, com parâmetros $N = 8$, $N_Y = 8$, $N_T = 10$ e $m = 9$. O algoritmo genético padrão converge à solução com 307 gerações, entretanto, o algoritmo híbrido converge à solução com 57 gerações.

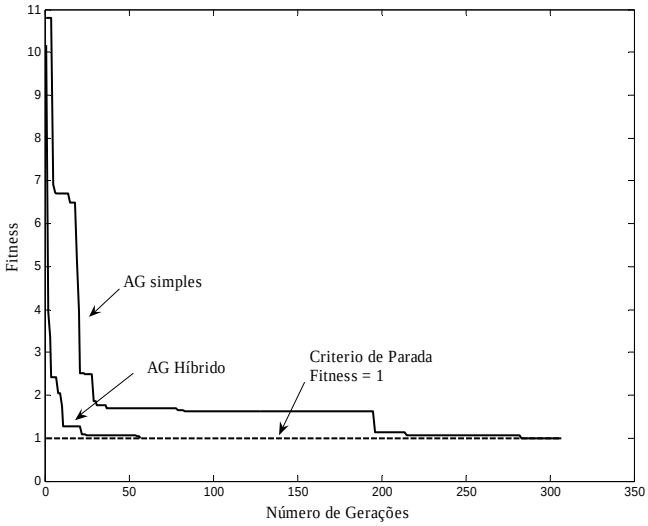

Figura 5. Simulação do AG Padrão e Algoritmo Híbrido com parâmetros $N = 8$, $N_T = 10$ e $m = 9$.

Na Figura 6 apresenta-se um gráfico típico de convergência para uma simulação dos algoritmos propostos, com parâmetros $N = 9$, $N_Y = 9$, $N_T = 11$ e $m = 3$. O algoritmo genético padrão converge à solução com 661 gerações , entretanto, o algoritmo híbrido converge à solução com 47 gerações.

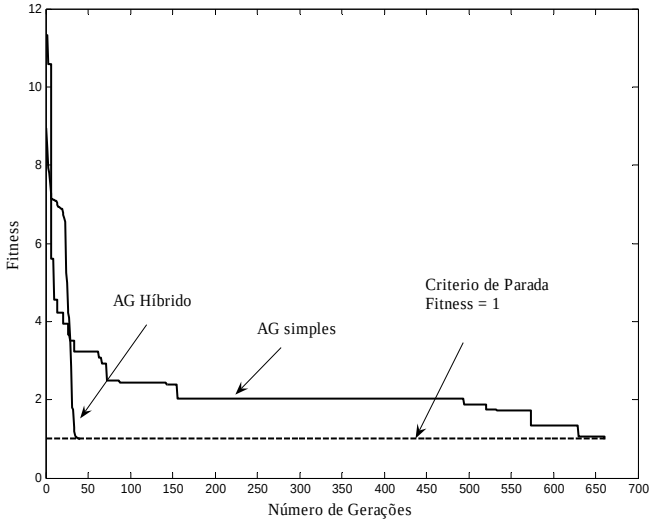

Figura 6. Simulação do AG Padrão e Algoritmo Híbrido com parâmetros $N = 9$, $N_T = 11$ e $m = 13$.

4.4 Resultados das soluções da Aproximação da Função Seno

Nas Figuras 7 e 8 apresentam-se as soluções para a representação da aproximação da função seno, sendo $S = y - y_q$ o erro de aproximação normalizado, E_{Amin} e E_{Amax} os limites de erros de aproximação normalizados. Observa-se que o erro normalizado S não ultrapassa os limites de erros de aproximação normalizados. Em ambos casos o valor do *fitness* foi menor que um.

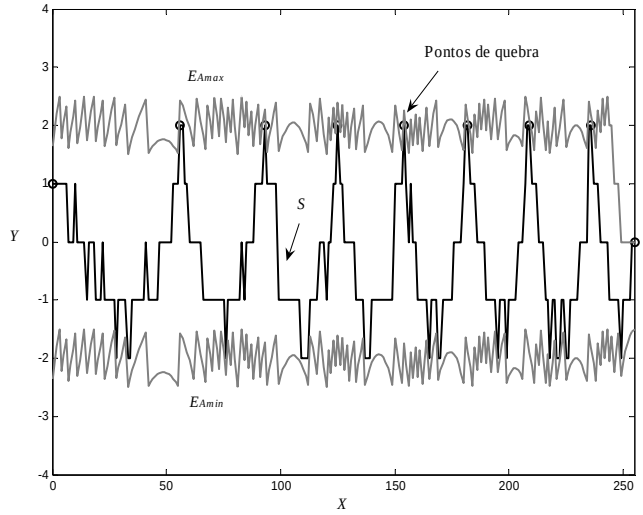

Figura 7.Erro de aproximação para a função seno com N = 8, N_Y = 8, N_T = 10, m = 9 e *fitness* = 0,9967.

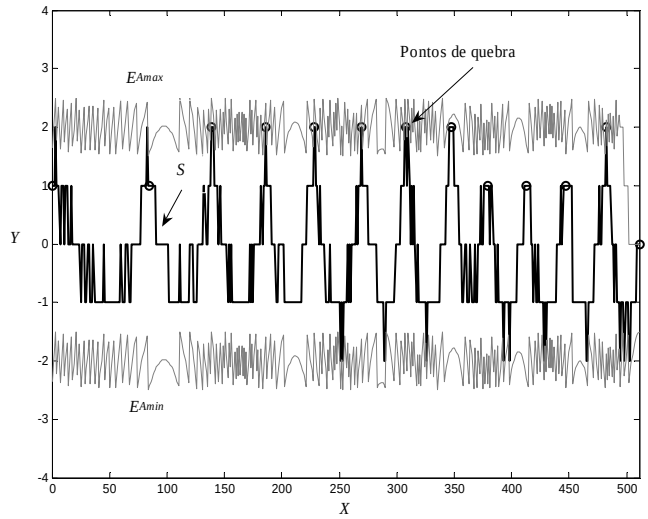

Figura 8. Erro de aproximação para a função seno com N = 9, N_Y = 9, N_T = 11, m = 13 e *fitness* = 0,9051.

5 CONCLUSÕES

Neste trabalho foi apresentado um procedimento baseado em meta heurística para determinar as posições dos pontos de quebra com um tamanho da tabela de equivalência (LUT) mínimo, para aproximação linear por partes de funções não lineares para sistemas embarcados de baixo custo com capacidade de cálculo em ponto fixo. Para isto, foram considerados aspectos como: incertezas associadas as variáveis livres, saturação na representação de números e efeitos de quantização devido à resolução de armazenamento dos pontos de quebra. Conseqüentemente, foram analisados as restrições e limites para aproximação de funções não lineares em sistemas embarcados com capacidade de calculo em ponto fixo.

A utilização de algoritmos genéticos para determinação de funções de aproximação mostrou ser uma alternativa viável para resolver este tipo de problemas. Os testes realizados mostraram que o algoritmo genético padrão apresenta um problema de convergência quando o espaço de busca de soluções é muito complexo, isto é, resoluções de N e N_Y maiores de 10 bits. Uma alternativa de solução para espaços de busca complexos foi o desenvolvimento de um algoritmo híbrido, composto por um algoritmo genético padrão e um algoritmo de programação matemática de busca local. O algoritmo híbrido mostrou um melhor desempenho na busca de soluções, conseguindo soluções próximas do ótimo com uma maior taxa de convergência com resoluções de N e N_Y até 12 bits. Um caso de estudo foi apresentado, a aproximação da função seno no primeiro quadrante. Os testes realizados mostram que o algoritmo é capaz de alcançar tamanhos mínimos da LUT para diferentes valores da resolução de entrada.

6 REFERÊNCIAS BIBLIOGRÁFICAS

Catunda, S.Y.C. and Saavedra, O.R. (2002), Constraints definition and evaluation of piecewise polynomial approximation functions for embedded systems, *19th IEEE Proceedings of Instrumentation and Measurement Technology Conference*, pp. 1103-1108 vol2.

Julian, P., Jordan, M., Desages., A. (1998), Canonical piecewise linear approximation of smooth functions, *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, Vol. 45, Issue: 5, pp: 567-571.

Lygouras, J.N. (1999), Memory reduction in look-up tables for fast symmetric function generators, *IEEE Transactions on Instrumentation and Measurement*, Vol. 48, Issue: 6, pp: 1254-1258.

Ahmed, M.A., Dejong, K.A. (1997), Function Approximator design using Genetic Algorithms, *IEEE International Conference on Evolutionary Computation*, pp: 519 – 524.

Catunda, S.Y.C., Saavedra, O.R., FonsecaNeto, J.V., e Morais, M.R.A. (2003), *Determinação de tabela de equivalência e pontos de quebra de funções de aproximação lineares por partes usando computação evolutiva*, 6° SBIA Simpósio Brasileiro de Automação Inteligente, pp: 662-667.

Mauricio, J.M., Catunda, S.Y.C., Saavedra, O.R., FonsecaNeto, J.V. (2004), *Aproximação de funções: Uma abordagem utilizando computação evolutiva híbrida*, 8th Simpósio Brasileiro de Redes Neurais, São Luis, Brasil, 3590.

Yeary, M.B., Fink, R.J., Beck, D., Guidry, D.W., Burns, M. (2004), A DSP-based mixed-signal waveform generator, *IEEE Transactions on Instrumentation and Measurement*, Volume: 53, Issue: 3, pp: 665-671.

Yen, J., Bogju Lee (1997), A simples genetic algorithm hybrid, *IEEE International Conference on Evolutionary Computation*, pp: 175-180.

Yen, J., Liao, J.C., Bogju Lee, Randolph, D. (1998), A hybrid approach to modeling metabolic systems using a genetic algorithm and simplex method, *IEEE Transactions on Systems, Man and Cybernetics*, Volume : 28, Issue : 2, pp: 173-191.