Desenvolvimento e Implementação de um Aplicativo *Mobile* para Cálculo de FFT e MRA

Thyago Leite de Vasconcelos Lima[1], Filipe Vidal Souto[2], Thaís Christine Borges da Silva[2], Abel Cavalcante Lima Filho[2], Francisco Antônio Belo[2]

1 Instituto Federal de Educação, Ciência e Tecnologia da Paraíba – IFPB, *Campus* Itabaiana

2 Universidade Federal da Paraíba, *Campus* Universitário I, João Pessoa

***Resumo*—O presente trabalho trata do desenvolvimento e implementação de um aplicativo *mobile* para cálculo da FFT e análise multiresolução (MRA). A escolha da plataforma é justificada por seu amplo uso em escala global, bem como seu poder de processamento superior comparado a muitas placas de desenvolvimento. Os resultados obtidos com a aplicação foram comparados com os resultados obtidos pelo MATLAB®, revelando um desempenho muito semelhante entre eles. Espera-se que a aplicação desenvolvida seja usada como base para o desenvolvimento de sistemas mais aprimorados, como sistemas de diagnóstico portáteis e de baixo custo.**

Palavras-Chave—Aplicativo Mobile, FFT, MRA.

***Abstract*— The present work deals with the development and implementation of a mobile application for FFT and multiresolution analysis (MRA) calculation. The choice of platform is justified by its wide use on a global scale as well as its superior processing power compared to many development boards. The results obtained with the application were compared to the results achieved by MATLAB®, revealing a very similar performance between them. It is expected that the developed application will be used as the basis for the development of more improved systems, such as portable and low cost diagnosis systems.**

Keywords—Mobile App, FFT, MRA.

INTRODUÇÃO

O processamento de sinais é utilizado para extrair informações importantes em várias aplicações. As ferramentas mais eficientes para o processamento de sinais estacionários e não estacionários são amplamente conhecidas e aceitas, como a Análise Multiresolução (MRA) e a Transformada Rápida de Fourier (FFT) [1] - [6] que foram escolhidas para este trabalho.

Smartphones possuem um poder de processamento comparável ao dos computadores *high-end* de seis anos atrás e significativamente superior ao de DSP's e microcontroladores empregados atualmente em aplicativos de automação e IoT [10]. Sua adoção por consumidores tem crescido amplamente nos últimos anos [7], [8], bem como o uso e desenvolvimento de aplicativos móveis [9]. Além disso, sua aplicação em pesquisas científicas já pode ser observada em áreas como robótica [11], medicina [12], [13], máquinas elétricas [14] e processamento de sinais em tempo real [15].

Neste contexto, o objetivo do presente trabalho é apresentar o desenvolvimento e a implementação de uma aplicação móvel como ferramenta de análise de sinais, utilizando as capacidades dos *smartphones* para ampliar possibilidades e recursos, sendo assim uma alternativa às placas de desenvolvimento. Para validação, os resultados obtidos com o aplicativo desenvolvido foram comparados com os resultados obtidos no MATLAB® para a MRA e a FFT, considerando-se os mesmos sinais em análise.

ALGORITMOS IMPLEMENTADOS

FFT

Seja o esforço computacional da Transformada Discreta de Fourier (DFT) calculado por:

$$EC = N^2 \tag{1}$$

Em que N é o número de elementos do sinal em análise. Dessa forma, um vetor de 8 posições exigiria o cálculo de 64 multiplicações complexas. Caso o vetor de N posições seja dividido em duas partes, uma referindo-se aos índices pares e outra aos índices ímpares, a sua DFT do sinal pode ser escrita como [16]:

$$X_k = \sum_{n=0}^{N/2-1} x(2n) W_N^{kn} + \sum_{n=0}^{N/2-1} x(2n+1) W_N^{kn} \tag{2}$$

em que o fator W_N é chamado de fator de giro (*twiddle factor*) e é definido como:

$$W_N = e^{\frac{-j2\pi}{N}} = \cos\left(\frac{2\pi}{N}\right) - \text{jsen}\left(\frac{2\pi}{N}\right). \tag{3}$$

O cálculo do esforço computacional referente à operação descrita em (1) agora não se referirá ao comprimento original do vetor (N), mas a duas vezes a metade desse comprimento. Portanto, tem-se:

$$EC = 2\left(\frac{N}{2}\right)^2 = \frac{N^2}{2}. \tag{4}$$

Pode-se observar que a divisão da amostra em termos de índices pares e ímpares resulta em uma redução de 50% do esforço computacional para o cálculo de uma DFT. A ideia principal é que o processo continue até que se tenham apenas dois valores para o cômputo da DFT. Essa operação é chamada de decimação no tempo [16] e encontra-se ilustrada, para um melhor entendimento, na Fig. 1.

A decimação no tempo realiza a ordenação das amostras na entrada dos dados, no domínio do tempo. Caso tal ordenação seja realizada após o processamento da DFT, a mesma acontecerá no domínio da frequência, sendo denominada decimação na frequência. Para o presente estudo, foi adotada a decimação no tempo, implementada pela rotina chamada bit *reverse*, que se encontra exemplificada na Fig. 2. Basicamente, a operação de bit *reverse* consiste em representar os índices relativos às posições das amostras em um vetor em número binário e então fazer a inversão na ordem dos elementos, de modo que o número comece no bit menos significativo (LSB) e terminará no bit mais significativo (MSB).

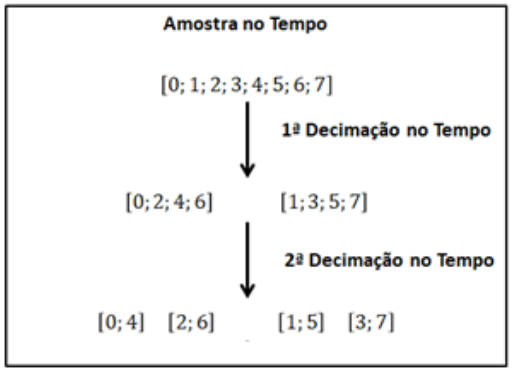


Fig. 1. Ilustração do processo de decimação no tempo.

Após a etapa de reordenamento das amostras, o próximo passo é prosseguir com a operação conhecida como *Butterfly*. O *Butterfly* usa a propriedade de simetria das amostras presentes na lógica da FFT [16], de modo que permite um alto ganho computacional. Na Fig. 3 é mostrado o esquema básico do processo.

Vetor	Binário
X[0]	000
X[1]	001
X[2]	010
X[3]	011
X[4]	100
X[5]	101
X[6]	110
X[7]	111

Bit Reverse

Binário	Vetor
000	X[0]
100	X[4]
010	X[2]
110	X[6]
001	X[1]
101	X[5]
011	X[3]
111	X[7]

Fig. 2. Reorganização de um vetor por *bit reverse*.

O algoritmo da FFT implementado no presente trabalho requer $\log_2(N)$ estágios de *Butterfly*; $\frac{N}{2}\log_2(N)$ multiplicações complexas e $N\log_2(N)$ adições complexas. [18].

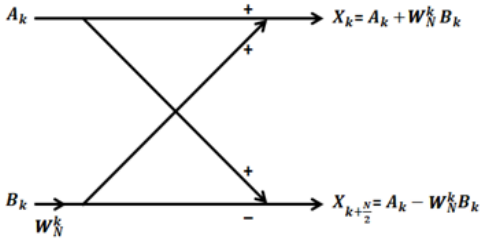


Fig. 3. Ilustração do esquema *Butterfly*.

O cálculo da DFT, conforme expresso em (2) contém produtos complexos redundantes, e tais repetições desses produtos podem ser eliminadas para produzir uma execução mais rápida [20]. A FFT é simplesmente um melhoramento que revolucionou o processamento digital do método para a computação da DFT [21]. Em 1965, os matemáticos Cooley e Tukey apresentaram a FFT [22], cuja principal ideia é fatorizar W_N^{kn} em um produto de matrizes esparsas que exigem um menor custo de implementação do que o processo da DFT [23].

Wavelet – Análise Multiresolução (MRA)

O algoritmo para o cálculo da MRA, também conhecido como algoritmo piramidal de Mallat, foi proposto em 1988 por Mallat [19]. É dividido em duas partes, a decomposição e reconstrução do sinal. O primeiro passo consiste em obter vetores com os coeficientes de aproximação e detalhes do sinal original na etapa de decomposição. Estes vetores são obtidos por convolução do sinal original com o filtro passa-baixa (Lo_D), para aproximações, e com o filtro passa-alta (Hi_D) para os detalhes. Em seguida, é executada a operação de *downsampling*. Uma decomposição de um nível de um sinal é ilustrada na Fig. 4. O processo de decomposição pode ser iterado, com aproximações sucessivas sendo decompostas por vez, de modo que um sinal é dividido em muitos componentes de resolução mais baixa [17].

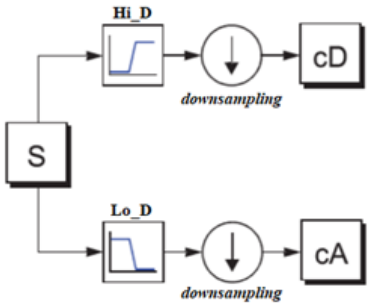


Fig. 4. Ilustração da decomposição do sinal em um nível – adaptado de [17].

Após a etapa de decomposição do sinal, é necessário que os detalhes e aproximações sejam reconstruídos de forma a apresentar o mesmo número de pontos do sinal original em análise. O processo de reconstrução consiste em realizar o *upsampling* do sinal e, em seguida, realizar a convolução desses valores com os filtros de reconstrução. Semelhante à decomposição do sinal, os coeficientes de aproximação participam da operação de convolução com o filtro passa-baixa (Lo_R), e os coeficientes de detalhes participam da convolução com o filtro passa-alta (Hi_R). O processo é ilustrado na Fig.5.

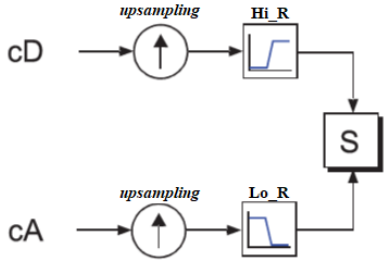
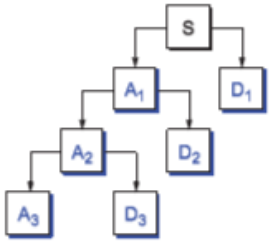


Fig. 5. Ilustração da reconstrução do sinal em um nível – adaptado de [17].

Na teoria de análise de multiresolução, um sinal original discreto S é decomposto em dois componentes A1 (aproximação de sinal) e D1 (detalhe de sinal), de modo que:

$$S = A_1 + D_1 . \quad (5)$$

Ao estender esse raciocínio para vários níveis de análise de multiresolução, observa-se que uma relação semelhante é válida para todos os componentes do sinal reconstruído. Assim, existem várias maneiras de reconstruir o sinal original [17], como mostrado na Fig. 6.



$$S = A_1 + D_1$$
$$= A_2 + D_2 + D_1$$
$$= A_3 + D_3 + D_2 + D_1$$

Fig. 6. Árvore de decomposição *wavelet* de três níveis de um sinal – adaptado de [17].

IMPLEMENTAÇÃO NO ANDROID

Breve descrição do App

Os algoritmos anteriormente descritos foram originalmente escritos em linguagem de programação C. Para assegurar que os códigos em linguagem C sejam escritos e compilados para *smartphones*, são utilizadas duas ferramentas de desenvolvimento: o Android *Studio* e o Android *Native Development Kit* (NDK). O Android Studio fornece um ambiente de desenvolvimento que abrange e incorpora a IDE IntelliJ IDEA, os *plug-ins* do Android SDK e um emulador. Já o NDK proporciona o suporte para incorporar códigos C/C++ no Android Studio.

Para esta aplicação, uma interface visual básica foi desenvolvida, como mostrado na Fig. 7. O usuário pode selecionar o arquivo de texto com as informações de sinal que deseja processar, e então um arquivo de texto contendo o resultado do cálculo da FFT ou da MRA é disponibilizado. Para a FFT o arquivo de saída contém informações de amplitude e frequência, separados em duas colunas. Já para a MRA, o arquivo disponibiliza os coeficientes resultantes para as aproximações e os detalhes, separados em colunas, de acordo com o nível de detalhamento escolhido. Além disso, as entradas da aplicação não foram restringidas pelo número de elementos e a MRA é capaz de executar qualquer nível de decomposição permitido, com base no comprimento do vetor de amostras, bem como de utilizar qualquer família *wavelet* disponível e aplicável ao processo. O aplicativo foi projetado para ser executado em qualquer *smartphone* que tenha a versão do Android 4.4 ou superior, garantindo sua execução em vários dispositivos.

Resultados

Para a avaliação dos algoritmos desenvolvidos, foi feita uma comparação entre os resultados apresentados por eles rodando em um *smartphone* e os resultados do MATLAB® para um mesmo sinal em ambos os casos (FFT e MRA). O aplicativo foi instalado e testado em um celular da marca Samsung®, modelo S3 Duos, com 1GB de memória RAM e com processador Qualcomm Snapdragon S4 Play MSM8225 Dual-Core ARM Cortex-A5, de 1.2GHz de *clock*. O *software* utilizado para comparação, MATLAB® - versão 2015B, foi executado em um *notebook* equipado com um processador Intel Core i7-4500U, com *clock* de 1.8GHz de dois núcleos, com memória RAM de 8GB.

Um sinal real contendo 1.048.576 pontos foi utilizado para realizar a comparação entre o desempenho da FFT apresentado na aplicação desenvolvida e no software MATLAB®. Os resultados obtidos nesta comparação podem ser observados na Fig. 8 e na Fig. 9. Para esta situação de comparação, o erro médio foi de 1,0915 x 10^{-11} e o valor de pico do erro foi de 3,0851 x 10^{-8} e foi considerado satisfatório para possíveis aplicações, como cálculo de valor de frequência fundamental para estimação de velocidade de máquinas rotativas. O tempo de execução do processo foi de aproximadamente 45s no MATLAB® e de 1 min 07s no *smartphone*.

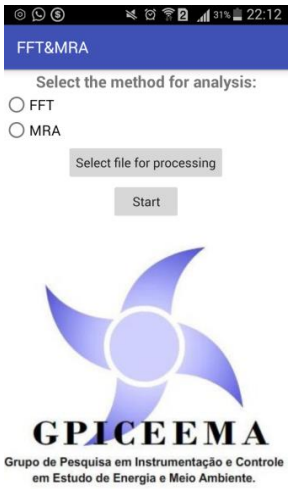


Fig. 7. Interface do *app*.

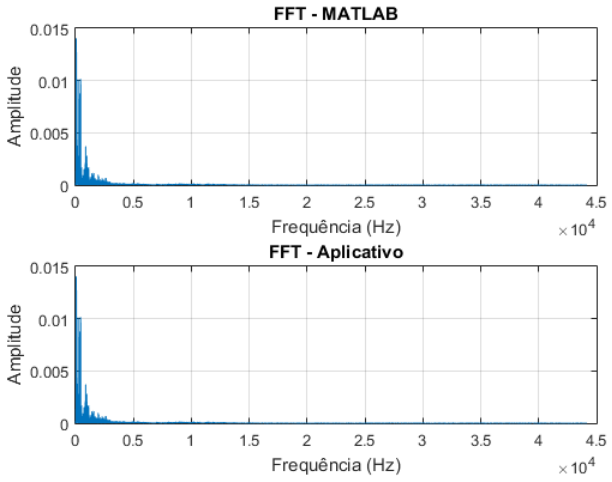


Fig. 8. Resultado da FFT de um Sinal Real – Aplicativo vs MATLAB®.

No caso do algoritmo da MRA, o comportamento do software em situações de decomposição e reconstrução do sinal foi avaliado. Inicialmente, o sinal em análise foi decomposto em uma aproximação e sete detalhes, com aplicação da *wavelet* db8, de modo que o vetor de coeficientes é da forma:

$$C = \left[CA_7\, CD_7\, CD_6\, CD_5\, CD_4\, CD_3\, CD_2\, CD_1\right] \qquad (6)$$

em que CAj são os coeficientes de aproximação no nível j, e CDj são os coeficientes de detalhe no nível j. Na Fig. 10 são mostrados os coeficientes de decomposição do MATLAB®, bem como os coeficientes de decomposição do *app*, na Fig.11. O erro médio para esta situação apresentou um valor de 4,3208 x 10^{-20}. Já o valor de pico do erro foi de 1,4876 x 10^{-15}, conforme pode ser observado na Fig. 12. Para esse caso, o tempo de execução no MATLAB® foi de 1 min 06s, enquanto que no *smartphone* foi de 1 min 39s.

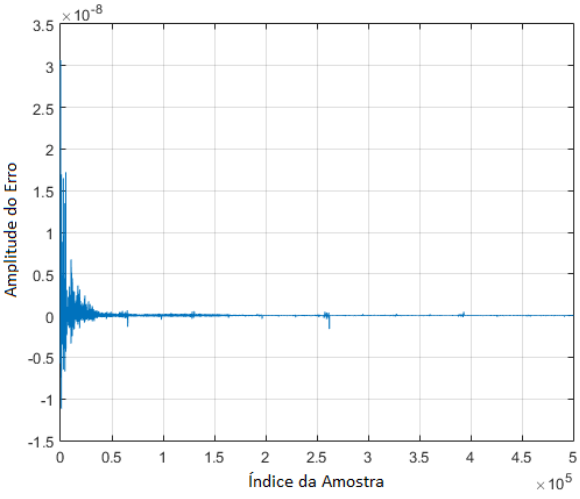


Fig. 9. Erro entre resultados do aplicativo e do MATLAB® - FFT.

Com os resultados da etapa de decomposição, os coeficientes de reconstrução foram obtidos. Na Fig. 13 são mostrados os coeficientes de reconstrução obtidos no MATLAB® e, na Fig. 14, os resultados obtidos no aplicativo. Analogamente ao anteriormente realizado, os resultados obtidos na etapa de reconstrução do sinal foram comparados, como visto na Fig. 15. O valor de pico do erro para essa situação foi 2,7103 x 10^{-16}, já o erro médio teve um valor de 1,2879 x 10^{-19}. Quanto ao tempo de execução, observou-se que no MATLAB® o tempo decorrido foi de 1 min 17s, enquanto que no *smartphone* foi de 1 min 58s.

CONCLUSÕES

À medida que a capacidade computacional e o poder de processamento de dispositivos móveis aumentam, a viabilidade do seu uso como ferramenta científica se expande, principalmente no campo do processamento de sinais. Este trabalho apresentou os resultados da implementação da FFT e da MRA como aplicativo *mobile* e sua comparação com o MATLAB®, um *software* amplamente aceito nos meios científico e acadêmico. Os resultados foram considerados satisfatórios, com diferenças da ordem de 10^{-11} a 10^{-20} para o valor médio do erro. Sobre o tempo de execução, verificou-se que o desempenho do smartphone foi inferior, com uma diferença média da ordem de 30s. Contudo, o desempenho foi considerado adequado, dadas as configurações do aparelho em uso. Espera-se que a aplicação desenvolvida seja utilizada como base para o desenvolvimento de sistemas mais aprimorados, como sistemas de diagnóstico portáteis e de baixo custo, levando-se em conta o fato de que ferramentas de desenvolvimento de *software* para *smartphones* são gratuitas e encontram-se bem desenvolvidas.

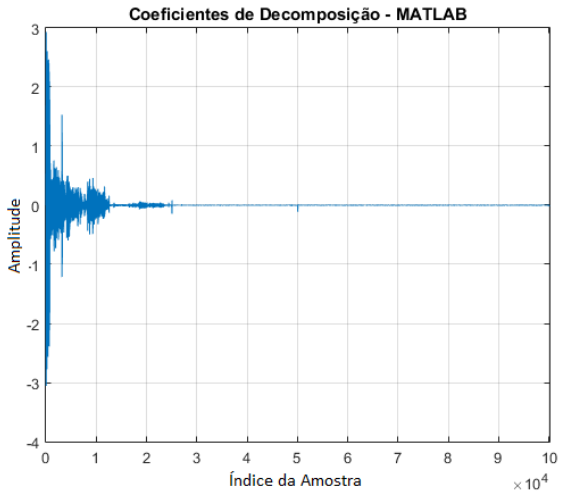


Fig. 10. Coeficientes de decomposição *wavelet* – MATLAB®.

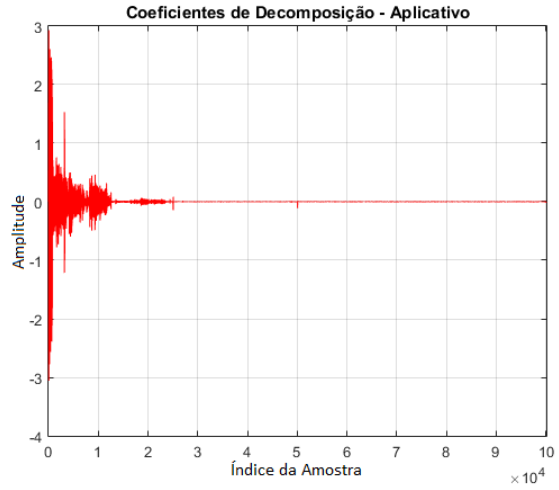


Fig. 11. Coeficientes de decomposição *wavelet* – Aplicativo

AGRADECIMENTOS

Os autores gostariam de agradecer ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), pelo apoio financeiro, e aos membros do GPICEEMA.

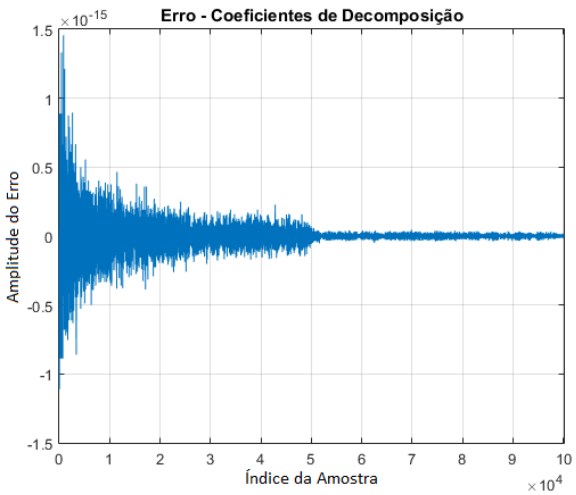


Fig. 12. Erro entre resultados do Aplicativo e do MATLAB® – Decomposição *wavelet*.

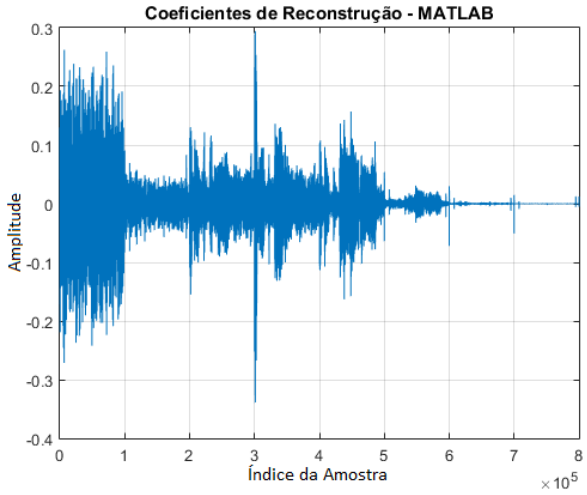


Fig. 13. Coeficientes de reconstrução *wavelet* – MATLAB®.

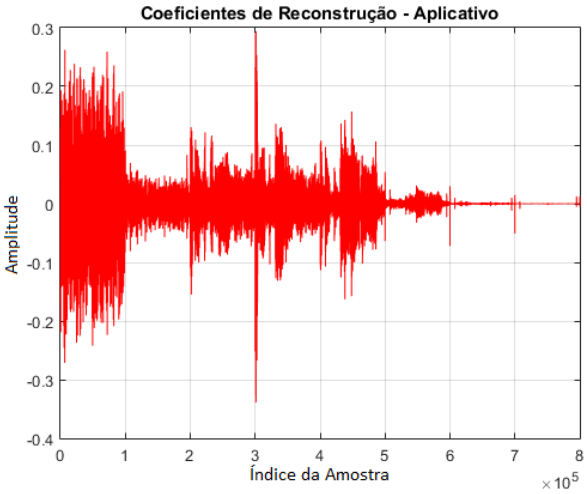


Fig. 14. Coeficientes de reconstrução *wavelet* – Aplicativo.

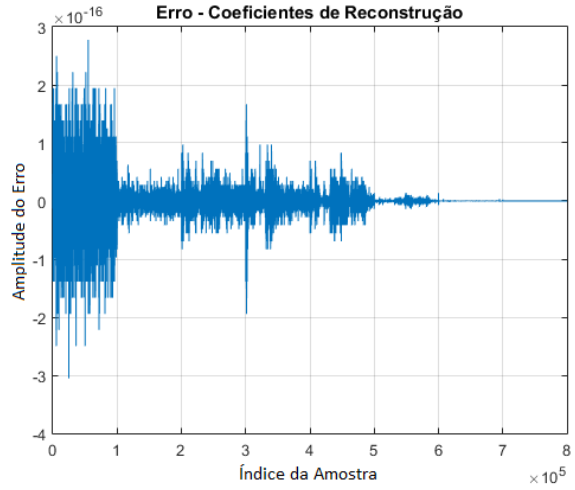


Fig. 15. Erro entre resultados do Aplicativo e do MATLAB® – Decomposição *wavelet*.

REFERÊNCIAS

[1] D. F. A. Santiago e R. Pederiva, "Diagnosis Of Mechanical Faults By Wavelet Transform" III Congreso Bolivariano de Inegeniería Mecánica– III COBIM, 2003.

[2] C. S. Burrus, R. A. Gopinath, H. Guo, J. E. Odegard and I. W. Selesnick, "Introduction to wavelets and wavelet transforms: a primer", New Jersey: Prentice Hall, vol. 1, 1998.

[3] A. Graps, "An introduction to wavelets", IEEE computational science and engineering, vol. 2, issue 2, pp. 50-61, 1995.

[4] S. Mallat, "A Wavelet Tour of Signal Processing", 3rd ed., The Sparse Way. Academic Press, 2008.

[5] E. O. Brigham, "The fast Fourier transform and its applications", Englewood Cliffs, NJ: Prentice Hall, vol. 1, 1988.

[6] B. S. Reddy e B. N. Chatterji, "An FFT-based technique for translation, rotation, and scale-invariant image registration", IEEE transactions on image processing, vol. 5, issue 8, pp. 1266-1271, 1996.

[7] "Mobile Fact Sheet" Internet: http://www.pewinternet.org/fact-sheet/mobile/, Feb. 5, 2018 [Feb. 27, 2018].

[8] "Number of mobile phone users worldwide from 2013 to 2019" Internet: https://www.statista.com/statistics/274774/forecast-of-mobile-phone-users-worldwide/, 2018 [Feb. 27, 2018].

[9] A. Dogtiev. "App Download and Usage Statistics 2017" Internet: http://www.businessofapps.com/data/app-statistics/, Jan. 9, 2018 [Feb. 27, 2018].

[10] B. C. Florea, "Smartphone input/output interface for IoT applications" Proc. of the 25th Telecommunication Forum (TELFOR), Sérvia, 2017, pp. 1-4.

[11] A. U. Bokade and V. R. Ratnaparkhe, "Video surveillance robot control using smartphone and Raspberry pi," Proc. of the 2016 International Conference on Communication and Signal Processing (ICCSP), Melmaruvathur, 2016, pp. 2094-2097.

[12] N. Yodpijit, M. Jongprasithporn, S. Saengmanee, P. Suriyapen, W. Imsamran, A. Chaiwerawattana, S. Younghorkijphaisarn; P. Choosanit, "The Thai Arthrometric Navigator (TAN) scale smartphone application for cancer patients", Proc. of the 4th International Conference on Control, Automation and Robotics (ICCAR), Auckland, 2018, pp. 509-513.

[13] Y. Fu and J. Guo, "Blood Cholesterol Monitoring With Smartphone as Miniaturized Electrochemical Analyzer for Cardiovascular Disease Prevention", IEEE Transactions on Biomedical Circuits and Systems, 2018, pp. 1-7.

[14] J. Grebenik, C. Bingham, S.Srivastava, "Continuous Acoustic Monitoring of Electrical Machines; Processing Signals from USB Microphone & Mobile Smartphone Sensors Detecting DC Motor Controller Fault", Proc. of the 5th International Conference on Control, Decision and Information Technologies (CoDIT), Thessaloniki, 2018, pp. 677-682.

[15] S. Parris, M. Torlak, and N. Kehtarnavaz, "Real-timeimplementation of cochlear implant speech processing pipeline on smartphones," Proc. of IEEE International Conference Engineering in Medicine and Biology (EMBC), Chicago, 2014, pp. 886-889.

[16] W. H. Press, B. P. Flannery, S. A. Teukolsky and W. T. Vetterling, "Numerical Recipes in C: The art of scientific computing", Cambridge University Press: New York, 1988.

[17] M. Misiti, Y. Misiti, G. Oppenheim and J.-M. Poggi, "Waveet Tollbox™ 4 – User's Guide", The MathWorks, 2009.

[18] M. McKeon, "FFT Implementation on the TMS320VC5505, TMS320C5505 and TMS320C5515 DSPs", Application Report SPRABB6B, Texas Instruments, 2013.

[19] S. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 11, issue 7, pp. 674–693, 1989.

[20] S. D. Steams e D. R. Hush, "Digital Signal Processing with examples in Matlab", CRC Press: Boca Raton, 2011.

[21] G. D. Bergland, "A guided tour of the Fast Fourier Transform", IEEE Spectrum, vol. 6, nº7, pp. 41-52, 1969.

[22] J. W. Cooley e J. W. Tukey, "An algorithm for the machine calculation of complex Fourier Series", Mathematics of Computation, vol. 19, nº 90, pp. 297-301, 1965.

[23] A. Mertins, "Signal Analysis: Wavelets, Filter Banks, Time-Frequency Transforms and Applications", 1ª edição, John Wiley & Sons: UK, 1999.