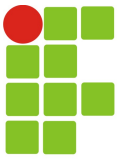


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

Análise tempo-escala de séries temporais com falhas tratadas com interpolação numérica

Paula Neves de Araújo

IFSP
São Paulo
2016

Análise tempo-escala de séries temporais com falhas tratadas com interpolação numérica

Paula Neves de Araújo

Trabalho de Conclusão do Curso Superior de Licenciatura em Matemática, orientado pelo Prof. Me. Luciano Aparecido Magrini

IFSP
São Paulo
2016

A662a Araújo, Paula Neves de

Análise tempo-escala de séries temporais com falhas tratadas com interpolação numérica / Paula Neves de Araújo.

São Paulo: [s.n.], 2017.

116 f.

Orientador: prof. Me. Luciano Aparecido Magrini

Monografia (Licenciatura em matemática) - Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, IFSP, 2017.

1. Séries temporais 2. Análise de wavelets 3. Interpolação numérica I. Instituto Federal de Educação, Ciência e Tecnologia de São Paulo II Título

CDU 510

PAULA NEVES DE ARAÚJO

ANÁLISE TEMPO-ESCALA DE SÉRIES TEMPORAIS COM FALHAS TRATADAS COM INTERPOLAÇÃO NUMÉRICA

Monografia apresentada ao Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, em cumprimento ao requisito exigido para a obtenção do grau acadêmico de Licenciada em Matemática.

APROVADA EM 05/12/2016 **CONCEITO:** 10,0

Mariana P. M. A. Baroni.

Profª. Drª. Mariana P. M. A. Baroni
Membro da Banca

Prof. Dr. Marco Aurélio Granero Santos
Membro da Banca

Prof. Me. Luciano Aparecido Magrini
Orientador

Paula Neves de Araújo

Aluna: Paula Neves de Araújo

"A Matemática é a mais simples, a mais perfeita e a mais antiga de todas as ciências".
Jacques Hadamard

Aos meus pais, a quem tanto amo, à Patrícia, que jamais permite que eu me sinta só, e ao Gustavo, que torna melhores os meus dias. Vocês são a melhor parte de mim.

Agradecimentos

Primeiramente, agradeço a Deus pela vida, por quem caminha comigo e pela oportunidade de fazer o que amo ao longo dos últimos anos.

Agradeço aos meus pais, Monteiro e Iva, por seu amor e apoio no decorrer de todo o curso e em especial durante a realização deste trabalho. Eu os amo mais do que posso dizer.

À Patrícia, minha irmã, companheira de toda a vida, por estar ao meu lado nos momentos bons e ruins, me acompanhando nesta jornada. Eu não conseguiria sem você.

Ao Gustavo, pela paciência, compreensão e companhia durante os últimos quatro anos. Você torna as coisas mais fáceis e faz meus dias mais bonitos.

Ao prof. Me. Luciano Aparecido Magrini, com quem tive o prazer de trabalhar, pela confiança, disponibilidade e paciência, por compartilhar conhecimento comigo e me orientar na realização desta pesquisa.

Ao prof. Dr. Rogério Ferreira da Fonseca, ao prof. Me. Wellington Pereira das Virgens e à profa. Dra. Alessandra Ferreira Ignez, pelas contribuições na realização deste trabalho.

À profa. Dra. Flávia Milo dos Santos e ao prof. Dr. Henrique Marins de Carvalho, por sua gentileza, seu apoio e sua disposição em sempre me ajudar no decorrer do curso.

Ao prof. Dr. Marco Aurélio Granero Santos e à profa. Dra. Mariana Pelissari Monteiro Aguiar Baroni pela disponibilidade em avaliar este trabalho.

A todos os professores que contribuíram com a minha formação e aos amigos e colegas que me ajudaram de tantas formas, com as disciplinas ou com sua companhia, tornando mais leves os dias difíceis.

Resumo

O presente trabalho tem como objetivo verificar a aplicabilidade de cinco métodos interpolatórios no tratamento de falhas presentes em séries temporais e o quanto cada um destes métodos interfere na análise tempo-escala dos dados. O estudo de possíveis aperfeiçoamentos na análise de séries temporais com falhas, conhecidas também como sinais, constitui um relevante objeto de pesquisa, uma vez que possui aplicações em diversas áreas científicas. Os procedimentos metodológicos para a obtenção dos resultados foram a revisão bibliográfica da teoria da Transformada *Wavelet* Contínua, ferramenta utilizada para análise de séries, e o estudo dos seguintes métodos de interpolação, que utilizamos nesta pesquisa como tentativas para solucionar as falhas dos sinais: interpolação linear, polinômio cúbico de Hermite, *splines* cúbicos, curva de Bézier e polinômio de Chebyshev. Realizamos também testes envolvendo séries com falhas utilizando o *software* MATLAB. Como resultado, obtivemos dados que, embora não permitam definir qual método de interpolação é o mais aplicável em todos os casos possíveis, oferecem uma estimativa das vantagens e desvantagens de cada método analisado.

Palavras-chave: Séries temporais, Análise de *wavelets*, Interpolação numérica.

Abstract

The present work aims to verify the applicability of five interpolatory methods in the treatment of gaps present in time series and how much each one of these methods interferes in the time-scale analysis of the data. The study of possible improvements in the time series with gaps analysis is a relevant research object, since it has applications in several scientific areas. The methodological procedures for obtaining the results were the literature review, a continuous wavelet transform, which was the tool used for series analysis and the study of the interpolation methods, which we use in this research as attempts to solve the gaps: linear interpolation, Hermite cubic polynomial, cubic splines, Bézier curve and Chebyshev polynomial. We also perform tests involving series with gaps using MATLAB software. As a result, we obtained data that, while not allowing us to interpolation is the most applicable in all possible cases, provide an estimate of the advantages of each method analyzed.

Keywords: Time series, *Wavelet* Analysis, Interpolation

Lista de figuras

Lista de tabelas

Sumário

1 Introdução

Segundo Morettin, “uma série temporal é qualquer conjunto de observações ordenadas no tempo” (MORETTIN; TOLOI, 1981, p. 1). As séries temporais podem ser utilizadas para representar sinais, que podem ser definidos, de acordo com Weeks, como fenômenos que variam e, em geral, tratam-se de quantidades físicas que se alteram de acordo com o tempo ou ainda com outros parâmetros, como o espaço. Ainda de acordo com o autor, entre as principais aplicações destes objetos matemáticos, podemos citar o processamento de sons e imagens (WEEKS, 2012, p. 10).

De acordo com Restivo, uma das primeiras ferramentas utilizadas para a análise de sinais foi a Transformada de Fourier (TF) (RESTIVO, 2011, p. 7), que pode ser aplicada de modo a transformar um sinal do domínio do tempo para o domínio da frequência. No entanto, de acordo com Polikar, a TF não permite a preservação da informação temporal do sinal, dificultando a análise de sinais não-estacionários, ou seja, cujas componentes frequenciais variam no decorrer do tempo. Por esse motivo, a Transformada *Wavelet* Contínua é frequentemente utilizada para essa finalidade (POLIKAR, 1999).

Em muitos casos, os dados coletados apresentam falhas, lacunas nas quais não se obtém informações sobre o comportamento do fenômeno observado. Nestes casos, a análise deve envolver ferramentas especiais e, nesta pesquisa, utilizamos a interpolação numérica para realizar estimativas acerca do comportamento dos fenômenos nos intervalos de tempo em que não se pôde obter dados precisos.

Tendo em vista a variedade de ferramentas matemáticas envolvidas na análise tempo-escala de séries temporais com falhas, como o cálculo diferencial e integral e a interpolação numérica, este é um estudo que pode proporcionar ao futuro professor de matemática uma visão mais ampla de conteúdos que, muitas vezes, são vistos como descontextualizados e excluídos de nosso cotidiano.

A principal questão a ser abordada nessa pesquisa é a forma como as interpolações realizadas afetam o resultado final da análise das séries. Procuramos também verificar se é possível estimar o melhor método de interpolação a ser utilizado nesses procedimentos. Para isso, realizamos uma pesquisa bibliográfica-experimental, com abordagem exploratória, a partir da revisão bibliográfica acerca da Transformada de Fourier, da Transformada *Wavelet* Contínua e dos métodos de interpolação, e testes de aplicação destes métodos em séries temporais com falhas.

No segundo capítulo, abordamos a Transformada de Fourier, suas propriedades, história e exemplos. Ainda neste capítulo, explicamos o que são funções *wavelets*, suas propriedades, o funcionamento da Transformada *Wavelet* Contínua, como são representados

os resultados obtidos por meio desse tipo de análise e trazemos também um panorama histórico do desenvolvimento das *wavelets* no decorrer do tempo.

No terceiro capítulo do trabalho, realizamos uma breve introdução à teoria das *interpolações numéricas*, que são algumas das ferramentas matemáticas que podem ser utilizadas para tratar as falhas encontradas nos dados. Aqui, utilizamos cinco tipos de interpolação: o polinômio cúbico de Hermite, os *splines* cúbicos, a curva de Bézier, o polinômio de Chebyshev e a interpolação linear.

Em seguida, no quarto capítulo, realizamos testes com estes cinco tipos de interpolação apresentando, inclusive, os procedimentos adotados para obtenção dos resultados, e verificamos de que forma o tipo de interpolação escolhido afeta os resultados finais, comparando os escalogramas e espectros globais obtidos.

Por fim, no quinto capítulo, trazemos as considerações finais sobre os resultados apresentados nos capítulos anteriores, verificamos se os mesmos oferecem subsídios para responder à nossa questão inicial, e fazemos apontamentos para continuação desta pesquisa.

2 De Fourier às *Wavelets*: conceitos fundamentais

Neste capítulo, expomos brevemente a história da análise de Fourier e de como esta contribuiu com o desenvolvimento da Transformada *Wavelet* Contínua, ferramenta que utilizaremos para a análise dos sinais com falhas. Abordamos a definição e algumas propriedades relacionadas à Transformada de Fourier, conforme os enunciados encontrados em Souza (2009), e trazemos suas demonstrações. Descrevemos os elementos relacionados à Transformada *Wavelet* Contínua e também apresentamos alguns exemplos do uso desta ferramenta. Por fim, trazemos o desenvolvimento histórico da análise de *wavelets* e uma breve abordagem do *software* MATLAB, utilizado em nosso trabalho.

2.1 Um pouco de história

Nesta seção, realizamos uma breve abordagem histórica da análise de Fourier, com base no livro *Introdução à História da Matemática* (EVES, 1995), onde é possível encontrar mais informações sobre o assunto.

Jean Baptiste Joseph Fourier (1768-1830) foi um matemático francês, considerado um dos mais influentes do século XIX. Seus estudos sobre o calor tiveram início em 1801, em Grenoble, e resultaram em um trabalho bastante relevante, enviado pelo pesquisador para aprovação da Academia de Ciências da França, em 1807. Seu foco era o estudo da propagação do calor em barras, sólidos e superfícies, e seu trabalho resultou em um artigo que, entre outras informações, afirmava que todas as funções definidas no intervalo $(-\pi, \pi)$ poderiam ser escritas como somas de funções seno e cosseno. Assim, dada uma função f em condições adequadas, seria possível escrever:

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty}(a_n cos(nx) + b_n sen(nx))$$

A série acima era chamada de *série trigonométrica*, e a equação dada já havia sido verificada para algumas funções. No entanto, a afirmação de Fourier foi considerada incorreta por ser muito ampla e, assim, acabou rejeitada pela Academia, de modo que o trabalho do matemático também não foi aceito. Apesar disso, uma versão revisada deste mesmo trabalho foi enviada à Academia anos depois, e Fourier obteve um prêmio oferecido a pesquisas sobre propagação de calor pela instituição. Ainda assim, seu trabalho não foi recomendado para a publicação nas *Mémoires* da Academia, sob a justificativa de ausência de rigor matemático.

Fourier prosseguiu então em suas pesquisas e, em 1822, desenvolveu seu trabalho mais influente: *Théorie Analytique de la Chaleur* (ou Teoria Analítica do Calor), considerado um dos grandes clássicos da matemática. A partir daí, a representação proposta por Fourier para as funções definidas no intervalo $(-\pi, \pi)$, também chamada de *série de Fourier*, passou a ser utilizada em diversas áreas da matemática e da física, tendo em vista a variedade das funções que podem ser representadas dessa forma, embora nem todas apresentem essa possibilidade.

Apesar de sua grande utilidade para diversas finalidades, as séries de Fourier só podem ser utilizadas para representar funções periódicas. No caso das funções não-periódicas, se as interpretarmos como *funções de período infinito*, chegamos à construção de uma ferramenta que permite que representemos tais funções de forma semelhante ao que ocorre com as séries de Fourier. Essa ferramenta, chamada de *Transformada de Fourier*, é o assunto da próxima seção, elaborada com base na apostila de Souza (2009).

2.2 Introdução à Transformada de Fourier

Consideremos um sinal contínuo $x(t) \in \mathbb{C}$. Embora o nosso estudo seja voltado para os sinais reais, é possível operar com sinais complexos de todos os tipos. Assim, por definição, a *Transformada de Fourier* do sinal $x(t)$ é dada por:

$$X(i\omega) = \int_{-\infty}^{\infty} x(t) \cdot e^{-i\omega t} dt \tag{2.1}$$

Essa Transformada permite expressar o sinal $x(t)$ da seguinte forma:

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(i\omega) \cdot e^{i\omega t} d\omega \tag{2.2}$$

A Transformada de Fourier possui propriedades que podem ser demonstradas a partir de ferramentas do cálculo diferencial e integral e com o uso da expressão obtida anteriormente, conhecida como *Transformada inversa de Fourier*. Algumas delas serão enunciadas e demonstradas a seguir; a partir dessas, outras podem ser desenvolvidas também.

Linearidade. *Se x_1 e x_2 são sinais contínuos e $y(t) = \alpha x_1(t) + \beta x_2(t)$, então:*

$$Y(i\omega) = \alpha X_1(\imath\omega) + \beta X_2(i\omega)$$

Demonstração. Por definição, a Transformada de Fourier de $y(t)$ é dada por:

$$Y(i\omega) = \int_{-\infty}^{\infty} y(t) \cdot e^{(-i\omega t)} dt \tag{2.3}$$

Assim, substituindo $y(t)$ por $\alpha x_1(t) + \beta x_2(t)$ em (2.3), obtemos:

$$Y(i\omega) = \int_{-\infty}^{\infty} (\alpha x_1(t) + \beta x_2(t)) \cdot e^{(-i\omega t)} dt \tag{2.4}$$

Dada a linearidade da operação de integração, separamos a integral obtida em (2.4) em duas partes, da seguinte forma:

$$Y(i\omega) = \int_{-\infty}^{\infty} \alpha x_1(t) \cdot e^{(-i\omega t)} dt + \int_{-\infty}^{\infty} \beta x_2(t) \cdot e^{(-i\omega t)} dt \tag{2.5}$$

Como α e β são constantes, podemos reescrever a equação (2.5) como:

$$Y(i\omega) = \alpha \int_{-\infty}^{\infty} x_1(t) \cdot e^{(-i\omega t)} dt + \beta \int_{-\infty}^{\infty} x_2(t) \cdot e^{(-i\omega t)} dt$$

Por fim, da definição dada em (2.3), temos:

$$Y(i\omega) = \alpha X_1(i\omega) + \beta X_2(i\omega)$$

□

Conjugação. *Se $x(t)$ é um sinal periódico com período T e $y(t) = x^*(t)$, ou seja, $y(t)$ é o complexo conjugado de $x(t)$, então:*

$$Y(i\omega) = X^*(-i\omega)$$

Demonstração. Pela definição, escrevemos:

$$Y(i\omega) = \int_{-\infty}^{\infty} x^*(t) \cdot e^{-i\omega t} dt \tag{2.6}$$

Tomando o complexo conjugado de ambos os membros da equação (2.6), obtemos:

$$Y^*(i\omega) = \int_{\infty}^{\infty} x(t) \cdot e^{j\omega t} dt \tag{2.7}$$

Se reescrevermos (2.7) da seguinte forma:

$$Y^*(i\omega) = \int_{\infty}^{\infty} x(t) \cdot e^{-(-i\omega t)} dt \tag{2.8}$$

obtemos no segundo membro da equação, pela definição, $X(-i\omega)$. Assim, em (2.8), temos:

$$Y^*(i\omega) = X(-i\omega) \tag{2.9}$$

Por fim, tomando o conjugado de ambos os membros de (2.9), concluímos que:

$$Y(i\omega) = X^*(-i\omega)$$

□

Sinal refletido/Reflexão no tempo ("time reversal"). *Se $x(t)$ é um sinal e $y(t) = x(-t)$, então:*

$$Y(i\omega) = X(-i\omega)$$

Demonstração. Pela definição da Transformada de Fourier, temos:

$$Y(i\omega) = \int_{-\infty}^{\infty} x(-t) \cdot e^{-i\omega t} dt \tag{2.10}$$

Dessa maneira, faz-se uma mudança de variável, da seguinte forma:

$$u = -t$$

$$du = -dt$$

Para os limites de integração, temos:

$$t \to -\infty \Rightarrow u \to \infty$$

$$t \to \infty \Rightarrow u \to -\infty$$

Assim, reescrevemos (2.10) como segue:

$$Y(i\omega) = \int_{\infty}^{-\infty} x(u) \cdot e^{-i\omega(-u)}(-du)$$

ou ainda:

$$Y(i\omega) = -\int_{-\infty}^{\infty} -x(u) \cdot e^{-(-i\omega)u} du$$

Como $-x(u) = (-1)[x(u)]$, podemos escrever:

$$Y(i\omega) = -(-1)\int_{-\infty}^{\infty} x(u) \cdot e^{-(-i\omega)u} du = \int_{-\infty}^{\infty} x(u) \cdot e^{-(-i\omega)u} du$$

Portanto, da equação anterior, concluímos que:

$$Y(i\omega) = X(-i\omega)$$

□

Translação no tempo ("time shifting"). *Se $x(t)$ é um sinal contínuo e $y(t)$ é o sinal $x(t)$ com uma translação de t_0 no tempo, ou seja, $y(t) = x(t - t_0)$, então:*

$$Y(i\omega) = e^{-i\omega t_0} X(i\omega)$$

Demonstração. A Transformada de Fourier do sinal $y(t)$ é definida como:

$$Y(i\omega) = \int_{-\infty}^{\infty} x(t - t_0) \cdot e^{-i\omega t} dt \tag{2.11}$$

A partir daí, realizamos uma mudança de variável da seguinte forma:

$$u = t - t_0$$

$$du = dt$$

Os limites de integração não se alteram, uma vez que:

$$t \to -\infty \Rightarrow u \to -\infty$$

$$t \to \infty \Rightarrow u \to \infty$$

Aplicando essa mudança de variável na equação (2.11), obtemos:

$$Y(i\omega) = \int_{-\infty}^{\infty} x(u) \cdot e^{-i\omega(u+t_0)} du \tag{2.12}$$

Podemos desenvolver a expressão obtida em (2.12) como segue:

$$Y(i\omega) = \int_{-\infty}^{\infty} x(u) \cdot e^{-i\omega u} \cdot e^{-i\omega t_0} du$$

O fator $e^{-i\omega t_0}$ é constante em relação a u, portanto:

$$Y(i\omega) = e^{-i\omega t_0} \int_{-\infty}^{\infty} x(u) \cdot e^{-i\omega u} du = e^{-i\omega t_0} \cdot X(i\omega)$$

□

Derivadas. *Se $x(t)$ é um sinal e $y(t) = \frac{dx}{dt}(t)$, então $Y(i\omega) = i\omega X(i\omega)$.*

Demonstração. Pela definição da Transformada Inversa de Fourier, podemos escrever o sinal $x(t)$ da seguinte forma:

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(i\omega) \cdot e^{i\omega t} d\omega \tag{2.13}$$

Se tomarmos a derivada em relação a t de ambos os membros da equação (2.13), teremos:

$$\frac{dx}{dt}(t) = \frac{d}{dt}\left(\frac{1}{2\pi} \int_{-\infty}^{\infty} X(i\omega) \cdot e^{i\omega t} d\omega \right) \tag{2.14}$$

Pelo fato de que $\frac{dx}{dt}(t) = y(t)$ e, além disso, podermos derivar $X(i\omega) \cdot e^{i\omega t}$ em relação a t, devido às propriedades da integração, em (2.14), teremos:

$$y(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \frac{d}{dt}(X(i\omega) \cdot e^{i\omega t}) d\omega \tag{2.15}$$

Como $X(i\omega)$ é constante em relação a t, podemos escrever:

$$y(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(i\omega) \cdot \frac{d}{dt} e^{i\omega t} d\omega$$

Podemos observar que:

$$\frac{d}{dt} e^{i\omega t} = i\omega \cdot e^{i\omega t}$$

Logo, na equação (2.15), temos:

$$y(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} i\omega X(i\omega) \cdot e^{i\omega t} d\omega \tag{2.16}$$

Uma vez que o segundo membro da equação (2.16) corresponde à Transformada inversa de Fourier da função $i\omega \cdot X(i\omega)$, se tomarmos a Transformada de Fourier em ambos os membros dessa equação, concluiremos que:

$$Y(i\omega) = i\omega \cdot X(i\omega)$$

□

Escalonamento no tempo ("time scalling"). *Se $x(t)$ é um sinal e $y(t) = x(\alpha t)$, então:*

$$Y(i\omega) = \frac{1}{|\alpha|} X\left(\frac{i\omega}{\alpha}\right)$$

Demonstração. Seja $y(t) = x(\alpha t)$. Então, pela definição da Transformada de Fourier, temos:

$$Y(i\omega) = \int_{-\infty}^{\infty} x(\alpha t) \cdot e^{-i\omega t} dt \tag{2.17}$$

Vamos analisar dois casos possíveis: $\alpha > 0$ ou $\alpha < 0$.

No primeiro caso, para desenvolver a integral obtida em (2.17), realizamos a seguinte mudança de variável:

$$\alpha t = u$$

$$\alpha dt = du \Rightarrow dt = \frac{du}{\alpha}$$

Para os limites de integração, temos:

$$t \to -\infty \Rightarrow u \to -\infty$$

$$t \to \infty \Rightarrow u \to \infty$$

Aplicando essa mudança de variável na equação (2.17), obtemos:

$$Y(i\omega) = \int_{-\infty}^{\infty} x(u) \cdot \frac{1}{\alpha} \cdot e^{-i\omega\frac{u}{\alpha}} du \tag{2.18}$$

Como $\frac{1}{\alpha}$ é constante em relação a u, podemos reescrever a equação (2.18) como segue:

$$Y(i\omega) = \frac{1}{\alpha} \int_{-\infty}^{\infty} x(u) \cdot e^{\frac{-i\omega}{\alpha}u} du$$

Por definição, $\int_{-\infty}^{\infty} x(u) \cdot e^{\frac{-i\omega}{\alpha}u} du = X\left(\frac{i\omega}{\alpha}\right)$, então:

$$Y(i\omega) = \frac{1}{\alpha} X\left(\frac{i\omega}{\alpha}\right)$$

Como $\alpha > 0$, podemos escrever:

$$Y(i\omega) = \left|\frac{1}{\alpha}\right| X\left(\frac{i\omega}{\alpha}\right)$$

No segundo caso ($\alpha < 0$), se $y(t) = x(\alpha t)$ e $\alpha < 0$, operamos uma mudança de variável:

$$\alpha t = u$$

$$\alpha dt = du \Rightarrow dt = \frac{du}{\alpha}$$

Para os limites de integração, obtemos:

$$t \to -\infty \Rightarrow u \to \infty$$

$$t \to \infty \Rightarrow u \to -\infty$$

Então, utilizando essa mudança de variável na equação (2.17), chegamos à seguinte igualdade:

$$Y(i\omega) = \int_{\infty}^{-\infty} x(u) \frac{1}{\alpha} \cdot e^{-i\omega\frac{u}{\alpha}} du$$

ou ainda:

$$Y(i\omega) = -\int_{-\infty}^{\infty} x(u) \cdot \frac{1}{\alpha} \cdot e^{-i\omega\frac{u}{\alpha}} du$$

De modo análogo ao primeiro caso, podemos reescrever a equação obtida anteriormente:

$$Y(i\omega) = -\frac{1}{\alpha}\int_{-\infty}^{\infty} x(u)\cdot e^{-i\omega\frac{u}{\alpha}}du \tag{2.19}$$

Como $\alpha < 0$, temos $\frac{1}{\alpha} < 0$ e, consequentemente, $-\frac{1}{\alpha} = \left|\frac{1}{\alpha}\right|$.

Portanto, na equação (2.19), chegamos a:

$$Y(i\omega) = \left|\frac{1}{\alpha}\right| X(\frac{i\omega}{\alpha})$$

. □

Derivada na frequência (dual da derivada). *Se $x(t)$ é um sinal e $y(t) = -(it)x(t)$, então:*

$$y(i\omega) = \frac{dX}{d\omega}(i\omega)$$

Demonstração. Pela definição da Transformada de Fourier, temos:

$$Y(i\omega) = \int_{-\infty}^{\infty} -(it)x(t)\cdot e^{-i\omega t}dt$$

Podemos reescrever a equação anterior do seguinte modo:

$$Y(i\omega) = \int_{-\infty}^{\infty} x(t)(-it)\cdot e^{-i\omega t}dt \tag{2.20}$$

Observando que:

$$(-it)\cdot e^{-i\omega t} = \frac{d}{d\omega}\cdot e^{-i\omega t}$$

e utilizando essa relação na equação (2.20), temos:

$$Y(i\omega) = \int_{-\infty}^{\infty} x(t)\frac{d}{d\omega}\cdot e^{-i\omega t}dt$$

Como $x(t)$ é constante em relação a ω, podemos reescrever a equação obtida como segue:

$$Y(i\omega) = \int_{-\infty}^{\infty} \frac{d}{d\omega}(x(t)\cdot e^{-i\omega t})dt \tag{2.21}$$

As propriedades das integrais nos permitem escrever a equação (2.21) do seguinte modo:

$$Y(i\omega) = \frac{d}{d\omega}\int_{-\infty}^{\infty} x(t)\cdot e^{-i\omega t}dt$$

E, portanto:

$$Y(i\omega) = \frac{d}{d\omega}X(i\omega)$$

□

Convolução. *Se* $x_1(t)$ *e* $x_2(t)$ *são sinais e* $y(t) = \int_{-\infty}^{\infty} x_1(t-\tau)x_2(\tau)d\tau = x_1 * x_2$, *então:*

$$Y(i\omega) = X_1(i\omega)X_2(i\omega)$$

Demonstração. Primeiramente, utilizaremos a propriedade da translação no tempo, observando que, sendo $x(t) = x_1(t-\tau)$ um sinal, logo $X(i\omega) = e^{-i\omega\tau}X_1(i\omega)$.

Pela definição da Transformada inversa de Fourier, podemos afirmar que:

$$x_1(t-\tau) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-i\omega\tau}\cdot X_1(i\omega)\cdot e^{i\omega t}d\omega \tag{2.22}$$

Multiplicando ambos os membros da equação (2.22) por $x_2(\tau)$, obtemos:

$$x_1(t-\tau)\cdot x_2(\tau) = x_2(\tau).\frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-i\omega\tau}\cdot X_1(i\omega)\cdot e^{i\omega t}d\omega$$

Como $x_2(\tau)$ é constante em relação a ω, é possível escrever:

$$x_1(t-\tau)\cdot x_2(\tau) = \frac{1}{2\pi}\int_{-\infty}^{\infty} x_2(\tau)\cdot e^{-i\omega\tau}\cdot X_1(i\omega)\cdot e^{i\omega t}d\omega$$

Integrando ambos os membros da equação anterior em relação a τ, temos:

$$\int_{-\infty}^{\infty} x_1(t-\tau)\cdot x_2(\tau)d\tau = \int_{-\infty}^{\infty}\left[\frac{1}{2\pi}\int_{-\infty}^{\infty} x_2(\tau)\cdot e^{-i\omega\tau}\cdot X_1(i\omega)\cdot e^{i\omega t}d\omega\right]d\tau$$

O primeiro membro da equação pode ser escrito como $x_1(t) * x_2(t)$ e $\frac{1}{2\pi}$ é constante em relação a τ, de modo que podemos escrever:

$$x_1(t) * x_2(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty} x_2(\tau)\cdot e^{-i\omega\tau}\cdot X_1(i\omega)\cdot e^{i\omega t}d\omega\right]d\tau$$

Pelo Teorema de Fubini, é possível inverter a ordem de integração na equação anterior:

$$x_1(t) * x_2(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty} x_2(\tau)\cdot e^{-i\omega\tau}\cdot X_1(i\omega)\cdot e^{i\omega t}d\tau\right]d\omega$$

Visto que $X_1(i\omega).e^{i\omega t}$ não varia em relação a τ, podemos escrever:

$$x_1(t) * x_2(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty}\left[X_1(i\omega).e^{i\omega t}\cdot\int_{-\infty}^{\infty} x_2(\tau)\cdot e^{-i\omega\tau}d\tau\right]d\omega$$

Pela definição da Transformada de Fourier, $\int_{-\infty}^{\infty} x_2(\tau)\cdot e^{-i\omega\tau}d\tau = X_2(i\omega)$. Consequentemente:

$$x_1(t) * x_2(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \left[X_1(i\omega) \cdot e^{i\omega t} \cdot X_2(i\omega) \right] d\omega$$

Sendo $x_1(t) * x_2(t) = y(t)$:

$$y(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \left[X_1(i\omega) \cdot e^{i\omega t} \cdot X_2(i\omega) \right] d\omega$$

Portanto, tomando a Transformada de Fourier de ambos os membros da equação, concluímos que:

$$Y(i\omega) = X_1(i\omega) \cdot X_2(i\omega)$$

□

Relação de Parseval. *Se $x(t)$ é um sinal, sua energia total pode ser expressa como:*

$$E = \int_{-\infty}^{\infty} |x^2(t)| dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} |X(i\omega)|^2 d\omega$$

Demonstração. Primeiramente, podemos verificar que:

$$|x^2(t)| = |x(t)|^2 = x(t) \cdot x^*(t) \tag{2.23}$$

Aqui, $x^*(t)$ denota o complexo conjugado do sinal $x(t)$. Considerando que $x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(i\omega) \cdot e^{i\omega t} d\omega$ (pela Transformada Inversa de Fourier),

$$x^*(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X^*(i\omega) \cdot e^{-i\omega t} d\omega$$

A partir dessa relação, podemos reescrever a equação (2.23) da seguinte forma:

$$|x^2(t)| = x(t) \cdot \frac{1}{2\pi} \int_{-\infty}^{\infty} X^*(i\omega) \cdot e^{-i\omega t} d\omega$$

Integrando ambos os membros da equação anterior em relação a t, temos:

$$\int_{-\infty}^{\infty} |x^2(t)| dt = \int_{-\infty}^{\infty} \left[x(t) \cdot \frac{1}{2\pi} \cdot \int_{-\infty}^{\infty} X^*(i\omega) \cdot e^{-i\omega t} d\omega \right] dt$$

Como $\frac{1}{2\pi}$ é constante em relação a ω e t, a integral no segundo membro da equação anterior pode ser reescrita:

$$\int_{-\infty}^{\infty} |x^2(t)| dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} \left[x(t) \cdot \int_{-\infty}^{\infty} X^*(i\omega) \cdot e^{-i\omega t} d\omega \right] dt$$

Além disso, como $x(t)$ é constante em relação a ω, temos:

$$\int_{-\infty}^{\infty} |x^2(t)|dt = \frac{1}{2\pi}\int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty} x(t)\cdot X^*(i\omega)\cdot e^{-i\omega t}d\omega\right]dt$$

O Teorema de Fubini permite que invertamos a ordem de integração na equação anterior, da seguinte forma:

$$\int_{-\infty}^{\infty} |x^2(t)|dt = \frac{1}{2\pi}\int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty} x(t)\cdot X^*(i\omega)\cdot e^{-i\omega t}dt\right]d\omega$$

Uma vez que $X^*(i\omega)$ independe de t, podemos escrever:

$$\int_{-\infty}^{\infty} |x^2(t)|dt = \frac{1}{2\pi}\int_{-\infty}^{\infty}\left[X^*(i\omega)\cdot\int_{-\infty}^{\infty} x(t)\cdot e^{i\omega t}dt\right]d\omega$$

Pela definição da Transformada de Fourier, $\int_{-\infty}^{\infty} x(t)\cdot e^{-i\omega t}dt = X(i\omega)$, então:

$$\int_{-\infty}^{\infty} |x^2(t)|dt = \frac{1}{2\pi}\int_{-\infty}^{\infty} X^*(i\omega)\cdot X(i\omega)d\omega \tag{2.24}$$

Portanto, pela equação (2.24), concluímos que:

$$\int_{-\infty}^{\infty} |x^2(t)|dt = \frac{1}{2\pi}\int_{-\infty}^{\infty} |X(i\omega)|^2 d\omega$$

□

Dualidade. *Sejam $x_1(t)$ e $x_2(t)$ sinais. Se $x_2(t) = X_1(i\omega)|_{\omega=t}$, então $X_2(-i\omega) = 2.\pi\cdot x_1(t)|_{t=\omega}$.*

Demonstração. Pela definição da Transformada Inversa de Fourier, podemos escrever o sinal $x_1(t)$ da seguinte forma:

$$x_1(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X_1(i\omega)\cdot e^{i\omega t}d\omega \tag{2.25}$$

Por hipótese, $x_2(t) = X_1(i\omega)|_{\omega=t}$, então $X_1(t) = x_2(t)$.

Daí, obtemos, na equação (2.25):

$$x_1(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X_1(i\omega)\cdot e^{i\omega t}dt$$

Logo:

$$2\pi\cdot x_1(t) = \int_{-\infty}^{\infty} X_1(i\omega)\cdot e^{i\omega t}dt$$

Reescrevendo o segundo membro da equação anterior, obtemos:

$$2\pi \cdot x_1(t) = \int_{-\infty}^{\infty} X_1(i\omega) \cdot e^{-(-i\omega t)} dt$$

Portanto, pela definição da Transformada de Fourier:

$$2\pi \cdot x_1(t) = X_2(-i\omega)$$

□

Translação na frequência ("frequency shifting"). *Se $x(t)$ é um sinal e $y(t) = e^{i\omega_0 t} \cdot x(t)$, então*

$$Y(i\omega) = X(i(\omega - \omega_0))$$

Demonstração. Pela definição da Transformada de Fourier, temos:

$$Y(i\omega) = \int_{-\infty}^{\infty} e^{i\omega_0 t} \cdot x(t) \cdot e^{-i\omega t} dt$$

Podemos reescrever a equação anterior da seguinte forma:

$$Y(i\omega) = \int_{-\infty}^{\infty} x(t) \cdot e^{-i(\omega - \omega_0)t} dt$$

Concluímos, assim, que:

$$Y(i\omega) = X(i(\omega - \omega_0))$$

□

Multiplicação (dual da convolução). *Se $x_1(t)$ e $x_2(t)$ são sinais contínuos e $y(t) = x_1(t) \cdot x_2(t)$, então:*

$$Y(i\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X_1(i\theta) \cdot X_2(i(\omega - \theta)) d\theta$$

Demonstração. Pela definição da Transformada Inversa de Fourier, podemos escrever:

$$x_1(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X_1(i\theta) \cdot e^{i\theta t} d\theta$$

$$x_2(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X_2(i\lambda) \cdot e^{i\lambda t} d\lambda$$

Assim, multiplicando as duas funções, temos:

$$x_1(t) \cdot x_2(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} (X_1(i\theta) \cdot e^{i\theta t} d\theta) \cdot \frac{1}{2\pi} \int_{-\infty}^{\infty} (X_2(i\lambda) \cdot e^{i\lambda t} d\lambda)$$

Podemos reescrever essa equação da seguinte forma:

$$y(t) = \frac{1}{2\pi} \cdot \frac{1}{2\pi} \int_{-\infty}^{\infty} X_1(i\theta) \cdot e^{i\theta t} d\theta \cdot \int_{-\infty}^{\infty} X_2(i\lambda) \cdot e^{i\lambda t} d\lambda \tag{2.26}$$

Como $\int_{-\infty}^{\infty} X_1(i\theta) \cdot e^{i\theta t} d\theta$ é constante em relação a λ, podemos escrever a equação (2.26) como segue:

$$y(t) = \frac{1}{2\pi} \cdot \frac{1}{2\pi} \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} (X_1(i\theta) \cdot e^{i\theta t} d\theta) \right] X_2(i\lambda) \cdot e^{i\lambda t} d\lambda \tag{2.27}$$

$X_2(i\lambda) \cdot e^{i\lambda t}$ é constante em relação a θ, então temos, em (2.27):

$$y(t) = \frac{1}{2\pi} \cdot \frac{1}{2\pi} \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} X_1(i\theta) \cdot e^{i\theta t} \cdot X_2(i\lambda) \cdot e^{i\lambda t} d\theta \right] d\lambda \tag{2.28}$$

Aqui, podemos realizar a seguinte mudança de variável:

$$\lambda = \omega - \theta$$

$$d\lambda = d\omega$$

Para os limites de integração, temos:

$$\lambda \to -\infty \Rightarrow \omega \to -\infty$$

$$\lambda \to \infty \Rightarrow \omega \to \infty$$

Usando essa mudança de variável na equação (2.28), obtemos:

$$y(t) = \frac{1}{2\pi} \cdot \frac{1}{2\pi} \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} X_1(i\theta) \cdot e^{i\theta t} \cdot X_2(i(\omega - \theta) \cdot e^{i(\omega-\theta)t} d\theta \right] d\omega$$

Podemos observar que, na equação anterior, temos:

$$y(t) = \frac{1}{2\pi} \cdot \frac{1}{2\pi} \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} X_1(i\theta) \cdot e^{i\theta t} \cdot X_2(i(\omega - \theta)) \cdot e^{i\omega t} \cdot e^{-i\theta t} d\theta \right] d\omega$$

Das operações usuais da potenciação, segue que:

$$y(t) = \frac{1}{2\pi} \cdot \frac{1}{2\pi} \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} X_1(i\theta) \cdot X_2(i(\omega - \theta)) \cdot e^{i\omega t} d\theta \right] d\omega \tag{2.29}$$

Usando o fato de que $e^{i\omega t}$ é constante em relação a θ, bem como $\frac{1}{2\pi}$ é constante em relação a ω, reescrevemos a equação (2.29) como:

$$y(t) = \frac{1}{2\pi} \cdot \int_{-\infty}^{\infty} \left[\frac{1}{2\pi} \int_{-\infty}^{\infty} (X_1(i\theta) \cdot X_2(i(\omega - \theta)) d\theta) \right] \cdot e^{i\omega t} d\omega \tag{2.30}$$

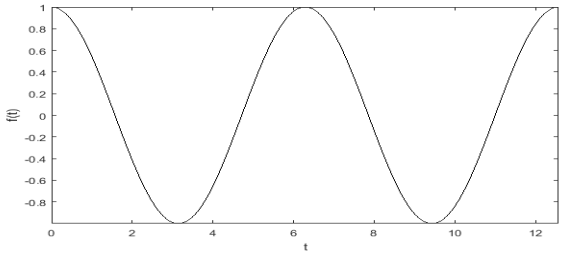
O segundo membro da equação (2.30) é a Transformada Inversa de Fourier da função $\frac{1}{2\pi} \int_{-\infty}^{\infty} (X_1(i\theta) \cdot X_2(i(\omega - \theta)))d\theta$. Portanto:

$$Y(i\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X_1(i\theta) \cdot X_2(i(\omega - \theta)) d\theta)$$

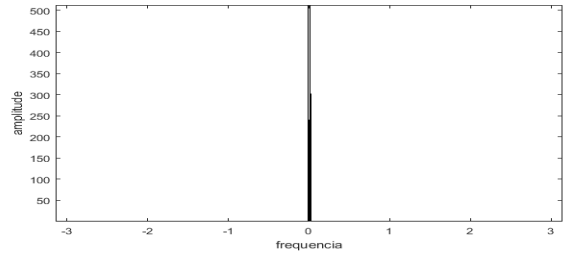
□

Observemos alguns exemplos da aplicação da Transformada de Fourier:

Sendo $f(t) = cos(t)$, os gráficos de f e de sua Transformada de Fourier serão os seguintes:



(a) Gráfico da função $f(t) = cos(t)$

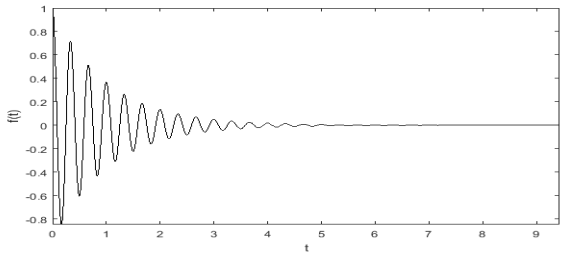


(b) Transformada de Fourier da função $f(t) = cos(t)$

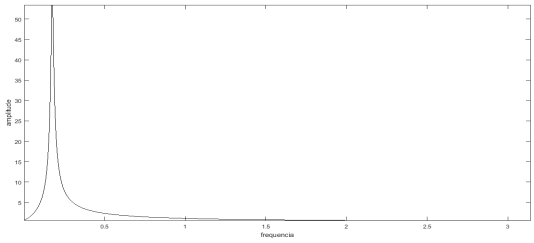
Figura 2.1 – Gráficos da função $f(t) = cos(t)$ e da sua Transformada de Fourier.

Podemos ver que o gráfico da Transformada de Fourier da função f é simétrico em relação à origem. Isso se deve ao fato de que f é um sinal real, ou seja, com parte imaginária nula. Então, no caso dos sinais reais, é usual representar graficamente apenas a parte positiva da Transformada. É o que faremos nos próximos exemplos.

Na imagem a seguir, temos os gráficos da função $f(t) = cos(6\pi \cdot t) \cdot e^{-t}$ e da sua Transformada de Fourier:



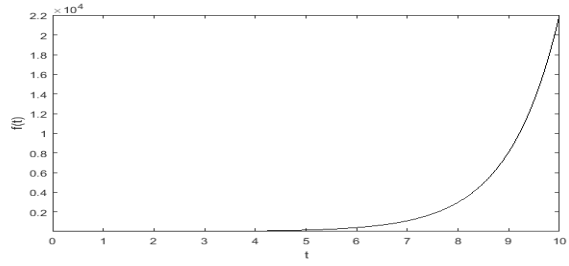
(a) Gráfico da função $f(t) = cos(6\pi \cdot t) \cdot e^{-t}$



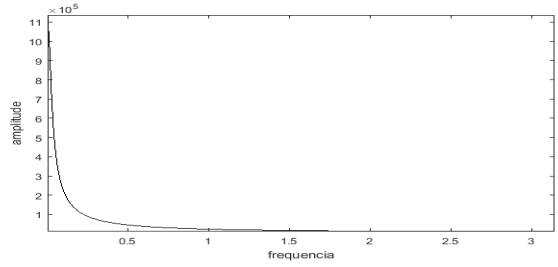
(b) Transformada de Fourier da função $f(t) = cos(6\pi \cdot t) \cdot e^{-t}$

Figura 2.2 – Gráficos da função $f(t) = cos(6\pi \cdot t) \cdot e^{-t}$ e da sua Transformada de Fourier.

A próxima imagem, por sua vez, representa a função $f(t) = e^t$ e sua respectiva Transformada de Fourier:



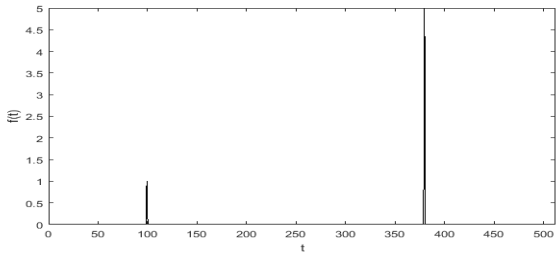
(a) Gráfico da função $f(t) = e^t$



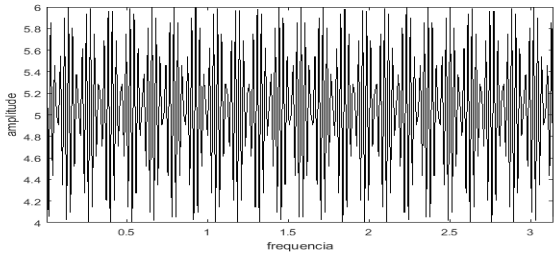
(b) Transformada de Fourier da função $f(t) = e^t$

Figura 2.3 – Gráficos da função $f(t) = e^t$ e da sua Transformada de Fourier.

Nosso último exemplo é uma função $f(t)$ que assume valor 0 para todos os valores de t, exceto em $t = 100$ e $t = 380$ (uma função semelhante a esta será utilizada na seção 4 deste capítulo):



(a) Gráfico da função $f(t)$



(b) Transformada de Fourier da função $f(t)$

Figura 2.4 – Gráficos da função $f(t)$ e da sua Transformada de Fourier.

Embora seja útil para a análise de sinais, a Transformada de Fourier utiliza combinações de senos e cossenos para representá-los, e tais funções não têm boa localização no domínio do tempo. Além disso, a TF não traz em sua representação a informação temporal do sinal, o que pode limitar sua utilização, do ponto de vista prático. Tendo essas limitações em vista, é possível considerar outras ferramentas para a análise de sinais que permitam representá-los nos domínios do tempo e da frequência simultaneamente. As Transformadas *Wavelet*, que abordaremos nas próximas seções, apresentam essa característica.

2.3 Introdução à Transformada *Wavelet* Contínua

As funções *wavelets* estão associadas à ideia de onda e possuem propriedades que tornam a Transformada *Wavelet* Contínua (também chamada de TWC) uma ferramenta útil para a decomposição e análise de sinais. Assim sendo, dada uma função $\Psi(t) \in \mathcal{L}^2$ (espaço das funções quadrado-integráveis), existem duas propriedades que a caracterizam como função *wavelet*:

1) Sua integral em relação a t é igual a 0, ou seja:

$$\int \Psi(t)dt = 0$$

Essa condição, também conhecida como condição de admissibilidade (DAUBECHIES et al., 1992), assegura o caráter oscilatório da função; em outras palavras, trata-se de uma onda, como o próprio nome sugere.

2) Sua energia é unitária (BURRUS; GOPINATH; GUO, 1997):

$$\int |\Psi(t)|^2 dt = 1$$

Assim, diz-se que a *wavelet* possui suporte compacto, o que significa que deve existir um intervalo $[a, b]$ fora do qual o valor da função é sempre igual a 0. Esta propriedade possibilita a preservação da informação temporal do sinal após a transformação, o que não era possível com a Transformada de Fourier.

Tendo em vista as características das funções *wavelets*, torna-se interessante definir a Transformada *Wavelet* Contínua de um sinal $f(t)$ da seguinte forma, de acordo com a notação adotada no texto *A Really Friendly Guide to Wavelets* (VALENS, 1999):

$$\gamma(a, b) = \int f(t) \cdot \Psi^*_{a,b}(t) dt$$

A função $\Psi^*_{a,b}(t)$ é o conjugado da chamada função *wavelet* e é dilatada e transladada no tempo como segue:

$$\Psi^*_{a,b}(t) = \frac{1}{\sqrt{a}} \cdot \overline{\Psi\left(\frac{t-b}{a}\right)}$$

A função Ψ é conhecida como *Mother-Wavelet* e a partir dela são obtidas as demais funções *wavelets*. O detalhamento da análise oferecida pela TWC é dado pelo parâmetro a, chamado de escala, e o fator $\frac{1}{\sqrt{a}}$ é usado para normalização. Já o parâmetro b define a localização da *wavelet* no decorrer do tempo. Ou seja, $\Psi^*_{a,b}(t)$ é obtida a partir da função Ψ, com um escalonamento de a e um deslocamento de b.

Vale destacar que, para a Transformada *Wavelet*, vale o teorema de Parseval (BURRUS; GOPINATH; GUO, 1997), que associa a energia total de um sinal com a energia de cada um de seus componentes e seus coeficientes de *wavelet*. Assim, dada uma função $f(t)$ e sua TWC $\gamma(a, b)$, temos:

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} |\gamma(a, b)|^2 da db = \int_{-\infty}^{\infty} |f(t)|^2 dt$$

Existem diversas funções *wavelet*, cujo uso varia de acordo com a finalidade da Transformada. Nesse trabalho, utilizamos prioritariamente a *wavelet* de Morlet, mas também apresentamos a seguir uma das funções *wavelets* mais utilizadas, que é a *wavelet* Chapéu Mexicano.

A *wavelet* Chapéu Mexicano é definida da seguinte forma:

$$f(t) = \frac{2}{3^{0.5} \cdot \pi^{0.25}} \cdot e^{\frac{-t^2}{2}} \cdot (1 - t^2)$$

Trata-se de uma função real, ou seja, com parte imaginária nula em todos os seus pontos, e é uma das funções mais utilizadas no processamento digital de sinais.

A função que utilizaremos em nossos testes, a *wavelet* de Morlet, é definida como:

$$g(t) = e^{\frac{i \cdot \pi}{2 \cdot t}} \cdot \frac{1}{(\pi \cdot 5^2)^{\frac{1}{4}}} \cdot e^{\frac{-t^2}{2 \cdot 5^2}}$$

A *wavelet* de Morlet, ao contrário da Chapéu Mexicano, é uma *wavelet* complexa, ou seja, sua parte imaginária não é necessariamente nula em todos os seus pontos.

Na imagem a seguir, temos a representação gráfica dessas duas funções (e, no caso da *wavelet* de Morlet, temos as partes real e imaginária).

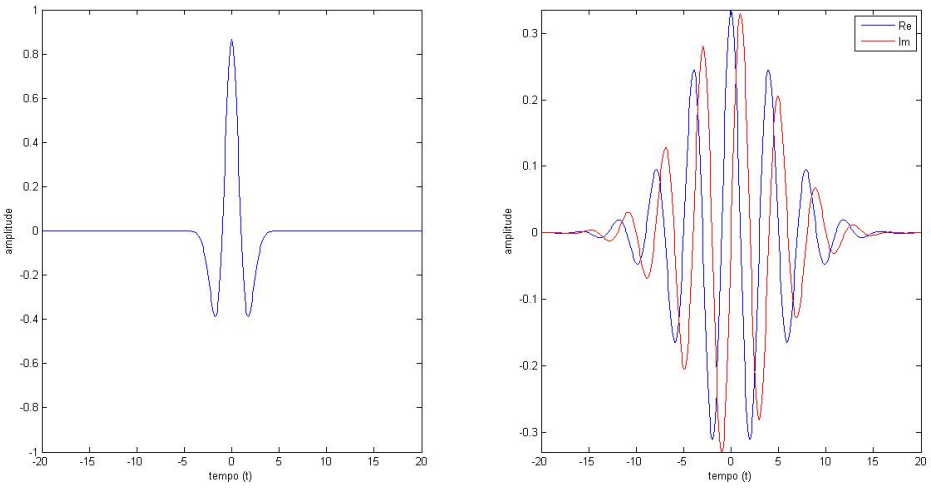


Figura 2.5 – Chapéu Mexicano (à esquerda) e *wavelet* de Morlet (à direita)

Na próxima seção, serão apresentados exemplos de análises de sinais utilizando a TWC mas, antes, precisamos entender de que forma é feita a representação dos dados obtidos por meio dessa transformada.

As informações obtidas a partir da TWC são geralmente expressas de duas formas: os escalogramas e o espectro global. O escalograma é uma ferramenta gráfica que permite exibir, em um mesmo plano, a informação temporal, as escalas e a energia local do sinal, que corresponde ao quadrado do valor absoluto dos coeficientes de *wavelet*. Os coeficientes de *wavelet* medem a regularidade do sinal, ou seja, o quanto esse sinal oscila no decorrer do tempo. Assim, é possível verificar, através de uma imagem, a quantidade de energia associada a um certo momento t, para uma escala a.

Já o espectro global é, por definição, a integração do quadrado do valor absoluto dos coeficientes de *wavelet* no decorrer do tempo. Assim, essa ferramenta é utilizada para exibir a energia global do sinal, ou seja, a quantidade de energia associada a uma certa

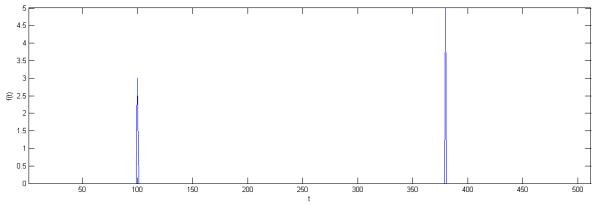
escala, no decorrer do tempo. O espectro global pode ser definido a partir da seguinte função:

$$E(a) = \int_{-\infty}^{\infty} |\gamma(a,b)|^2 dt$$

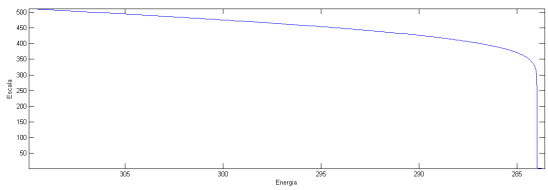
Vale observar que, quanto mais regular for o sinal, mais próximo será seu espectro global de sua Transformada de Fourier.

2.4 Analisando Funções com a TWC

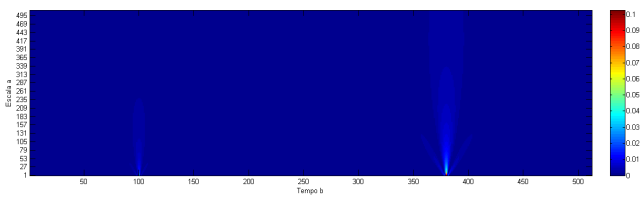
A seguir, analisaremos uma função que assume valores iguais a 0 para todos os valores de t, exceto $t = 100$, de modo que $f(100) = 3$ e em $t = 380$, com $f(380) = 5$. Aqui, utlizaremos a *wavelet* Chapéu Mexicano e as seguintes imagens apresentam o escalograma e o espectro global referentes a essa análise.



(a) Primeira função teste



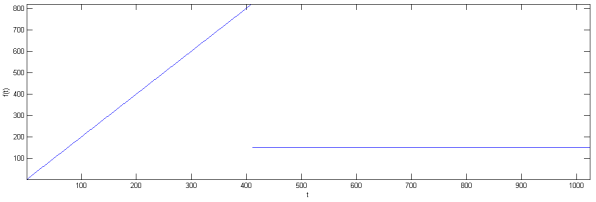
(b) Espectro global



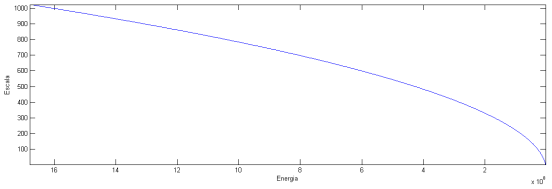
(c) Escalograma

Figura 2.6 – Primeiro exemplo de função analisada com a *wavelet* Chapéu Mexicano

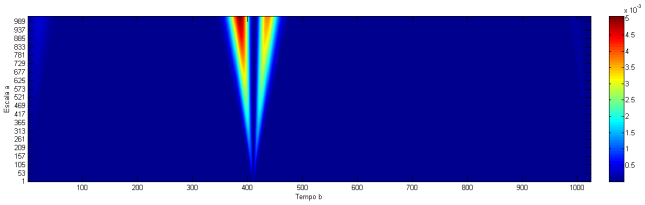
Nosso segundo exemplo é uma função f com duas características diferentes: para um certo intervalo do domínio, a função assume a forma de uma função linear crescente; para os demais valores do domínio, o valor de f é constante.



(a) Segunda função teste



(b) Espectro global

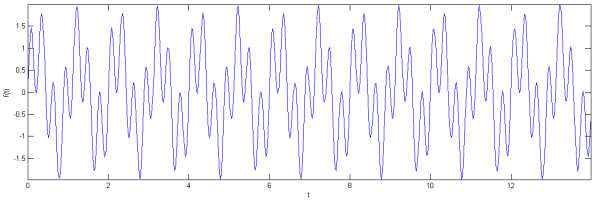


(c) Escalograma

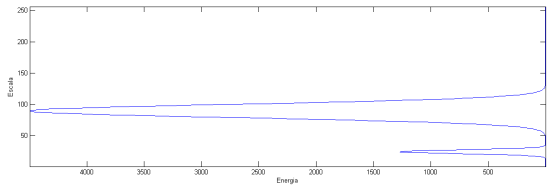
Figura 2.7 – Segundo exemplo de função analisada com a *wavelet* Chapéu Mexicano

O terceiro exemplo abordado neste trabalho é o da seguinte função trigonométrica:

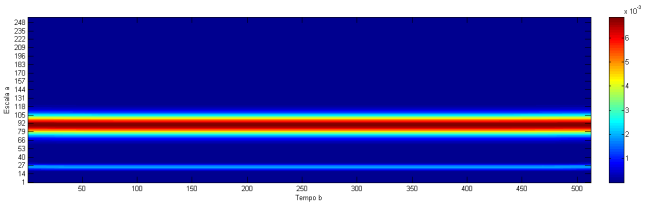
$$f(t) = sin(2\pi t) + sin(7\pi t)$$



(a) Terceira função teste



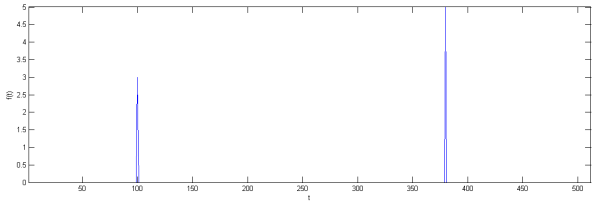
(b) Espectro global



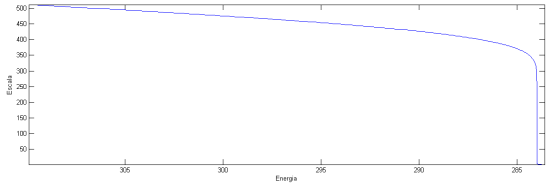
(c) Escalograma

Figura 2.8 – Primeiro exemplo de função analisada com a *wavelet* de Morlet

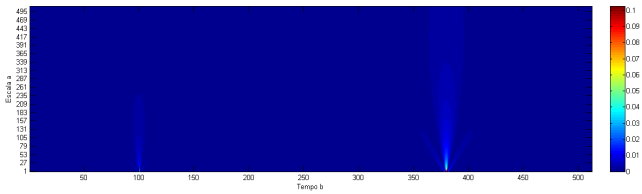
O quarto e último exemplo desta seção é uma função trigonométrica em que há mudança de amplitude. Os efeitos dessa mudança podem ser percebidos com clareza no escalograma, como veremos a seguir:



(a) Quarta função teste



(b) Espectro global



(c) Escalograma

Figura 2.9 – Segundo exemplo de função analisada com a *wavelet* de Morlet

2.5 Um breve histórico da teoria das *wavelets*

As *wavelets* foram desenvolvidas no decorrer do tempo a partir de motivações diversas. Aqui, abordaremos apenas algumas delas, com base no livro *Wavelets: Algorithms & Applications* (MEYER, 1993), e nos textos *Ten Lectures on Wavelets* (DAUBECHIES et al., 1992) e *Where do wavelets come from? - A personal point of view* (DAUBECHIES, 1996). O desenvolvimento da teoria da análise de *wavelets* teve início nas primeiras décadas do século XX. Nessa época, cientistas e pesquisadores de diversas áreas buscavam ferramentas para resolver problemas que não podiam ser solucionados com a análise de Fourier. Inspirados por essa teoria, pesquisadores como o húngaro Haar, o ucraniano Schauder e o alemão Faber passaram a buscar conjuntos de funções que possibilitassem a representação de uma função dada em certas condições. Em outras palavras, os cientistas procuravam funções $f_0(x), f_1(x), ..., f_n(x), ...$ tais que a série

$$< f_0(x), f(x) > \cdot f_0(x) + < f_1(x), f(x) > \cdot f_1(x) + ... + < f_n(x), f(x) > \cdot f_n(x) + ...,$$

onde $< f_i(x), f(x) >$ representa o produto interno entre $f_i(x)$ e $f(x)$, convergisse para $f(x)$ no intervalo $[0, 1)$. Esta busca era motivada, em alguns casos, pela má localização temporal das funções utilizadas por Fourier. Os estudos desenvolvidos por estes cientistas tiveram resultados que, embora permitissem a reescrita de funções nestas condições, ainda apresentavam dificuldades teóricas que limitavam sua aplicação em situações práticas, como o estudo de fenômenos físicos com estruturas multifractais. O que se tinha era a tentativa de obter informações acerca de um sinal, mas cuja análise dependia diretamente dos dados experimentais, o que nem sempre era conveniente. Tais dificuldades seriam solucionadas, posteriormente, com as funções *wavelets*.

Além da má loclização temporal das funções utilizadas nas séries de Fourier, havia ainda a dificuldade de analisar características como a energia de um sinal. A ferramenta

provida por Fourier determinava que o valor médio da energia seria dado pela integração dos quadrados dos coeficientes de Fourier, dividida por 2π. Este cálculo, no entanto, não indicava se a energia estava dispersa em um sinal ou concentrada em alguns pontos. Tendo isso em vista, Littlewood e Paley desenvolveram estudos com resultados que permitiam relacionar a série de Fourier de uma função à localização de sua energia e, mais do que isso, motivaram pesquisas que resultaram em ferramentas relevantes para o processamento de imagens. Os responsáveis por essas pesquisas foram os integrantes do grupo de Antoni Zygmund na Universidade de Chicago, que buscava generalizar os resultados de Littlewood e Paley (adequados ao caso unidimensional) para situações n-dimensionais. Como resultado, obteve-se uma função $\psi(x)$ sob as seguintes condições:

1) Denotando por $\Psi(i\omega)$ a transformada de Fourier de $\psi(x)$, tem-se que:

$$\Psi(i\omega) = 1, 1 - \alpha \leq |i\omega| \leq 2 - 2\alpha$$

Onde $\alpha \in \left(0, \frac{1}{3}\right]$.

2)$\Psi(i\omega) = 0, |i\omega| < 1 - \alpha, 2 - 2\alpha < |i\omega|$

3)$\Psi(i\omega)$ é infinitamente diferenciável em $\mathbb{R}^n$ e

4) $\sum_{-\infty}^{\infty} \Psi\left(\frac{i\omega}{2^{-j}}\right) = 1, \omega \neq 0.$

Tais condições possibilitaram o desenvolvimento da análise Littlewood-Paley-Stein que, por sua vez, trazia variação de escalas e conservação de energia, precedendo a análise de *wavelets*.

Apesar de todos os esforços que levaram ao desenvolvimento da teoria das *wavelets*, as ferramentas desenvolvidas até os anos de 1960 ainda não possuíam os nomes pelos quais as conhecemos hoje. A primeira definição do que é, de fato, uma *wavelet* é devida a Morlet e Grossmann e surgiu por volta de 1970. Os dois definiram uma *wavelet* como uma função $\psi(x)$ em $\mathcal{L}^2$ cuja Transformada de Fourier $\Psi(i\omega)$ atende à igualdade $\int_0^\infty |\Psi(i\omega t)|\frac{dt}{t} = 1$ em quase todos os seus pontos. Esta definição surgiu como resultado de estudos desenvolvidos por Morlet naquela década, iniciados a partir de uma possível alteração para a Transformada de Fourier. No decorrer de seus estudos, Morlet surgiu com uma nova forma de utilizar a Transformada de Fourier, que consistia em dividir o sinal em intervalos curtos de tempo e realizar a análise de cada intervalo. Assim, mantinha-se a informação temporal do sinal analisado e o resultado poderia ser interpretado a partir de dois parâmetros: o tamanho da "janela" utilizada e o deslocamento desta janela no decorrer do tempo. Este método ficou conhecido como *Short Time Fourier Transform* (STFT) e ainda apresentava dificuldades relacionadas à diferença das frequências analisadas. Para solucionar essa questão, Morlet teve a ideia de gerar as funções para as transformadas de um jeito diferente: utilizou uma "janela" em uma função cosseno, comprimiu-a no tempo para obter uma função de maior frequência e dilatou-a para obter uma função de frequência mais baixa. Por fim, as funções

obtidas neste procedimento eram deslocadas no tempo para que se pudesse investigar o que ocorria em diferentes momentos; dependiam, portanto, de dois parâmetros, um que representava a localização no tempo e outro que determinava sua compressão ou dilatação. Para que se realizasse a análise, tomava-se o produto interno do sinal com as funções obtidas.

A análise desenvolvida por Morlet foi incrementada com o auxílio de Grossmann, que providenciou uma operação que revertia a transformação descrita anteriormente e os dois desenvolveram juntos diversas aplicações para as ferramentas obtidas. A definição dada por Morlet e Grossmann foi modificada no decorrer do tempo de modo a especificar mais claramente o sentido de função *wavelet* e, ao mesmo tempo, possibilitar análises em espaços diversos com funções variadas.

2.6 Transformada de Fourier no MATLAB e pacote YAWTB

Em nosso trabalho, utilizamos o *software* MATLAB para calcular e obter representações gráficas da Transformada de Fourier e da Transformada *Wavelet* Contínua de funções. O MATLAB é um *software* com diversas finalidades matemáticas e possui ferramentas que facilitam desde as operações com matrizes e vetores até as resoluções de equações diferenciais, esboços de gráficos bidimensionais e tridimensionais, entre outras possibilidades. Nesta pesquisa, utilizamos o MATLAB para efetuar as interpolações mais rapidamente, seja por meio de suas ferramentas específicas, como é o caso dos *splines* cúbicos e do polinômio cúbico de Hermite, para os quais a interpolação é feita por comandos já programados, seja pela programação manual, utilizada na interpolação linear, a curva de Bézier e o polinômio de Chebyshev. Tais procedimentos, embora sejam trabalhosos, podem ser mais vantajosos do que o cálculo manual e render resultados mais precisos.

Embora o MATLAB ofereça inúmeras ferramentas, é possível, ainda, utilizar outros pacotes de ferramentas neste programa. Aqui, utilizamos o pacote *Yet Another Wavelet Toolbox*, também conhecido como YAWTB. Trata-se de um pacote de ferramentas que pode ser adquirido gratuitamente, voltado para a análise multiescala de sinais. Foi projetado de modo que seu uso seja simples e possua boa documentação. Conta, inclusive, com uma ferramenta de ajuda, chamada *yahelp*, com a qual se pode obter informações sobre determinada funcionalidade dentro do próprio programa. Para utilizá-lo no MATLAB, seguimos alguns passos: após realizar o *download* dos arquivos do pacote em seu próprio site, selecionamos, no diretório, a pasta em que foram extraídos tais arquivos. Em seguida, digitamos na linha de comando a palavra *yaload*. A partir daí, o pacote já pode ser utilizado. O cálculo da Transformada de Fourier de um sinal pode ser feita por meio do comando *fft*, enquanto a TWC é realizada pelo comando *cwt1d*. No *site* do YAWTB, desenvolvido

por Jacques et al. (2001), o leitor pode realizar o *download* do pacote, verificar manuais e exemplos do uso de suas ferramentas. Em nosso Apêndice B, também apresentamos os códigos que geram os resultados apresentados; pode-se verificar, portanto, o uso desta *toolbox*.

Apesar de existirem *toolboxes* voltadas para a análise de sinais no próprio MATLAB, a YAWTB oferece a vantagem de ser gratuito e poder ser utilizado em outros *softwares* também gratuitos e semelhantes ao MATLAB, como o Octave.

3 Interpolação numérica: alguns métodos

Neste capítulo, realizaremos uma breve abordagem do conceito de interpolação, com base em leituras dos livros *Cálculo Numérico: Aspectos teóricos e computacionais* (RUGGIERO; LOPES, 1997) e *Análise Numérica* (BURDEN; FAIRES; TASKS, 2008). A escolha de tais autores se deve à forma como as ferramentas são construídas em seus livros, que se mostra adequada aos objetivos deste trabalho por sua objetividade e clareza. Abordaremos, primeiramente, os cinco tipos de interpolação que utilizaremos e, em seguida, aplicaremos os conceitos expostos em uma série com falhas, analisando, posteriormente, os resultados obtidos.

3.1 O que são polinômios interpoladores?

Dado um conjunto de informações obtidas por meio de um experimento, pode ser interessante ajustá-los por meio de uma função, de modo a estimar os dados que não foram coletados. Um exemplo desse tipo de situação é a medição de temperatura em alguma localidade, realizada de hora em hora. Se quisermos estimar a temperatura alcançada em algum momento entre duas medições, o uso de uma função que associe os dados coletados pode ser útil. O processo de obtenção dessa função é chamado de *interpolação*.

Em nosso estudo, utilizaremos prioritariamente a *interpolação polinomial*, que consiste na interpolação por meio de polinômios, chamados *polinômios interpoladores*. Então, seja f uma função cujos valores conhecemos em $n+1$ pontos $x_0, x_1, ..., x_n$. Ou seja, conhecemos $n+1$ valores $y_0, y_1, ..., y_n$ de modo que:

$$f(x_0) = y_0, f(x_1) = y_1, ..., f(x_n) = y_n$$

Nosso objetivo com a interpolação polinomial é encontrar um polinômio $P_n(x)$, de grau menor ou igual a n, de modo que, para todo $i = 0, 1, ..., n$, tenha-se $P_n(x_i) = f(x_i)$.

A existência de tal polinômio é garantida pelo Teorema de Weierstrass, que pode ser enunciado como a seguir (BURDEN; FAIRES; TASKS, 2008):

Teorema da Aproximação de Weierstrass: Suponha que f seja definida e contínua em $[a, b]$. Para cada $\epsilon > 0$, existirá um polinômio $P(x)$, com a propriedade que $|f(x) - P(x)| < \epsilon$ para todo x em $[a, b]$. Temos ainda que tal polinômio terá grau menor do que ou igual a n, uma vez que existem $n+1$ condições a serem satisfeitas, e que esse polinômio é único.

A seguir, examinaremos as cinco formas de interpolação que são utilizadas para tratar as falhas das séries em nosso trabalho: interpolação linear, polinômio cúbico de Hermite, *splines* cúbicos, polinômio de Chebyshev e curva de Bézier.

3.2 Interpolação linear

O método de interpolação linear adotado em nosso trabalho foi considerado por ser a técnica mais utilizada para tratamento de falhas em geofísica espacial. Este método consiste em preencher as falhas com segmentos de retas que passem pelas duas extremidades de cada lacuna. Suponhamos, então, que temos dois pontos A e B cujas coordenadas no plano cartesiano são (x_1, y_1) e (x_2, y_2), respectivamente.

Seja $y = a \cdot x + b$ a equação reduzida da reta que passa por A e B. Assim, temos o seguinte sistema:

$$\begin{cases} a \cdot x_1 + b &= y_1 \\ a \cdot x_2 + b &= y_2 \end{cases}$$

Subtraindo a segunda equação da primeira, membro a membro, temos:

$$a \cdot (x_1 - x_2) = y_1 - y_2$$

$$a = \frac{y_1 - y_2}{x_1 - x_2}$$

Tendo estabelecido o valor de a, basta substituirmos este valor em alguma das equações do sistema obtido. Assim, encontramos a equação da reta que passa por A e B.

3.3 Polinômio cúbico de Hermite

Além dos valores de uma função em determinados pontos, os valores das suas derivadas nesses mesmos pontos podem ser bastante úteis e fornecer ferramentas para alguns tipos de interpolação. Observemos a seguinte definição.

Definição 3.3.1. Sejam $x_0, x_1, ..., x_n$ pontos distintos, associados aos valores $m_0, m_1, ..., m_n$ e $m = max\Big\{m_0, m_1, ..., m_n\Big\}$. O *polinômio osculador* é o polinômio de menor grau que tem a propriedade de coincidir com f e suas derivadas de grau menor ou igual a m_i em x_i, com $i = 0, ..., n$. Ou seja:

$$\frac{d^k P(x_i)}{dx^k} = \frac{d^k f(x_i)}{dx^k}$$

para todo $k = 0, 1, ..., m_i$. Quando $m_i = 1$ para cada $i = 0, ..., n$, temos o *polinômio de Hermite*. Assim, o polinômio de Hermite que aproxima f é o polinômio P que coincide com f em x_i, e cuja primeira derivada também coincide com a primeira derivada de f em x_i.

O polinômio de Hermite pode ser obtido por meio do polinômio interpolador de Lagrange ou pelo método das diferenças divididas. Tal polinômio terá grau máximo igual a $2n + 1$ e assumirá a seguinte forma:

$$H_{2n+1}(x) = \sum_{n=0}^{n} f(x_i) \cdot H_{n,i} f(x_i) + \sum_{n=0}^{n} f'(x_i) \cdot H'_{n,i} f(x_i)$$

Para obtê-lo a partir do método das diferenças divididas, primeiramente, consideremos a seguinte sequência, obtida a partir de $x_0, x_1, ..., x_n$:

$$z_{2i} = z_{2i+1} = x_i$$

Podemos construir as diferenças divididas relacionadas a esse conjunto de dados. Observe que as diferenças divididas $[z_{2i}, z_{2i+1}]$ não são definidas como as demais, uma vez que $z_{2i} = z_{2i+1}$ e, consequentemente, $f(z_{2i}) = f(x_{2i+1})$. No entanto, assumimos que a substituição[1] adequada nesse caso é a seguinte:

$$[z_{2i}, z_{2i+1}] = f'(x_i)$$

Considerando a forma geral do polinômio cúbico de Hermite e suas propriedades, temos, usando a relação acima:

$$H_{2n+1}(x) = f[z_0] + \sum_{k=1}^{2n+1} f[z_0, z_1, ..., z_k](x - z_0)(x - z_1)...(x - z_k)$$

O polinômio cúbico de Hermite também pode ser definido da seguinte forma:

$$H_{2n+1}(x) = \sum_{j=0}^{n} f(x_j) \cdot H_{n,j}(x) + \sum_{j=0}^{n} f'(x_j) \cdot \hat{H}_{n,j}(x)$$

Onde:

$$H_{n,j}(x) = [1 - 2(x - x_j) \cdot L'_{n,j}(x_j)] L^2_{n,j}(x)$$

$$\hat{H}_{n,j}(x) = (x - x_j) \cdot L_{n,j}(x_j)^2(x)$$

[1] A razão para que assumamos essa substituição encontra-se no Apêndice A.

O termo $L_{n,j}(x_j)$ é obtido através da teoria do polinômio interpolador de Lagrange. Mais detalhes sobre a construção desse polinômio interpolador podem ser encontrados no Apêndice A.

A partir destes resultados, podemos obter um polinômio que, sob certos aspectos, interpola os valores de $f(x)$ e $f'(x)$, justificado pela forma do polinômio interpolador de Newton. Na prática, construímos um polinômio que coincide com $f(x)$ em x_i, $i = 1, 2, ..., n$ e com sua derivada nestes mesmos pontos, o que caracteriza o resultado como um polinômio cúbico de Hermite. Este aspecto justifica sua escolha para a interpolação em séries com falhas, introduzindo no polinômio interpolador mais uma característica dos dados originais.

3.4 *Splines* cúbicos

Segundo Ruggiero (1997, p. 243), a interpolação por meio de um polinômio de grau n em uma função cujos valores conhecemos em $n+1$ pontos pode gerar resultados distantes da função original. Assim, uma opção para minimizar este problema são as chamadas funções *spline*, que se caracterizam da seguinte forma (RUGGIERO, 1997, p. 245):

Definição 3.4.1. Uma função $S_p(x)$ é denominada *spline* interpolante quando satisfaz três condições:

i) Em cada subintervalo $[x_i, x_{i+1}]$, com $i = 1, 2, ..., n-1$, $S_p(x)$ é um polinômio de grau p;

ii) $S_p(x)$ é contínua e possui derivada contínua até a ordem $p-1$ em cada subintervalo;

iii) $S_p(x_i) = f(x_i)$ para todo $i = 0, 1, ..., n-1$.

Nesse trabalho, utilizaremos os *splines* cúbicos, que são as *splines* interpolantes de grau 3. Tais funções possuem a vantagem de serem contínuas e possuírem a primeira e a segunda derivadas contínuas nos nós, evitando picos e mudanças abruptas nos nós (RUGGIERO, 1997, p. 248).

A seguinte caracterização dos *splines* cúbicos foi dada por Burden (2008, p. 136-137):

1. $S_j(x_j) = f(x_j)$, para cada $j = 0, ..., n-1$;
2. $S_{j+1}(x_{j+1}) = f(x_{j+1})$, para cada $j = 0, ..., n-1$;
3. $S_{j+1}(x_{j+1}) = S_j(x_{j+1})$ para cada $j = 0, ..., n-2$;
4. $S'_{j+1}(x_{j+1}) = S'_j(x_{j+1})$ para cada $j = 0, ..., n-2$;
5. $S''_{j+1}(x_{j+1}) = S''_j(x_{j+1})$ para cada $j = 0, ..., n-2$;

6. Um dos conjuntos de condições a seguir é satisfeito:

   a) $S''(x_0) = S''(x_n) = 0$;

   b) $S'(x_0) = f'(x_0)$ e $S'(x_n) = f'(x_n)$.

As duas primeiras condições estão relacionadas à interpolação propriamente dita, e permitem que o polinômio S coincida com a função f em cada ponto dado. A condição 3 faz com que os *splines* obtidos em cada subintervalo coincidam nos nós, de modo que não haja falha na construção como um todo. As condições 4 e 5 asseguram que as derivadas também coincidam nesses pontos e, assim, não ocorram problemas com a primeira e a segunda derivada de S. Por fim, a condição 6 se relaciona às condições de contorno. No item *a)* temos as condições de contorno livres ou naturais, de modo que, nesse caso, o polinômio toma a forma de uma linha reta após o intervalo $[a, b]$. No item *b)*, temos condições de contorno fixadas, que definem a forma que a curva tomará após o intervalo.

Para obter a forma dos *splines* cúbicos, utilizaremos a seguinte forma geral:

$$S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3$$

O seguinte desenvolvimento dos *splines* cúbicos e o cálculo dos coeficientes a_j, b_j, c_j e d_j foi realizado de acordo com o texto de Burden (2008, p. 147-148).

Primeiramente, substituímos x por x_j na forma geral adotada. Conforme a primeira condição, temos:

$$S_j(x_j) = a_j + b_j(x_j - x_j) + c_j \cdot (x_j - x_j)^2 + d_j(x_j - x_j)^3 = f(x_j)$$

$$a_j + b_j \cdot 0 + c_j \cdot 0^2 + d_j \cdot 0^3 = f(x_j)$$

$$a_j = f(x_j)$$

Da mesma forma, temos que:

$$a_{j+1} = f(x_{j+1}) = S_{j+1}(x_{j+1})$$

Seja $h_j = x_{j+1} - x_j$. Assim, da condição 3, temos que:

$$S_{j+1}(x_{j+1}) = S_j(x_{j+1}) = a_j + b_j(x_{j+1} - x_j) + c_j(x_{j+1} - x_j)^2 + d_j(x_{j+1} - x_j)^3$$

$$a_{j+1} = a_j + b_j \cdot h_j + c_j \cdot h_j^2 + d_j \cdot h_j^3$$

Calculemos, então, $S'_j(x)$:

$$S'_j(x) = b_j + 2 \cdot c_j(x - x_j) + 3 \cdot d_j(x - x_j)^2$$

Logo, substituindo novamente x por x_j na expressão anterior:

$$S'j(x_j) = b_j + 2 \cdot c_j(x_j - x_j) + 3 \cdot d_j(x_j - x_j)^2$$
$$S'_j(x_j) = b_j + 2 \cdot c_j \cdot 0 + 3 \cdot d_j \cdot 0^2$$
$$S'_j(x_j) = b_j$$

De modo análogo, obtemos:

$$S'_{j+1}(x_{j+1}) = b_{j+1}$$

Da condição 4, temos:

$$b_{j+1} = b_j + 2 \cdot c_j \cdot h_j + 3 \cdot d_j \cdot h_j^2$$

Calculando $S''_j(x)$:

$$S''_j(x) = 2 \cdot c_j + 6 \cdot d_j(x - x_j)$$

Então, segue que:

$$S''_j(x_j) = 2 \cdot c_j + 6 \cdot d_j(x_j - x_j)$$
$$S''_j(x_j) = 2 \cdot c_j + 6 \cdot 0$$
$$c_j = \frac{S''_j(x_j)}{2}$$

Por fim, utilizando a condição 5:

$$S''_{j+1}(x_{j+1}) = S''_j(x_{j+1})$$
$$2 \cdot c_{j+1} = 2 \cdot c_j + 6 \cdot d_j \cdot h_j$$

Temos, então, as seguintes relações:

$$a_{j+1} = a_j + b_j \cdot h_j + c_j \cdot h_j^2 + d_j \cdot h_j^3 \tag{3.1}$$

$$b_{j+1} = b_j + 2 \cdot c_j \cdot h_j + 3 \cdot d_j \cdot h_j{\cdot}^2 \tag{3.2}$$

$$c_{j+1} = c_j + 3 \cdot d_j \cdot h_j \tag{3.3}$$

De (3.3), temos:

$$d_j = \frac{c_{j+1} - c_j}{3 \cdot h_j}$$

Substituindo em (3.1):

$$a_{j+1} = a_j + b_j \cdot h_j + c_j \cdot h_j^2 + \frac{c_{j+1} - c_j}{3 \cdot h_j} \cdot h_j^3 = a_j + b_j \cdot h_j + c_j \cdot h_j^2 + \frac{c_{j+1} - c_j}{3} \cdot h_j^2$$

$$a_{j+1} = a_j + b_j \cdot h_j + \frac{c_{j+1} + 2 \cdot c_j}{3} \cdot h_j^2 \tag{3.4}$$

E em (3.2):

$$b_{j+1} = b_j + 2 \cdot c_j \cdot h_j + 3 \cdot \frac{c_{j+1} - c_j}{3 \cdot h_j} \cdot h_j^2$$

$$b_{j+1} = b_j + (c_{j+1} + c_j) \cdot h_j \tag{3.5}$$

Podemos reduzir os índices na equação (3.5) de modo a obter:

$$b_j = b_{j-1} + h_{j-1}(c_{j-1} + c_j) \tag{3.6}$$

Isolando b_j em (3.4), obtemos:

$$b_j = \left(a_{j+1} - a_j - \frac{c_{j+1} + 2 \cdot c_j}{3} \cdot h_j^2\right) \cdot \frac{1}{h_j}$$

$$b_j = \frac{1}{h_j}(a_{j+1} - a_j) - \frac{h_j}{3} \cdot (c_{j+1} + 2 \cdot c_j) \tag{3.7}$$

Reduzindo novamente os índices:

$$b_{j-1} = \frac{1}{h_{j-1}}(a_j - a_{j-1}) - \frac{h_{j-1}}{3} \cdot (c_j + 2 \cdot c_{j-1}) \tag{3.8}$$

Por fim, substituindo os valores obtidos em (3.7) e (3.8) na equação (3.6), obtemos o seguinte resultado:

$$\frac{1}{h_j}(a_{j+1} - a_j) - \frac{h_j}{3} \cdot (c_{j+1} + 2 \cdot c_j) = \frac{1}{h_{j-1}}(a_j - a_{j-1}) - \frac{h_{j-1}}{3} \cdot (c_j + 2 \cdot c_{j-1}) + h_{j-1}(c_{j-1} + c_j)$$

$$\frac{1}{h_j}(a_{j+1}-a_j)-\frac{1}{h_{j-1}}(a_j-a_{j-1})=\frac{h_j}{3}\cdot(c_{j+1}+2\cdot c_j)-\frac{h_{j-1}}{3}\cdot(c_j+2\cdot c_{j-1})+h_{j-1}(c_{j-1}+c_j)$$

$$\frac{3}{h_j}(a_{j+1}-a_j)-\frac{3}{h_{j-1}}(a_j-a_{j-1})=h_j\cdot(c_{j+1}+2\cdot c_j)-h_{j-1}\cdot(c_j+2\cdot c_{j-1})+h_{j-1}(c_{j-1}+c_j)$$

$$\frac{3}{h_j}(a_{j+1}-a_j)-\frac{3}{h_{j-1}}(a_j-a_{j-1})=c_{j+1}\cdot h_j+c_j\cdot(2\cdot h_j-h_{j-1}+h_{j-1})+c_{j-1}\cdot(h_{j-1}-2\cdot h_{j-1})$$

$$\frac{3}{h_j}(a_{j+1}-a_j)-\frac{3}{h_{j-1}}(a_j-a_{j-1})=c_{j+1}\cdot h_j+c_j\cdot 2\cdot h_j-c_{j-1}\cdot h_{j-1}$$

3.5 Curvas de Bézier

Antes de caracterizarmos as curvas de Bézier, relembremos os polinômios de Bernstein, definidos da seguinte forma:

$$B_i^n(t)=\binom{n}{i}t^i\cdot(1-t)^{n-i}$$

onde

$$\binom{n}{i}=\frac{n!}{i!(n-i)!}$$

A partir daí, podemos dar a seguinte definição para as curvas de Bézier (MADUREIRA; RINCON; SCHECHTER, 2013):

Definição 3.5.1. Seja $n \geq 1$ um número inteiro. Então, a curva de Bézier de grau n é dada por:

$$Q_n(t)=\sum_{i=0}^{n}B_i^n(t)\cdot P_i$$

onde $0 \leq t \leq 1$ e P_i são os chamados pontos de controle.

Assim, os polinômios de Bernstein são uma base para as curvas de Bézier. Em nosso trabalho, foram utilizadas as curvas cúbicas de Bézier, para as quais são necessários 4 pontos de controle. Estes pontos definem uma região chamada *polígono de controle* e, sabendo que as curvas de Bézier podem ser descritas por meio de polinômios de Bernstein, podemos inferir algumas propriedades acerca das mesmas (BERTKA, 2008):

1. $B_i^n(t) \in \mathbb{R}$
2. $B_0^k = B_k^k = 1$
3. $\sum B_i^k = 1$
4. $B_i^k > 0$

Pelo item 2, temos que a curva começa em P_0 e termina em P_n, e as condições 3 e 4 significam que cada ponto da curva de Bézier é uma média ponderada dos pontos de controle. Assim, a curva de Bézier entre os dois pontos P_0 e P_n está definida dentro do polígono de controle.

Podemos calcular a curva cúbica de Bézier sabendo que ela terá a seguinte forma:

$$Q_n(t) = \sum_{i=0}^{3} B_i^3(t) \cdot P_i$$

$$Q_3(t) = B_0^3 \cdot P_0 + B_1^3 \cdot P_1 + B_2^3 \cdot P_2 + B_3^3 \cdot P_3$$

Da definição do polinômio de Bernstein, temos:

$$B_0^3(t) = \binom{3}{0} t^0 \cdot (1-t)^{3-0} = (1-t)^3 = 1 - 3 \cdot t + 3 \cdot t^2 - t^3$$

$$B_1^3(t) = \binom{3}{1} t^1 \cdot (1-t)^{3-1} = 3 \cdot t \cdot (1-t)^2 = 3 \cdot t - 6 \cdot t^2 + 3 \cdot t^3$$

$$B_2^3(t) = \binom{3}{2} t^2 \cdot (1-t)^{3-2} = 3 \cdot t^2 \cdot (1-t) = 3 \cdot t^2 - 3 \cdot t^3$$

$$B_3^3(t) = \binom{3}{3} t^3 \cdot (1-t)^{3-3} = t^3$$

Assim, temos que:

$$Q_3(t) = (1 - 3 \cdot t + 3 \cdot t^2 - t^3) \cdot P_0 + (3 \cdot t - 6 \cdot t^2 + 3 \cdot t^3) \cdot P_1 + (3 \cdot t^2 - 3 \cdot t^3) \cdot P_2 + (t^3) \cdot P_3$$

Em notação matricial, podemos escrever:

$$Q_3(t) = \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} t^3 \\ t^2 \\ t \\ 1 \end{pmatrix} \cdot \begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{pmatrix}$$

O uso desta notação é útil para a implementação do código que cria a curva de Bézier adotado em nosso trabalho.

3.6 Polinômios de Chebyshev

Os polinômios de Chebyshev são polinômios ortogonais para o intervalo $(-1, 1)$, com relação à função peso $\omega(x) = (1 - x^2)^{-\frac{1}{2}}$. Eles são obtidos a partir de uma relação de

recorrência, que descreveremos a seguir, de acordo com Burden (2008, p. 480). Definimos, primeiramente, para $x \in [-1, 1]$, o seguinte polinômio:

$$T_n(x) = cos[n \cdot \text{arccos x}]$$

Podemos observar que:

$$T_0(x) = cos(0 \cdot \text{arccos x}) = 1$$
$$T_1(x) = cos(1 \cdot \text{arccos x}) = x$$

Para facilitar o desenvolvimento deste polinômio, substituímos arccos x por θ e definimos a seguinte função:

$$T_n(\theta) = cos(n \cdot \theta)$$

A partir daí, podemos obter uma relação de recorrência que resulta das relações que permitem calcular cossenos de somas e diferenças de arcos, observando que:

$$T_{n+1}(\theta) = cos((n+1) \cdot \theta)$$
$$T_{n+1}(\theta) = cos(n \cdot \theta + \theta)$$

$$T_{n+1}(x) = cos(n \cdot \theta) \cdot cos(\theta) - sen(n \cdot \theta) \cdot sen(\theta) \tag{3.9}$$

E, de modo análogo:

$$T_{n-1}(\theta) = cos((n-1) \cdot \theta)$$
$$T_{n-1}(\theta) = cos(n \cdot \theta - \theta)$$

$$T_{n-1}(x) = cos(n \cdot \theta) \cdot cos(\theta) + sen(n \cdot \theta) \cdot sen(\theta) \tag{3.10}$$

Somando membro a membro as equações (3.9) e (3.10), obtemos:

$$T_{n+1}(\theta) + T_{n-1}(\theta) = 2 \cdot cos(n \cdot \theta) \cdot cos(\theta)$$
$$T_{n+1}(\theta) = 2 \cdot cos(n \cdot \theta) \cdot cos(\theta) - T_{n-1}(\theta)$$

Desfazendo a substituição feita anteriormente, temos:

$$T_{n+1}(x) = 2 \cdot cos(n \cdot \text{arccos x}) \cdot cos(\text{arccos x}) - T_{n-1}(x)$$

Sabendo que $cos(\text{arccos x}) = x$, a equação anterior pode ser reescrita:

$$T_{n+1}(x) = 2 \cdot cos(n \cdot \text{arccos x}) \cdot x - T_{n-1}(x)$$

E ainda:

$$T_{n+1}(x) = 2 \cdot x \cdot cos(n \cdot \text{arccos x}) - T_{n-1}(x)$$

Daí, cada polinômio $T_{n+1}(x)$ pode ser obtido a partir dos anteriores como segue:

$$T_{n+1}(x) = 2 \cdot x \cdot T_n(x) - T_{n-1}(x)$$

Este polinômio possui aplicações em aproximação polinomial e também é utilizado de modo a diminuir os erros na interpolação, fornecendo um "posicionamento ótimo de pontos interpoladores para minimizar o erro na interpolação de Lagrange"(BURDEN, 2008, p. 482). No entanto, a escolha de utilizar o polinômio de Chebyshev para tratar séries com falhas se deve ao fato de que, para $x \in [-1, 1]$, temos $T_n(x) \in [-1, 1]$ o que nos permite realizar uma adaptação para atender aos nossos propósitos, como descrevemos no próximo capítulo.

Para obter o polinômio cúbico, basta seguir a relação de recorrência dada:

$$T_0(x) = 1$$

$$T_1(x) = x$$

$$T_2(x) = 2 \cdot x \cdot x - 1 = 2 \cdot x^2 - 1$$

$$T_3(x) = 2 \cdot x \cdot (2 \cdot x^2 - 1) - x = 4 \cdot x^3 - 2 \cdot x - x = 4 \cdot x^3 - 3 \cdot x$$

O polinômio $T_3(x)$ obtido anteriormente foi utilizado nos testes que expomos a seguir. A infinidade de polinômios que podem ser obtidos desta forma oferece, consequentemente, infinitas possibilidades do uso dos mesmos nesta e em outras aplicações.

4 Resultados e discussões

Neste capítulo, apresentamos os procedimentos adotados para realização dos testes de interpolação e análise de séries temporais com falhas e seus resultados, destinados a responder à questão inicial desta pesquisa.

4.1 Metodologia adotada no trabalho

Em nosso trabalho, realizamos testes envolvendo interpolação e análise de séries temporais com falhas, utilizando a TWC, o MATLAB e o pacote YAWTB, abordados no capítulo 2, e os métodos de interpolação descritos no capítulo 3. Tais testes foram feitos a partir de um sinal sintético, um conjunto de 2500 dados sem falhas; em outras palavras, todas as suas entradas são numéricas. Após realizarmos a leitura do arquivo referente a estes dados no programa MATLAB, introduzimos três tipos de falhas no sinal de acordo com o seguinte critério:

1. Tipo I: As falhas deste tipo possuem no máximo 1% do tamanho total do sinal. Em nosso caso, essas lacunas nos dados são consideradas "falhas pequenas" e possuem extensão de 25 entradas. Foram introduzidas 10 falhas deste tipo em nosso sinal sintético, espalhadas por toda a sua extensão;

2. Tipo II: São falhas consideradas médias em nosso trabalho, que possuem aproximadamente 5% do tamanho total do sinal, ou seja, correspondem a 125 entradas. Aqui, utilizamos 8 falhas deste tipo;

3. Tipo III: O terceiro tipo de falha corresponde a 10% do tamanho total do sinal, ou seja, 250 entradas. Tais falhas, consideradas "grandes" em nosso trabalho, foram introduzidas em dois locais diferentes.

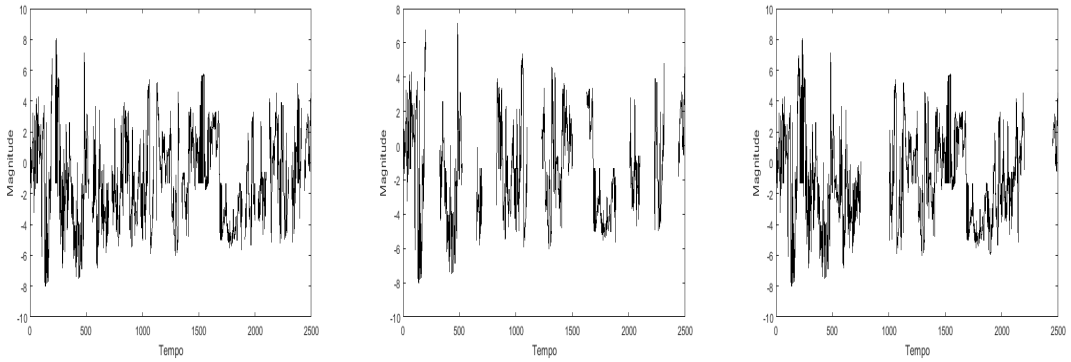
A classificação das falhas proposta anteriormente é a mesma utilizada no trabalho *Análise tempo-escala de séries temporais de geofísica espacial com lacunas: estudo de caso* (MAGRINI; DOMINGUES; MENDES, 2016). As lacunas foram espalhadas de modo arbitrário em toda a extensão do sinal, seguindo apenas os critérios citados anteriormente e mais três condições: duas falhas não devem aparecer juntas, não temos falhas no começo e/ou no final do sinal e cada padrão de falha introduzido possui lacunas de um único tipo. O número de falhas de cada tipo foi escolhido de modo a gerar alterações visíveis no espectro global e nas estruturas de energia do escalograma.

Resumidamente, os padrões de falha foram elaborados da seguinte forma:

Tabela 4.1 – Padrões de falha adotados

Tipo de falha	Quantidade	Intervalos com falha	Porcentagem do sinal correspondente
Tipo I	10	$[201, 225]$;$[526, 550]$; $[701, 725]$;$[1101, 1125]$; $[1326, 1350]$;$[1886, 1910]$; $[2001, 2025]$; $[2101, 2125]$; $[2301, 2325]$;$[2416, 2440]$	10
Tipo II	8	$[201, 325]$; $[526, 650]$; $[701, 825]$; $[1101, 1225]$; $[1501, 1625]$;$[1886, 2010]$; $[2101, 2225]$; $[2316, 2440]$	40
Tipo III	2	$[751, 1000]$; $[2201, 2450]$	20

Os padrões de falha elaborados geraram as seguintes séries temporais:



(a) Falhas pequenas (b) Falhas médias (c) Falhas grandes

Figura 4.10 – Padrões de falhas

Em seguida, utilizamos em cada série encontrada os métodos de interpolação vistos anteriormente. No caso do polinômio cúbico de Hermite, foi gerado, para cada falha, um polinômio que atendesse às condições referentes ao método, ou seja, tais polinômios deveriam coincidir com as extremidades das falhas, e suas derivadas coincidem com as derivadas do sinal original nestes mesmos pontos. Já no caso da interpolação linear, o procedimento consistiu em criar uma reta que passasse pelas extremidades de cada lacuna. Para utilizar o *spline* cúbico, procedemos de forma análoga à interpolação pelo polinômio cúbico de Hermite, ou seja, foram criados polinômios para cada intervalo com falha, respeitando as condições impostas pelo método. Para interpolar utilizando a curva cúbica de Bézier, definimos 4 pontos de controle, sendo P_0 o ponto em que a falha se inicia, P_1 e P_2 pontos obtidos atribuindo ao valor central do intervalo com falha os valores máximo e mínimo observados na série, respectivamente, e P_4 a outra extremidade do intervalo com

falha. Definindo o polinômio cúbico de Chebyshev no intervalo $(-1, 1)$, obtivemos um vetor u, que utilizamos para interpolar nossas séries. Para cada intervalo de tempo (a, b) em que se tem uma falha, sendo $[y(a), y(b)]$ os valores indefinidos no sinal, realizamos a seguinte transformação:

$$[y(a), y(b)] = \frac{1}{2}[(b - a) \cdot u + a + b]$$

Após realizarmos os procedimentos descritos anteriormente em cada padrão de falha, analisamos com a TWC cada uma das 15 séries obtidas, além da série original, utilizando o MATLAB e o pacote YAWTB. A função *wavelet* utilizada foi a *wavelet* de Morlet, descrita brevemente no capítulo 2, de modo a obter o escalograma e o espectro global referentes a cada série.

Para verificarmos qual método interpolatório apresenta resultados mais próximos do original, comparamos os escalogramas referentes a cada método, para cada tipo de falha, com o escalograma do sinal original, obtendo informações acerca de alterações nas estruturas de energia, na concentração da mesma no decorrer do tempo, além de apresentar as escalas que são mais afetadas. Ainda em relação ao escalograma, observamos a barra de energia, que indica a quantidade de energia representada por cada cor no escalograma e nos dá parâmetros para verificar a conservação de energia no sentido de Parseval, conforme o capítulo 2. Para a visualização destas informações, utilizamos a palheta de cor conhecida como *jet*. Comparamos também os espectros globais, que se relacionam à energia global por escala presente no sinal. Esta comparação nos permite identificar alterações mais sutis que são apresentadas no escalograma por pequenas variações de cor.

Nas próximas seções, apresentaremos a representação gráfica das séries interpoladas, os escalogramas e espectros globais referentes a cada tipo de falha obtidos com os cinco métodos de interpolação adotados. No início de cada seção, também trazemos os resultados obtidos com o sinal original, de modo a facilitar a comparação e a compreensão das conclusões apresentadas.

4.2 Falhas pequenas

Tabela 4.2 – Energia dos sinais interpolados (Tipo I)

Comparação de energia (energia do sinal original = $6,43 \cdot 10^8$)		
Método interpolatório	Energia total do sinal interpolado	Energia original representada (%)
Curva de Bézier	$5,63 \cdot 10^8$	87,58
Polinômio cúbico de Chebyshev	$6,48 \cdot 10^8$	100,88
Polinômio cúbico de Hermite	$6,59 \cdot 10^8$	102,58
Interpolação linear	$6,50 \cdot 10^8$	101,19
Splines cúbicos	$1,03 \cdot 10^9$	160,26

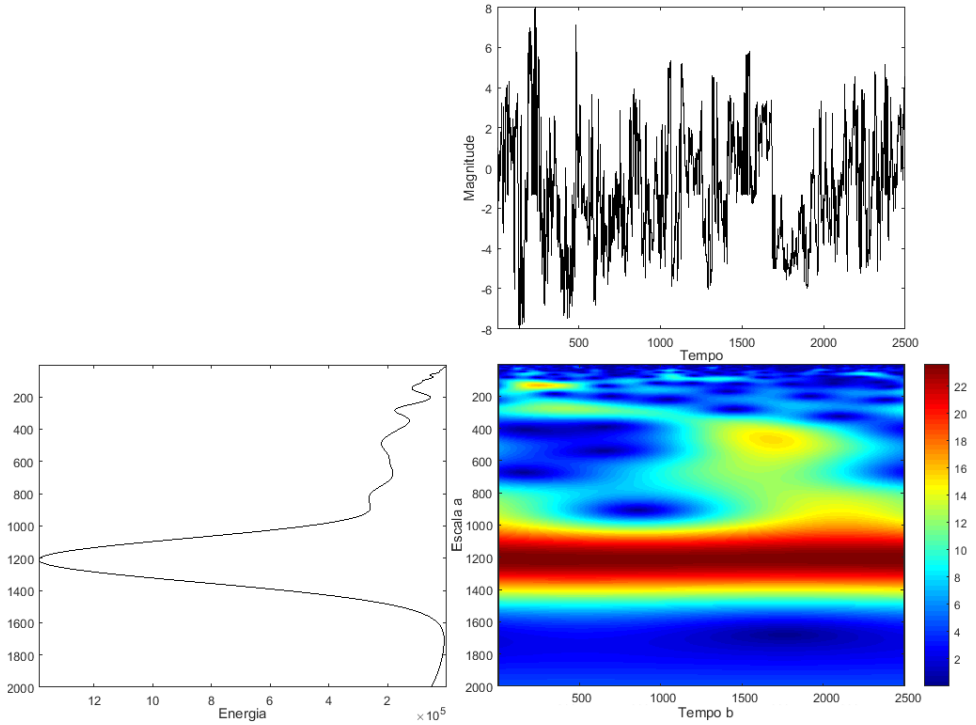


Figura 4.11 – Série original

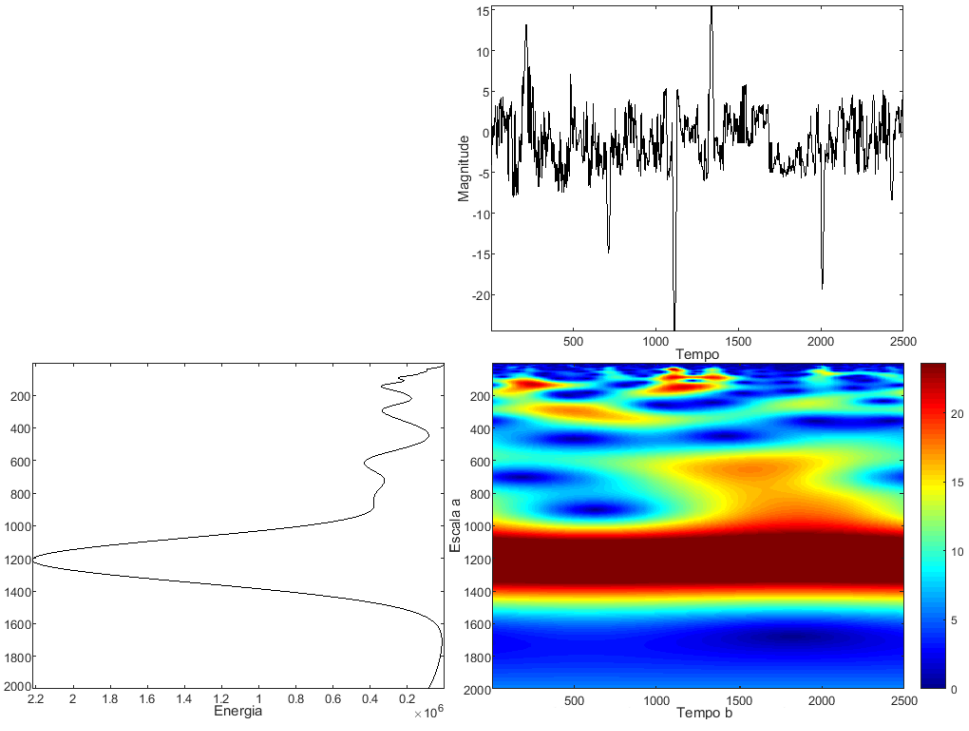


Figura 4.12 – Falhas do tipo I - *Spline* cúbico

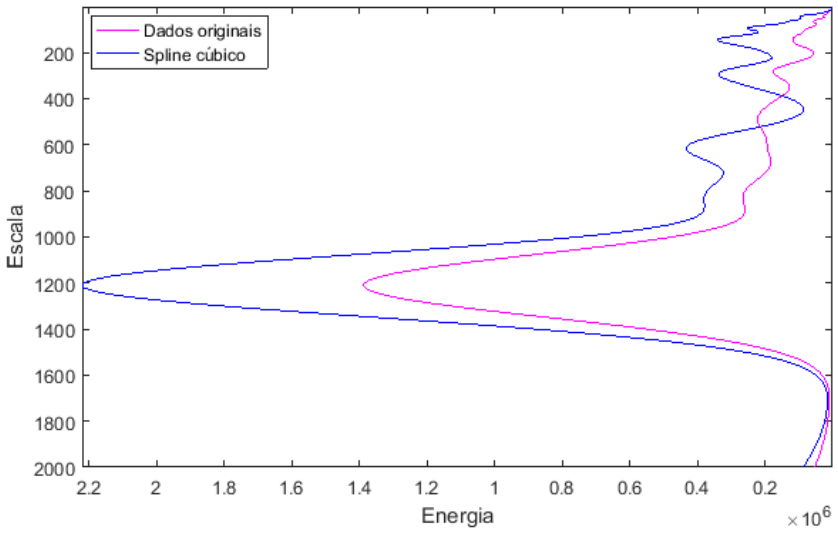


Figura 4.13 – Comparação entre o espectro global original e o interpolado com *spline* cúbico (Tipo I)

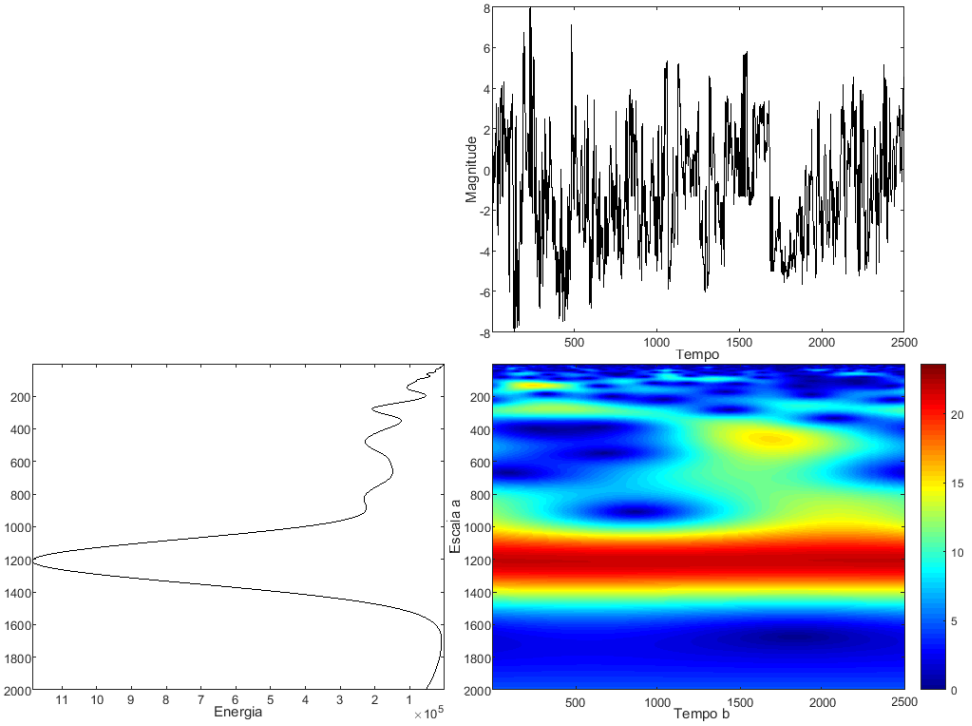


Figura 4.14 – Falhas do tipo I - Curva de Bézier

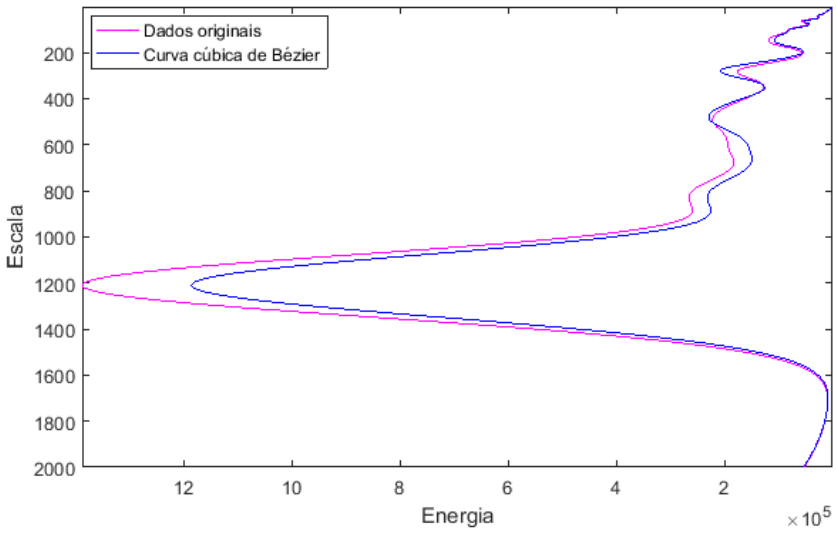


Figura 4.15 – Comparação entre o espectro global original e o interpolado com curva de Bézier (Tipo I)

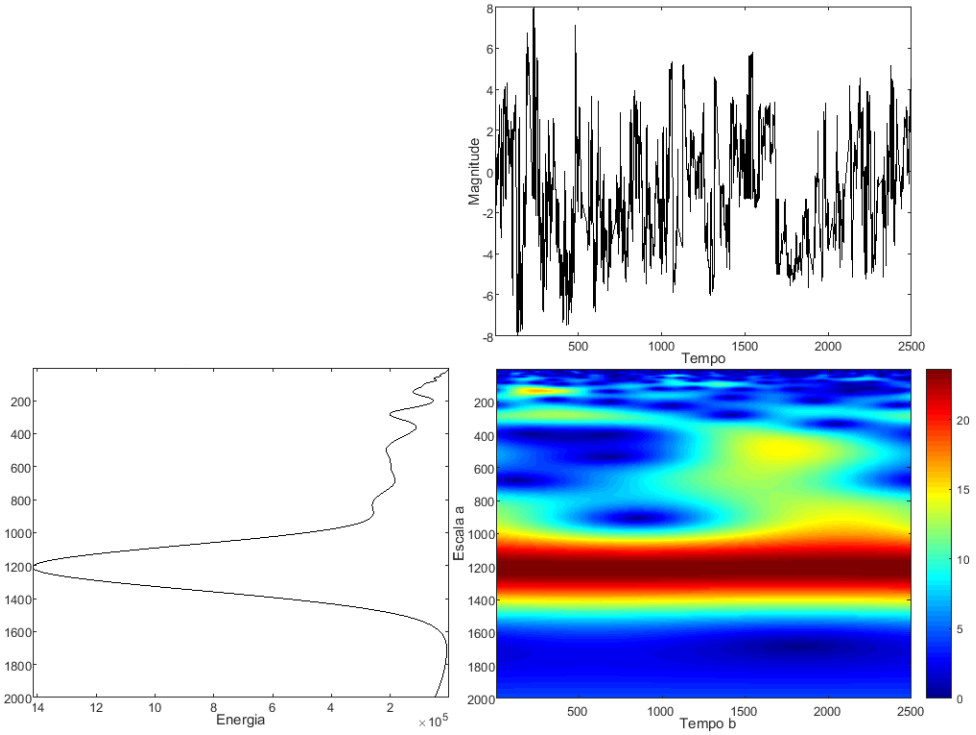


Figura 4.16 – Falhas do tipo I - Interpolação linear

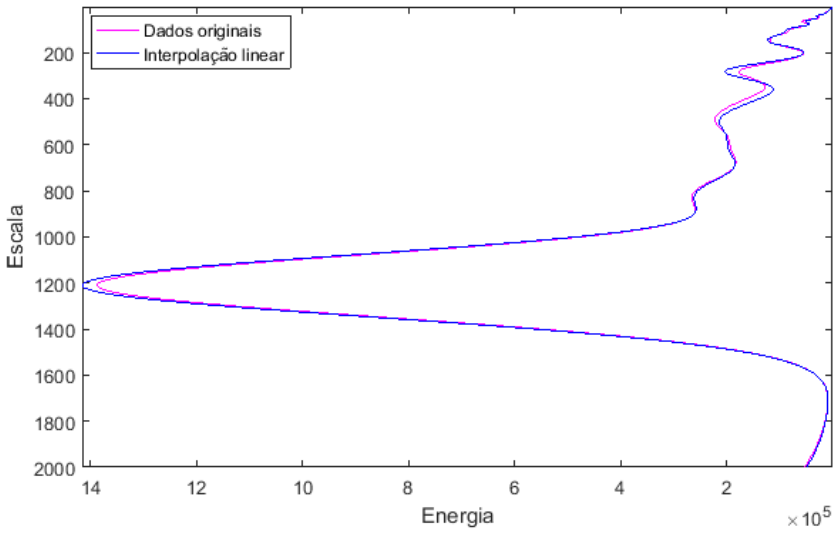


Figura 4.17 – Comparação entre o espectro global original e o interpolado com interpolação linear (Tipo I)

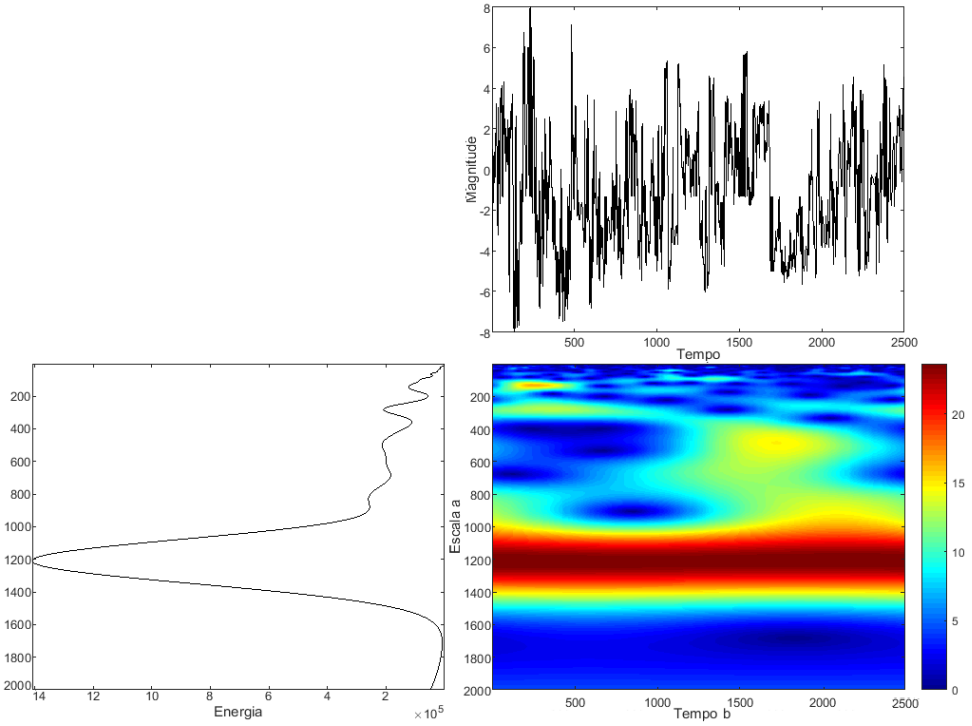


Figura 4.18 – Falhas do tipo I - Polinômio de Chebyshev

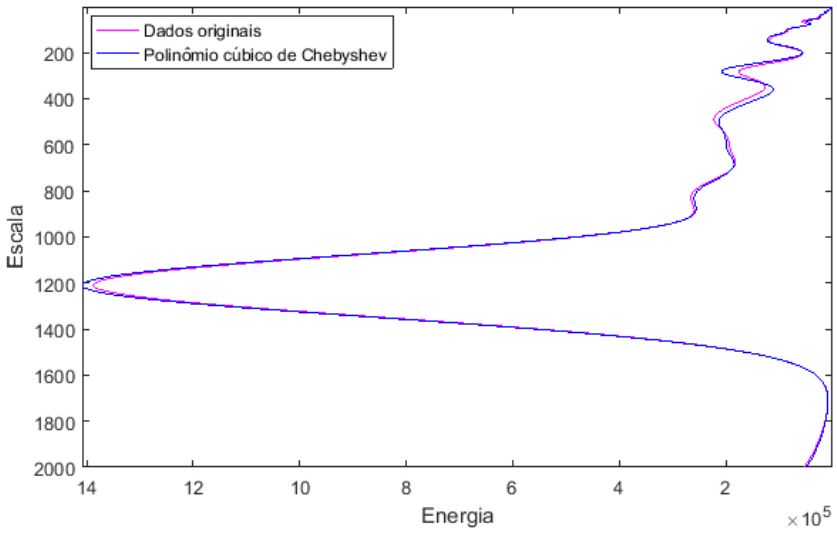


Figura 4.19 – Comparação entre o espectro global original e o interpolado com polinômio de Chebyshev (Tipo I)

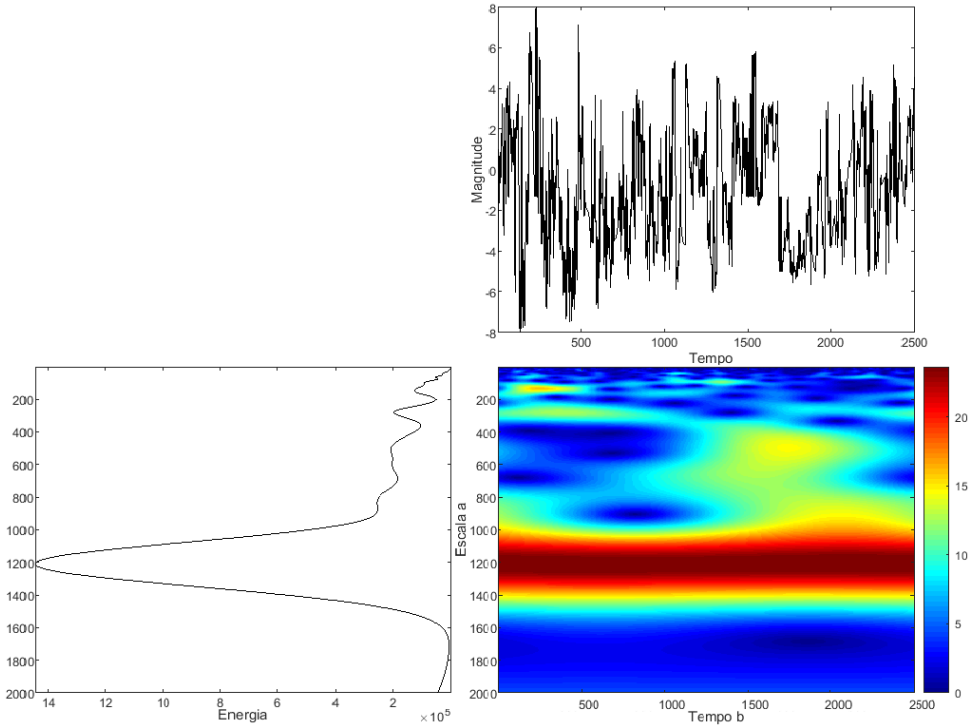


Figura 4.20 – Falhas do tipo I - Polinômio cúbico de Hermite

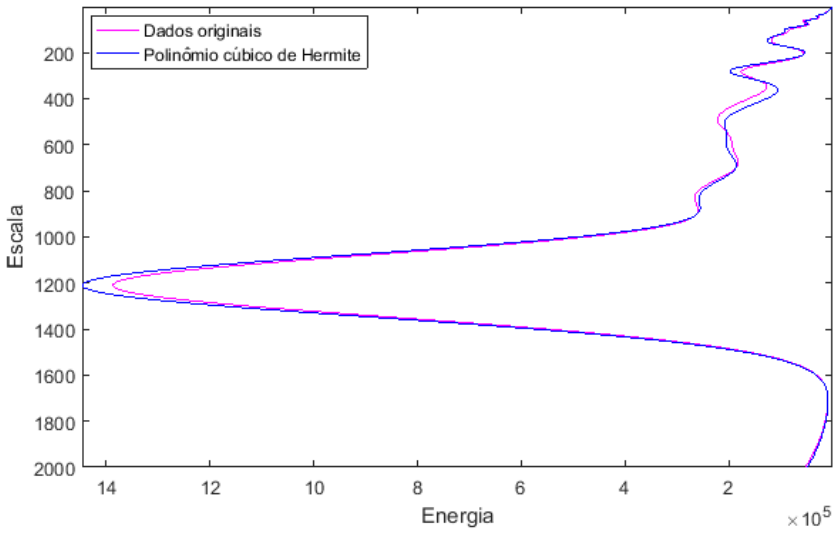


Figura 4.21 – Comparação entre o espectro global original e o interpolado com polinômio cúbico de Hermite (Tipo I)

Como podemos observar na comparação entre os escalogramas referentes à série original e a interpolada com *splines* cúbicos, nos escalogramas das imagens 4.11 e 4.12, as estruturas de energia não são conservadas; um exemplo bastante visível ocorre próximo à escala 600, na qual se vê uma faixa que não corresponde aos valores originais. Algo semelhante ocorre próximo à escala 200, para t próximo de 2000. É possível ver que algumas estruturas foram deslocadas no tempo, como ocorre nas escalas entre 800 e 1000. Os espectros globais da figura 4.13, por sua vez, permitem observar que o método interpolatório utilizado resulta em superestimação da energia em grande parte dos picos, que nos oferecem alguns parâmetros para análise e comparação. Estes picos não coincidem e não permitem, portanto, inferir algo sobre a energia do sinal original.

A interpolação com a curva cúbica de Bézier permite a conservação de estruturas de energia, mas não da quantidade de energia associada, conforme o escalograma da imagem 4.14, onde podemos observar que as escalas cujas energias são mais altas se preservam em comparação ao escalograma original. Na figura 4.15, percebemos que a energia global do sinal interpolado se comporta de forma diferente do que ocorre no sinal sintético, principalmente nas escalas de 300 a 600, onde ocorrem mudanças mais abruptas. Além disso, podemos ver que a quantidade de energia associada a grande parte das escalas é menor do que no original, refletindo a diferença da energia global do sinal descrita na tabela 4.2.

No caso da interpolação linear, cujos resultados estão apresentados na figura 4.16, podemos observar que a energia máxima detectada se preserva em relação à energia original, bem como as estruturas que constam no escalograma. Nele, as energias observadas são ligeiramente mais intensas do que vemos no escalograma da série original, na figura 4.11, e essa diferença se reflete no espectro global, como podemos ver nos gráficos da figura 4.17, principalmente nas escalas de 400 a 1000. Como veremos a seguir, o resultado obtido com este método para o tratamento de falhas pequenas se assemelha ao que encontramos com o uso do polinômio cúbico de Chebyshev aplicado ao mesmo caso.

A interpolação por polinômio de Chebyshev permite a conservação da energia do sinal, bem como das estruturas do escalograma. No entanto, assim como no caso anterior, algumas regiões têm suas intensidades alteradas, como vemos na figura 4.18. Tais diferenças, no entanto, não são tão expressivas, como vemos nos espectros globais da figura 4.19. Os picos de energia ocorrem nas mesmas escalas e a quantidade de energia é próxima nestes pontos. As diferenças ocorrem nas mesmas escalas do caso linear: entre 400 e 1000.

O último caso é o polinômio cúbico de Hermite. O uso deste polinômio interpolador possibilitou a conservação da energia e as estruturas observadas no escalograma da série original. As diferenças observadas no espectro global, como podemos observar na imagem 4.21, são bastante sutis, uma vez que os picos de energia coincidem e a energia global verificada difere levemente nas escalas de 300 a 400, de 600 a 900 e em torno de 1200.

Comparando todos os resultados obtidos, podemos ver que o polinômio cúbico de Hermite é o que fornece resultados mais próximos do esperado, preservando grande parte das características da série original, seguido da interpolação linear e do polinômio de Chebyshev, que apresentam resultados semelhantes entre si e cujos principais problemas são pequenas alterações de energia em escalas intermediárias. O polinômio de Chebyshev, conforme podemos ver na tabela 4.2, é o método que possui a melhor conservação da energia global do sinal original, seguido da interpolação linear. Em seguida, temos a curva de Bézier, que preserva as estruturas de energia mas apresenta uma pequena perda na energia global do sinal. Por fim, o resultado mais distante do original foi obtido com os *splines* cúbicos, cujos picos de energia não coincidem com o original e cuja energia total, de acordo com as comparações realizadas, corresponde a $160,26\%$ da energia esperada.

4.3 Falhas médias

Tabela 4.3 – Energia dos sinais interpolados (Tipo II)

Comparação de energia (energia do sinal original = $6,43 \cdot 10^8$)		
Método interpolatório	Energia total do sinal interpolado	Energia original representada (%)
Curva de Bézier	$5,03 \cdot 10^8$	78,22
Polinômio cúbico de Chebyshev	$8,29 \cdot 10^8$	128,99
Polinômio cúbico de Hermite	$1,01 \cdot 10^9$	157,42
Interpolação linear	$8,67 \cdot 10^8$	134,98
Splines cúbicos	$6,30 \cdot 10^{10}$	9807,10

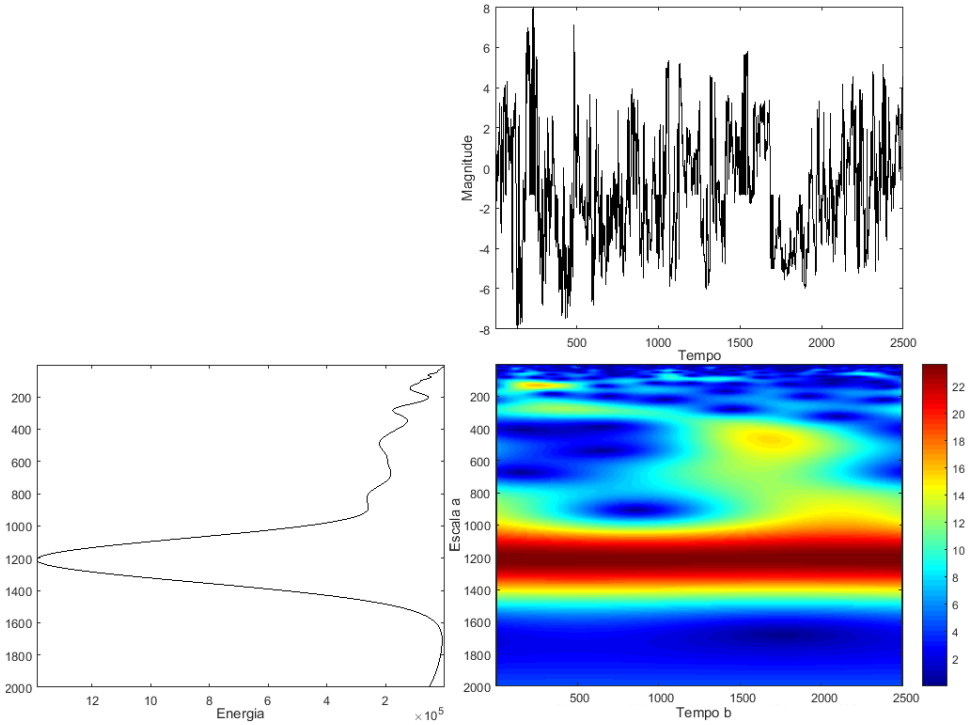


Figura 4.22 – Série original

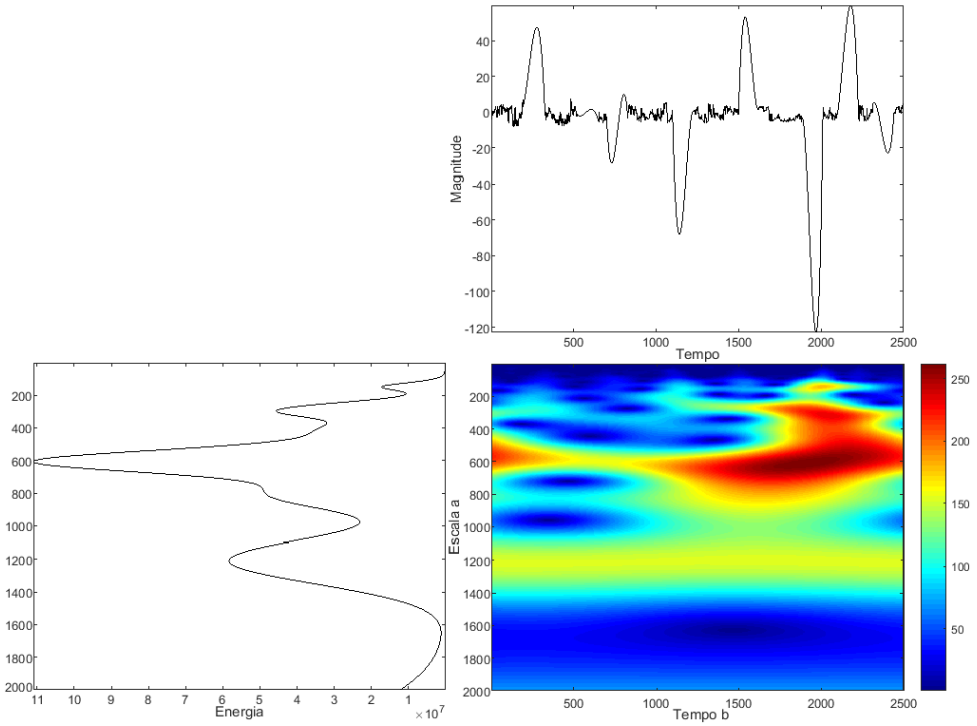


Figura 4.23 – Falhas do tipo II - *Spline* cúbico

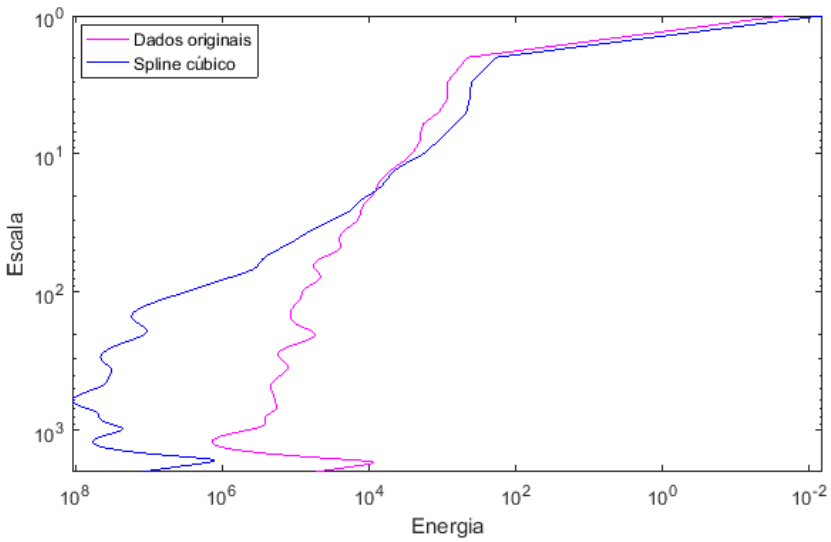


Figura 4.24 – Comparação entre o espectro global original e o interpolado com *spline* cúbico (Tipo II)

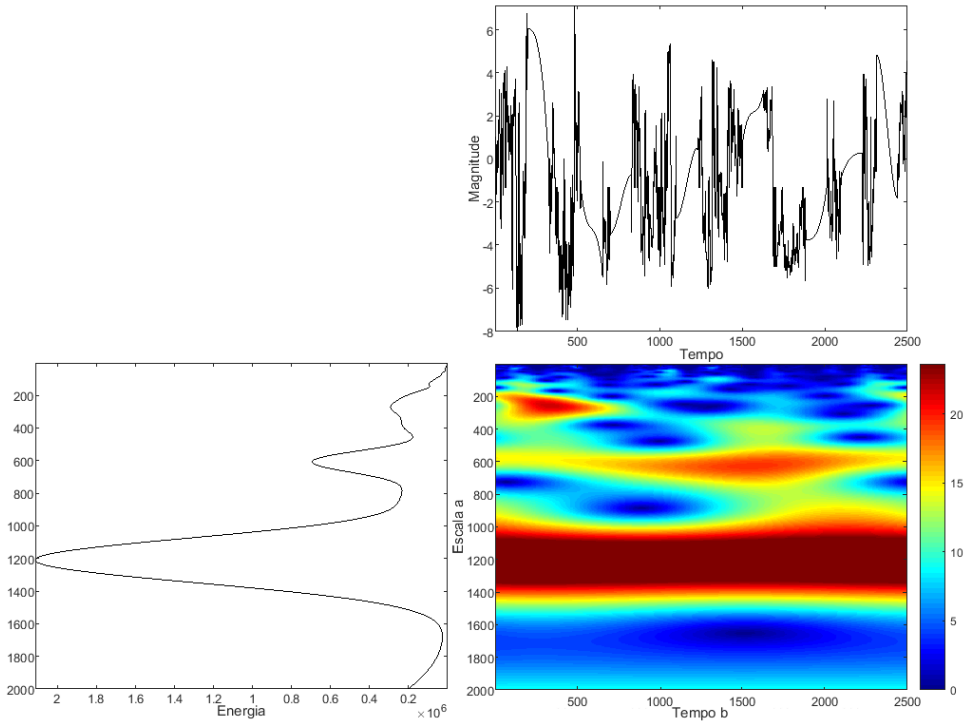


Figura 4.25 – Falhas do tipo II - Polinômio cúbico de Hermite

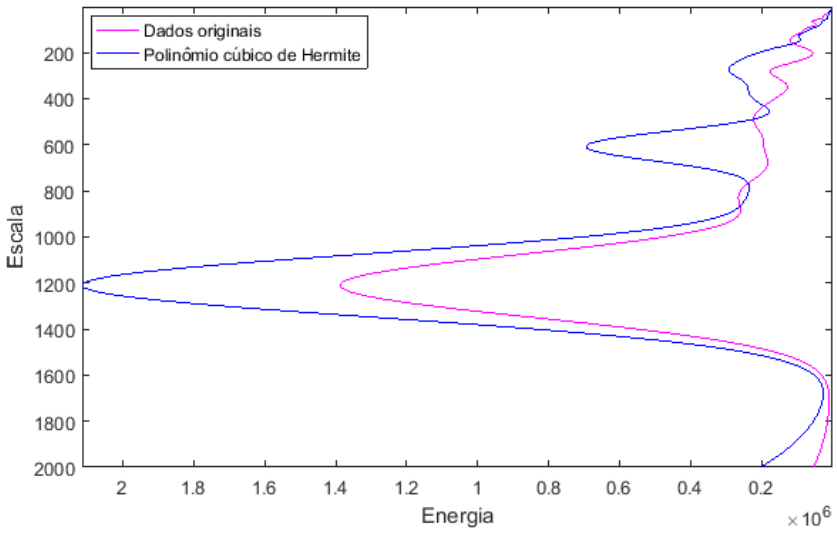


Figura 4.26 – Comparação entre o espectro global original e o interpolado com polinômio cúbico de Hermite (Tipo II)

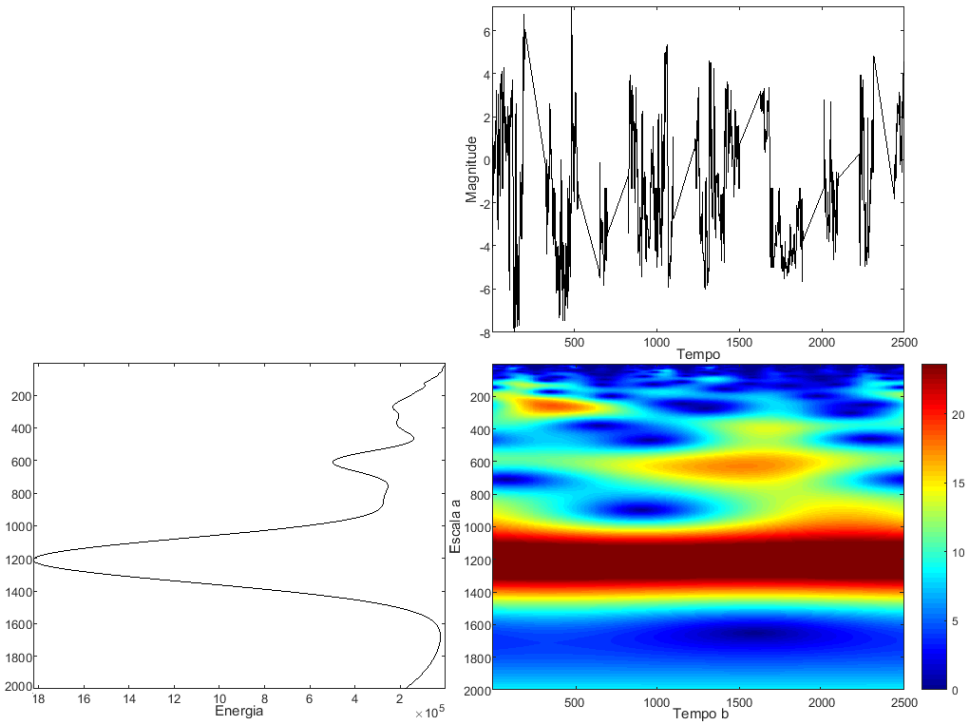


Figura 4.27 – Falhas do tipo II - Interpolação linear

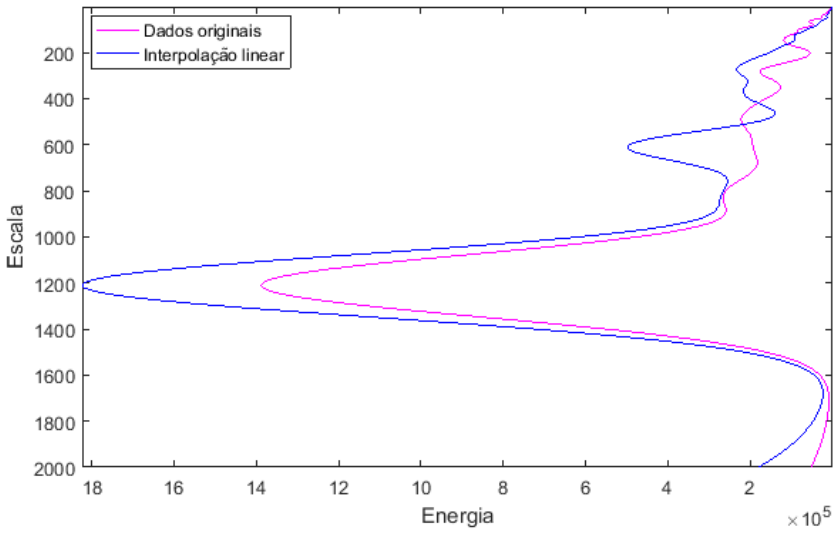


Figura 4.28 – Comparação entre o espectro global original e o interpolado com interpolação linear (Tipo II)

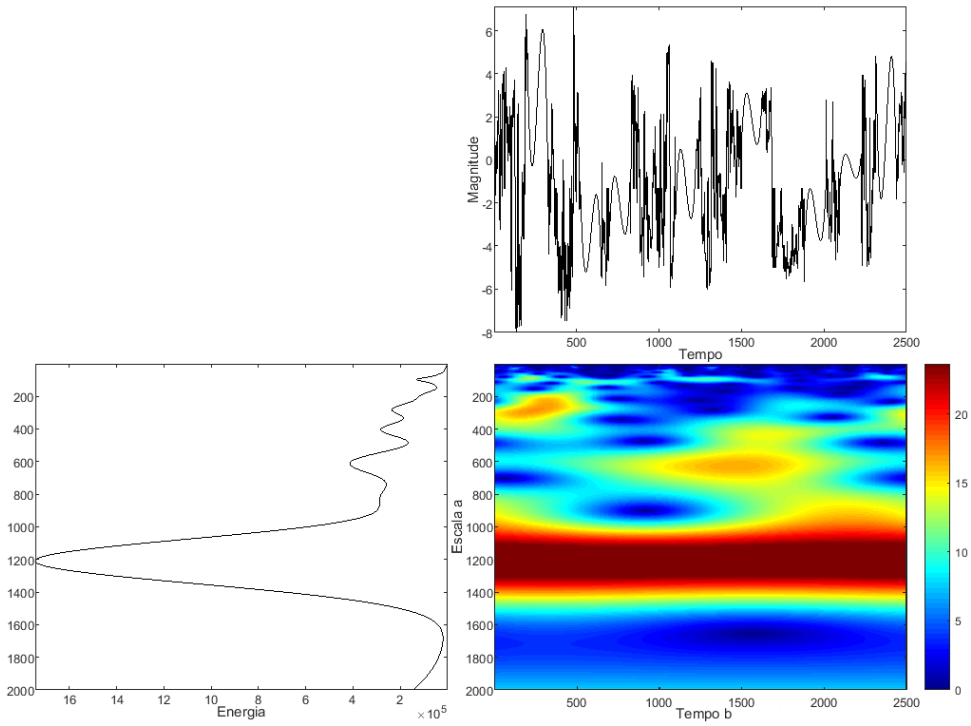


Figura 4.29 – Falhas do tipo II - Polinômio de Chebyshev

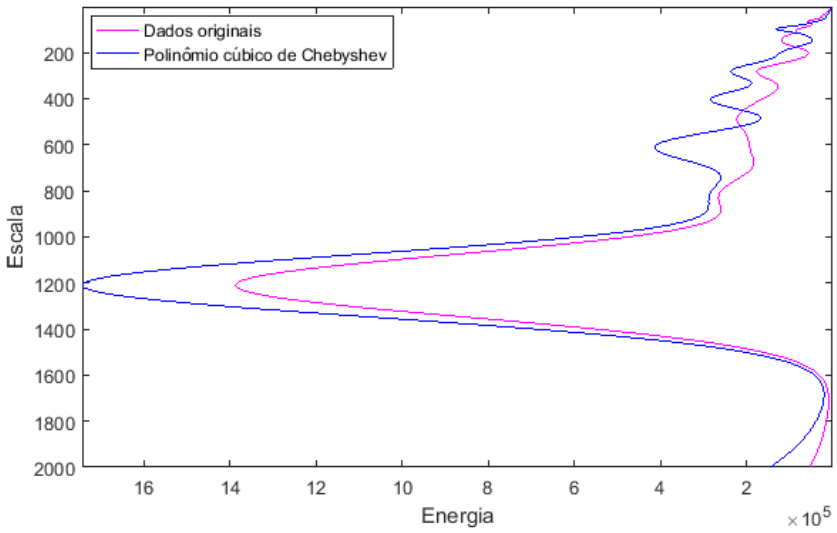


Figura 4.30 – Comparação entre o espectro global original e o interpolado com polinômio de Chebyshev (Tipo II)

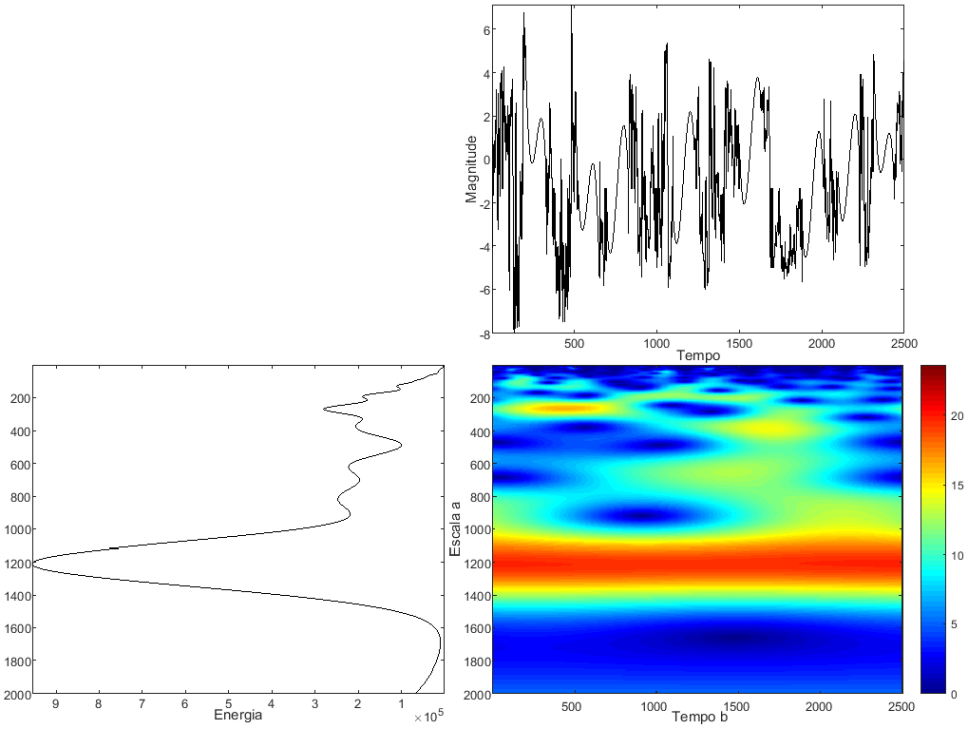


Figura 4.31 – Falhas do tipo II - Curva de Bézier

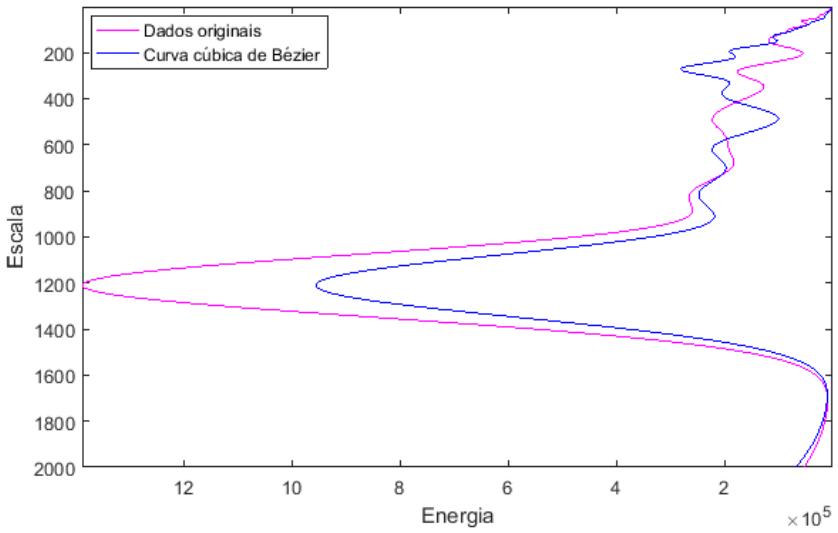


Figura 4.32 – Comparação entre o espectro global original e o interpolado com curva de Bézier (Tipo II)

Dentre as interpolações utilizadas no tratamento da série com falhas médias, a interpolação com *splines* cúbicos é a que apresenta resultados mais distantes do original. As barras de energia apresentam disparidades e o mesmo se reflete na energia total do sinal, expressa na tabela 4.3, e também no escalograma e suas estruturas. Vemos que a estrutura mais acentuada no escalograma, que é uma faixa entre as escalas 1000 e 1400, é mantida de forma mais suave, enquanto vemos mais energia nas escalas próximas de 600. É importante observar que os *splines* cúbicos foram o único método cujo escalograma não pôde ser gerado com uma barra de energia semelhante à original, pois isso dificultaria a observação das estruturas, dada a diferença de energia verificadas nos dois casos. A comparação entre os espectros globais da figura 4.24 mostra que a análise do sinal interpolado não permite inferir algo sobre o sinal sem falhas, uma vez que a diferença dos dados apresentados em ambos os gráficos é bastante significativa.

Um resultado um pouco melhor pode ser observado no caso do polinômio cúbico de Hermite, conforme o escalograma da figura 4.25, embora este também apresente diferenças em relação ao original. As divergências observadas com mais clareza são, principalmente, nas escalas próximas de 600, cujas estruturas são mais acentuadas do que as referentes às escalas próximas de 400. É possível ver que, além destas, as escalas entre 200 e 400 também apresentam diferenças. A divergência relacionada à escala 600 também aparece no espectro global, como podemos verificar com mais clareza na figura 4.26, que apresenta um pico de energia distante do original. Pelo espectro global vemos também que a energia apresentada pelo sinal interpolado é menor do que a original em algumas escalas, como no intervalo de 0 a 200 e por volta de 500 e 800, mas nas demais ocorre o inverso, ou seja, a original apresenta menos energia. A energia global apresentada na tabela 4.3 também apresenta divergências, com o segundo valor mais distante do original.

Analisando o resultado obtido por meio da interpolação linear, apresentado na figura 4.27, vemos que algumas estruturas aparecem alteradas e distorcidas, de modo que a energia aparenta estar menos concentrada no tempo. Isso ocorre, principalmente, entre as escalas 200 e 800. No espectro global da figura 4.28, vemos que as escalas de 200 a 400 e de 500 a 800 exibem divergências em relação aos picos de energia global presentes no espectro original. A escala 600, inclusive, apresenta um pico de energia que não corresponde ao sinal sintético.

A interpolação pelo polinômio cúbico de Chebyshev apresenta resultados semelhantes à interpolação linear: a energia também é superestimada e existem estruturas distorcidas no escalograma, em escalas próximas às observadas no caso linear. As estruturas referentes às escalas acima de 800 se preservaram mas, por meio do espectro global (figura 4.30, é possível ver a diferença entre a energia verificada nestas escalas na série original e na interpolada. Alguns picos de energia diferem nos dois casos, como nas escalas 100, 500 e 600, enquanto outros coincidem mas diferem em intensidade, como, por exemplo, nas escalas 300 e 1200. É importante observar que a interpolação por polinômio de Chebyshev

foi o método que apresentou a segunda melhor preservação da energia original, conforme a tabela 4.3.

No caso da curva cúbica de Bézier, podemos observar, primeiramente, que a quantidade de energia detectada não se preserva. Como vemos no escalograma da figura 4.31, algumas estruturas de energia se preservam com alterações na intensidade, enquanto outras aparecem de forma distorcida, fazendo com que a energia se apresente de forma mais espalhada no decorrer do tempo em uma mesma escala. Isso ocorre, por exemplo, nas escalas de 600 a 800. A perda de energia pode ser vista nos espectros globais da figura 4.32, uma vez que os únicos intervalos que apresentam energia maior na série interpolada do que na original são entre 150 e 400 e entre 600 e 800. Para as escalas acima de 800, os picos de energia coincidem e, em relação às demais, as principais divergências são entre as escalas 200 e 500. Pela tabela 4.3, podemos ver que este foi o único método em que houve perda de energia, embora tenha sido o caso em que a energia global do sinal interpolado mais se aproximou da energia total do sinal original.

Tendo em vista as análises realizadas, vemos que, no caso das falhas médias, um dos métodos interpolatórios que forneceram resultados mais próximos do original foi a curva de Bézier. Este caso, inclusive, foi o único que não apresentou alterações nas escalas mais altas, acima de 1600, característica observada em todos os demais métodos. É importante observar, no entanto, que o polinômio de Chebyshev também apresentou resultados interessantes, apesar das divergências em altas escalas e, para algumas escalas, representou melhor a energia do sinal original do que a curva de Bézier. Assim, os dois métodos tiveram resultados interessantes, apresentando semelhanças e diferenças em relação ao resultado esperado, de modo que seu uso seria determinado pelo uso que se deseja fazer dos dados. Se procurássemos observar as escalas mais altas, a curva de Bézier apresentaria melhores resultados, enquanto para escalas intermediárias e para a representação da energia original, o melhor método seria o polinômio de Chebyshev.

Os resultados obtidos com a interpolação linear e o polinômio cúbico de Hermite apresentaram maior diferença em relação ao sinal sintético, mas permitiriam obter algumas informações sobre o mesmo, como a energia associada a certas escalas. O método de interpolação por *splines* cúbicos, por sua vez, apresentou novamente os resultados mais distantes do esperado, divergindo principalmente em relação ao espectro global.

4.4 Falhas grandes

Tabela 4.4 – Energia dos sinais interpolados (Tipo III)

Comparação de energia (energia do sinal original = $6,43 \cdot 10^8$)		
Método interpolatório	Energia total do sinal interpolado	Energia original representada (%)
Curva de Bézier	$7,58 \cdot 10^8$	117,95
Polinômio cúbico de Chebyshev	$7,73 \cdot 10^8$	120,31
Polinômio cúbico de Hermite	$6,21 \cdot 10^8$	96,60
Interpolação linear	$7,19 \cdot 10^8$	111,93
Splines cúbicos	$2,17 \cdot 10^{11}$	33817,31

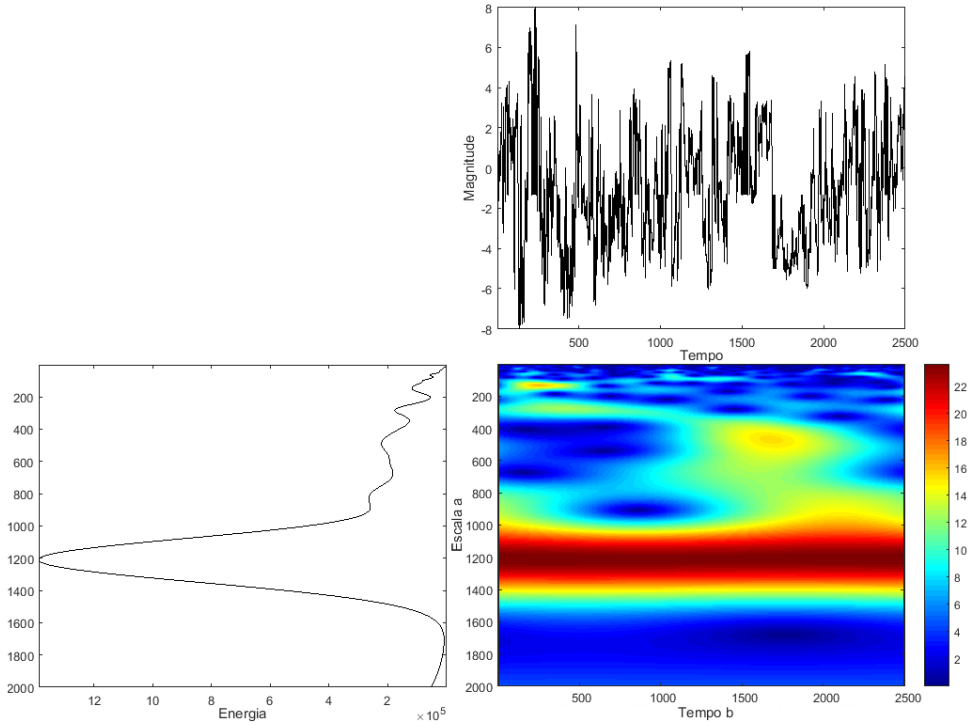


Figura 4.33 – Série original

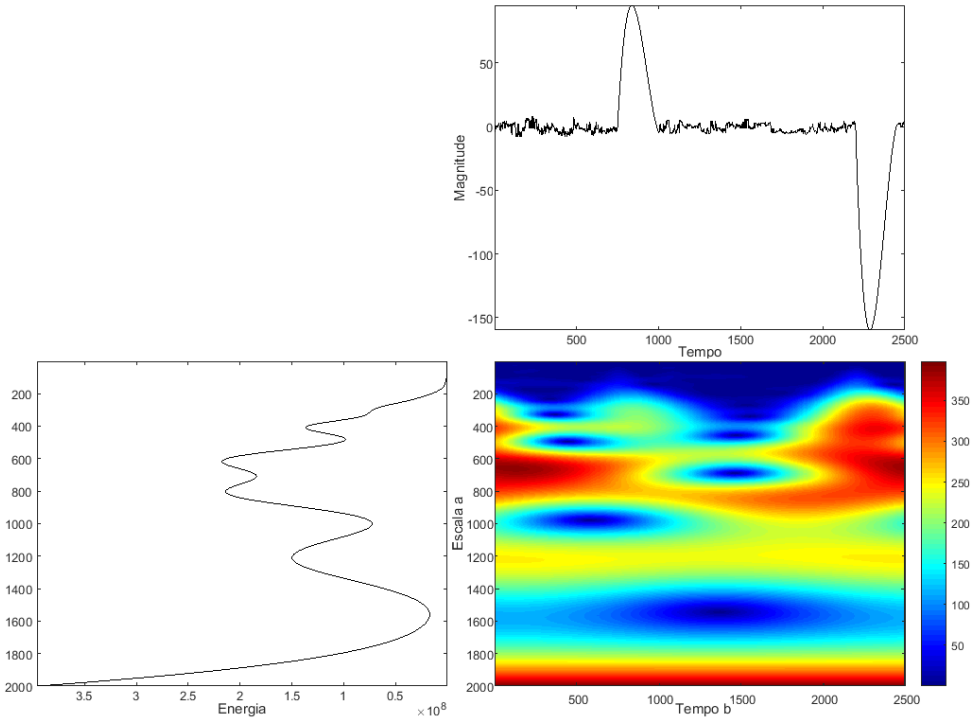


Figura 4.34 – Falhas do tipo III - *Spline* cúbico

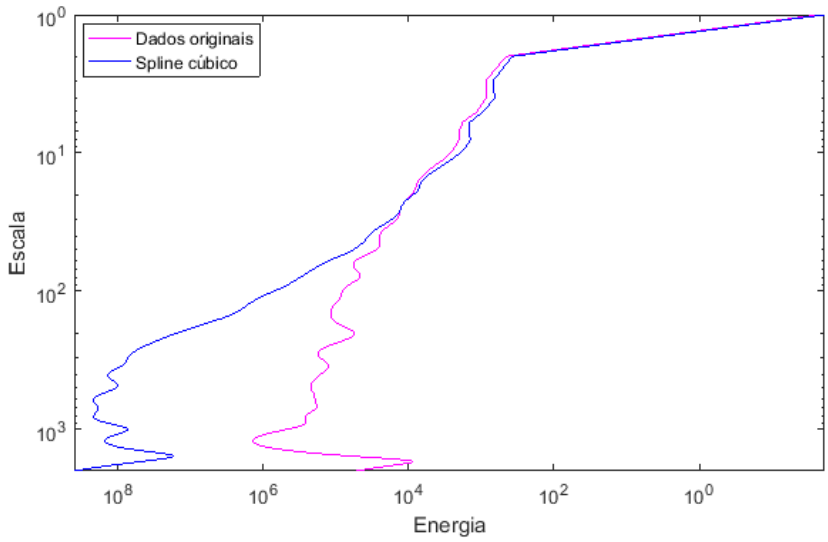


Figura 4.35 – Comparação entre o espectro global original e o interpolado com *spline* cúbico (Tipo III)

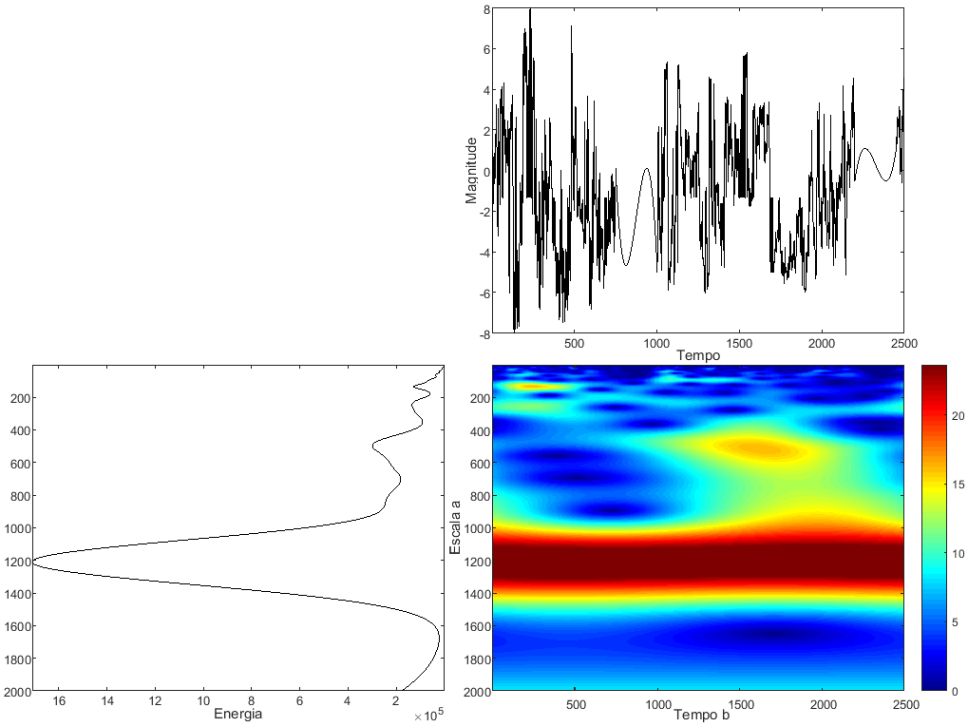


Figura 4.36 – Falhas do tipo III - Polinômio de Chebyshev

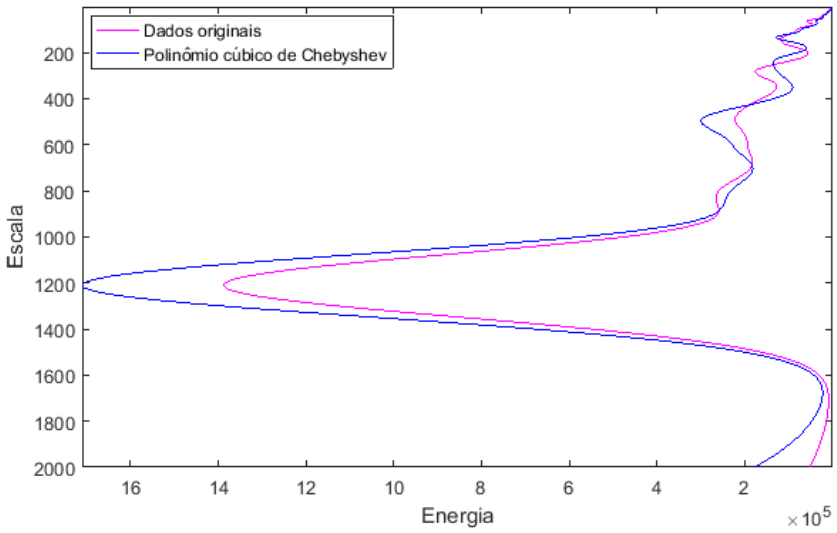


Figura 4.37 – Comparação entre o espectro global original e o interpolado com polinômio de Chebyshev (Tipo III)

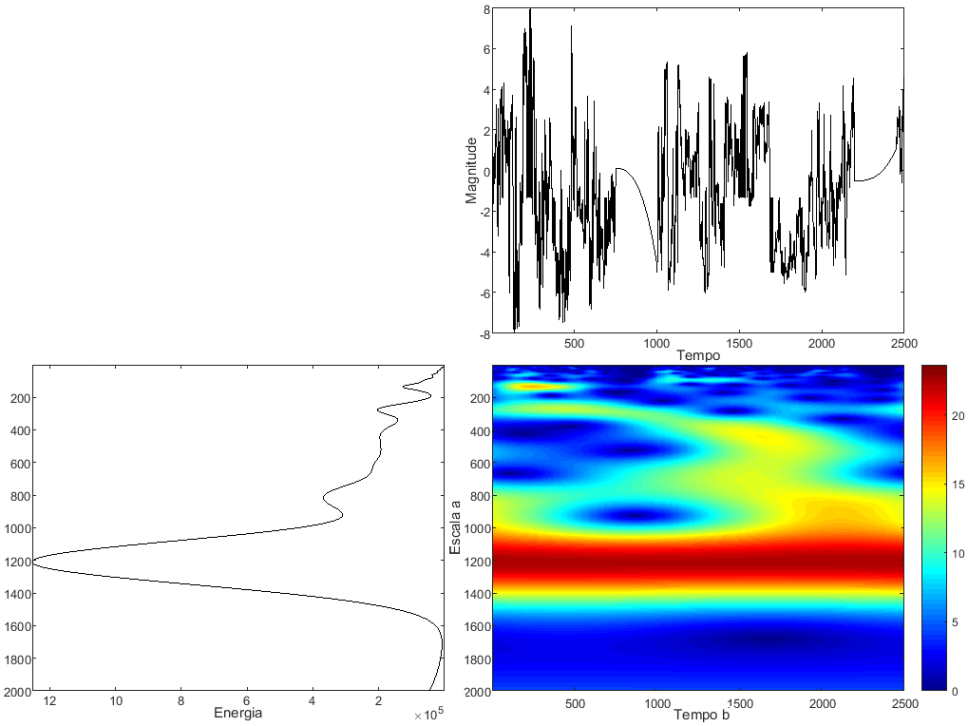


Figura 4.38 – Série interpolada com o polinômio cúbico de Hermite

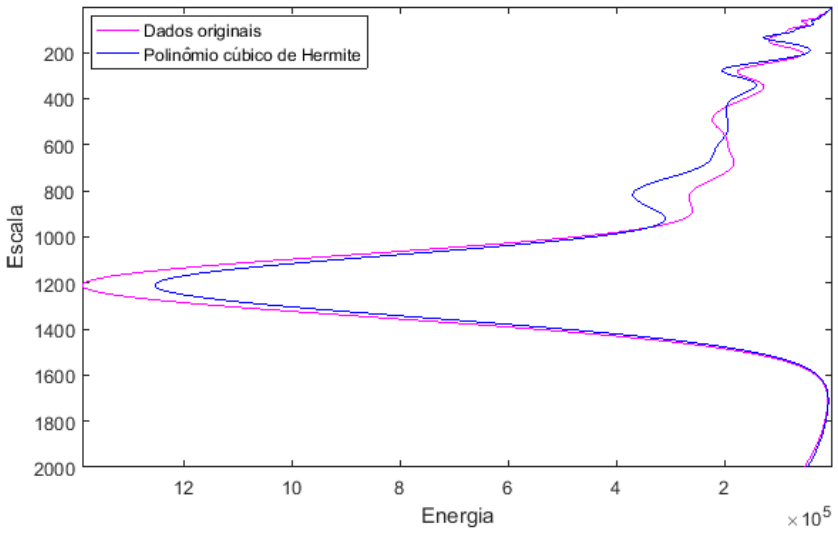


Figura 4.39 – Comparação entre o espectro global original e o interpolado com polinômio cúbico de Hermite (Tipo III)

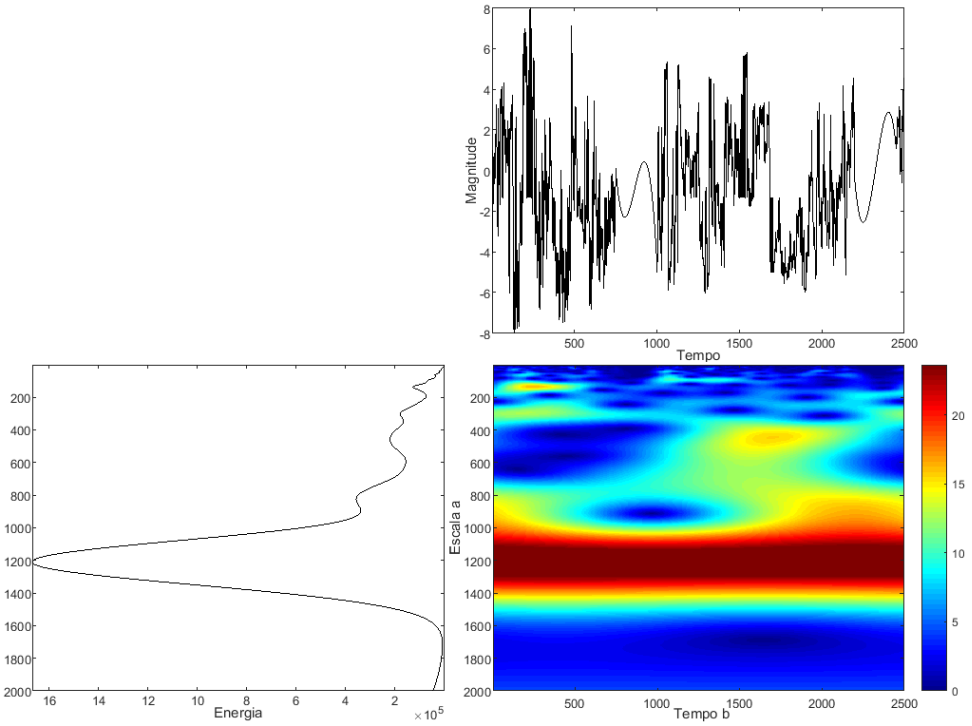


Figura 4.40 – Falhas do tipo III - Curva de Bézier

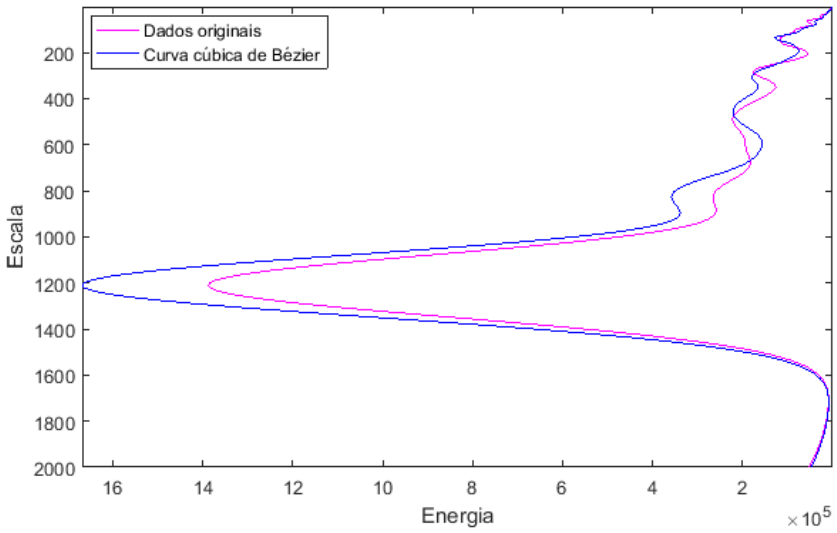


Figura 4.41 – Comparação entre o espectro global original e o interpolado com curva de Bézier (Tipo III)

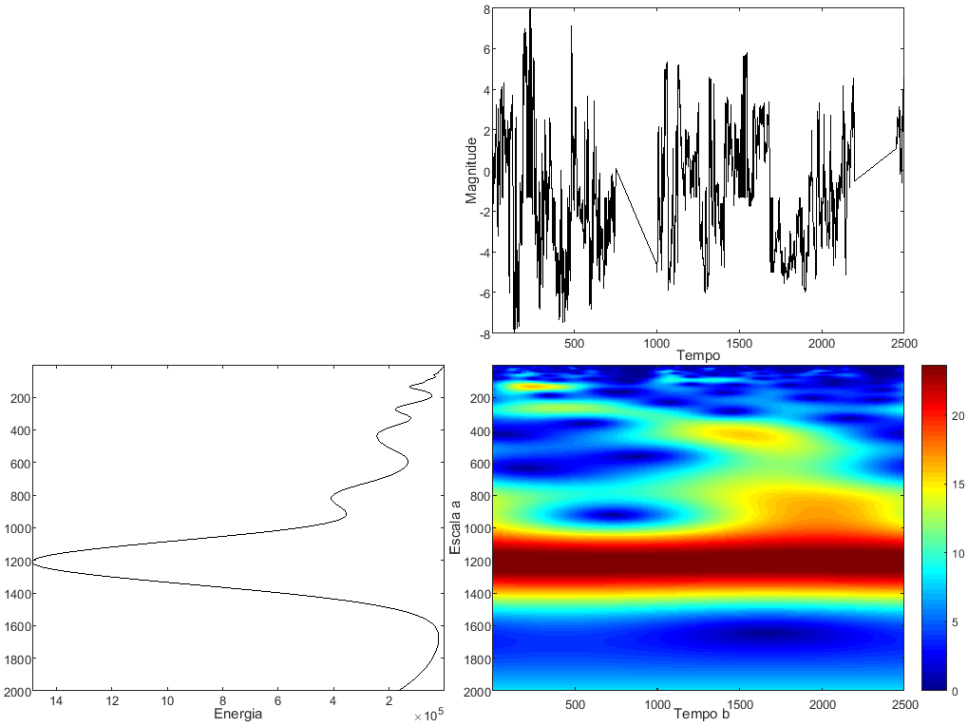


Figura 4.42 – Falhas do tipo III - Interpolação linear

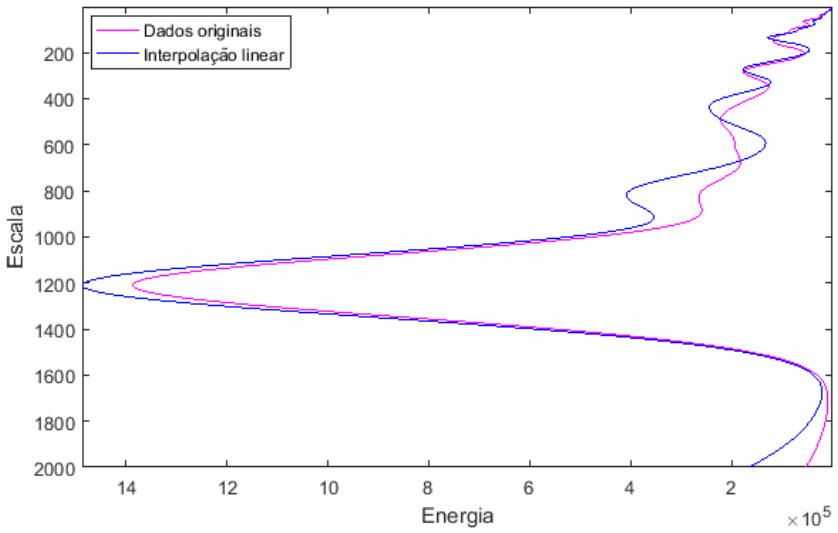


Figura 4.43 – Comparação entre o espectro global original e o interpolado com interpolação linear (Tipo III)

No caso da interpolação por *splines* cúbicos, a energia do sinal não é mantida, e as estruturas e picos de energia também são alterados, assim como no caso das falhas de tipo II. A energia global do sinal é superestimada, chegando a $3,5 \cdot 10^8$ em altas escalas, enquanto a energia máxima observada no original é $1,4 \cdot 10^6$, em escalas intermediárias. De modo geral, o resultado encontrado é muito distante do original e, assim, concluímos que o uso deste método poderia resultar em interpretações incorretas acerca do sinal sintético.

Por sua vez, o polinômio de Chebyshev não permite a conservação local de energia e o escalograma da figura 4.36 indica a presença de regiões com energia mais alta no sinal interpolado do que no sinal original. As estruturas do escalograma nas escalas entre 1000 e 1400 se mantém, embora com uma quantidade maior de energia. A mesma também é menos concentrada ao longo do tempo para certas escalas, como no intervalo entre 400 e 600. As estruturas em baixas escalas são mais suaves, mas se apresentam em locais próximos ao original. A comparação entre os espectros globais da imagem 4.37 mostra que a energia global associada à maioria das escalas é mais alta no sinal interpolado do que no sintético. Os picos de energia são mantidos em sua maioria mesmo após a interpolação e, assim como o escalograma nos mostra, não há grandes divergências na energia global das escalas abaixo de 200. Há, porém, diferenças em altas escalas, nas quais é verificada mais energia.

A interpolação pelo polinômio cúbico de Hermite em nossa terceira série com falhas permitiu a conservação da energia do sinal original, como podemos ver no escalograma da figura 4.38. No entanto, algumas escalas aparecem com estruturas alteradas em forma ou intensidade, como as próximas de 800. No espectro global (figura 4.39), vemos bastante proximidade entre a energia do sinal interpolado e a do original, embora a mesma seja subestimada em algumas escalas após a interpolação. As maiores diferenças podem ser vistas para escalas entre 600 e 1000 e próximas de 1200. A tabela 4.4 nos mostra que este método foi o que permitiu a melhor conservação da energia do sinal original.

A interpolação por curva de Bézier resulta em energia superestimada em algumas escalas, mas as estruturas apresentadas no escalograma não diferem muito das que encontramos no original, como nos mostra o escalograma da imagem 4.40. A energia em algumas escalas aparece mais espalhada no decorrer do tempo, mas mesmo para baixas escalas, onde se verificam os principais problemas, as estruturas se assemelham às originais com valores mais baixos de energia associados. A comparação entre os espectros globais apresentados na figura 4.41 indica que, mesmo com as diferenças apresentadas anteriormente, grande parte dos picos de energia se mantém e o resultado apresentado pela série interpolada se aproxima do original. As principais diferenças são encontradas nas escalas de 500 a 700.

A interpolação linear é um dos métodos que permitem melhor preservação de energia entre os utilizados, juntamente com o polinômio cúbico de Hermite, conforme a tabela 4.4. As estruturas de energia apresentadas na análise do sinal sem falhas são, em

grande parte, mantidas na série interpolada, de acordo com a figura 4.42, e isso se reflete no espectro global da imagem 4.43. A energia associada a cada escala é bastante próxima nos dois casos comparados. Devemos observar, no entanto, que este método apresenta problemas em altas escalas, nas quais é detectada mais energia na série interpolada do que a existente na original.

Por fim, comparando as análises dos resultados obtidos com todos os métodos utilizados, concluímos que, novamente, os *splines* cúbicos são o tipo de interpolação que apresenta maior divergência em relação ao resultado esperado. Os resultados obtidos com o polinômio cúbico de Chebyshev se assemelham ao apresentado pelo polinômio cúbico de Hermite, embora este possua um espectro global um pouco mais próximo do original do que o primeiro. O resultado obtido com o método de Hermite é, ainda, próximo do que temos com a curva de Bézier, cujo escalograma é, por sua vez, mais semelhante visualmente ao original. O método interpolatório cujo resultado foi mais satisfatório é a interpolação linear, apresentando resultados similares ao original tanto no escalograma quanto no espectro global.

5 Conclusões e considerações finais

Tendo em vista os testes realizados e seus resultados apresentados no capítulo anterior, vemos que, com exceção dos *splines* cúbicos, todos os tipos de interpolação utilizados podem ser considerados no tratamento de séries temporais com falhas. Embora nossos resultados não permitam afirmar, sem dúvidas, qual método interpolatório ofereceria melhores aproximações na análise tempo-escala realizada com a TWC, nossos experimentos nos permitem fazer afirmações sobre o uso destes métodos para o tratamento de falhas, que apresentaremos a seguir.

O polinômio cúbico de Hermite forneceu a melhor aproximação em nosso padrão de falhas do tipo I, embora, nos outros tipos de falha, outros métodos apresentassem resultados mais próximos do original, o que indica que o método mais adequado pode variar de acordo com o tamanho das falhas. Em todo caso, o resultado obtido com este método nas falhas de tipo I merece destaque.

A interpolação por *splines* cúbicos, por sua vez, foi o método que apresentou maiores problemas para os três tipos de falha. Trata-se de uma curva cuja característica principal é a variação lenta; no entanto, em nossos testes, os intervalos interpolados apresentaram valores muito acima ou abaixo dos valores máximos e mínimos encontrados na série original, o que pode ter causado as discrepâncias observadas.

A curva cúbica de Bézier, por permitir maior controle sobre os valores incluídos nos intervalos com falha por meio do polígono de controle, não apresentou os problemas dos *splines* cúbicos. Todavia, uma escolha distinta de pontos de controle resultaria a um polígono de controle diferente, o que afetaria o comportamento da curva obtida e, assim, geraria resultados distintos. A investigação de como escolher os pontos de controle de modo a garantir o melhor resultado por este método é, inclusive, uma proposta para um trabalho futuro.

O polinômio de Chebyshev, por sua vez, forneceu resultados satisfatórios, embora outros métodos fossem preferíveis principalmente no caso das falhas de tipo III. Outras transformações para o uso desta ferramenta ou a aplicação de polinômios ortogonais distintos poderiam oferecer resultados que permitam inferir mais informações sobre o comportamento do sinal original, com melhores representações de seus componentes frequenciais.

Por fim, o método de interpolação linear, cuja obtenção é a mais simples, proporcionou resultados notáveis, principalmente para falhas grandes. Apesar disso, não é possível afirmar que este é o melhor método nestes casos; para tanto, seriam necessários outros testes ou ainda uma análise mais detalhada.

É importante salientar que não pretendemos esgotar o tema ao qual esta pesquisa se refere, dada a infinidade de possibilidades oferecidas pelos métodos adotados e tantos outros que poderiam ser utilizados como tentativa de responder à nossa questão inicial. No entanto, dados os resultados apresentados, esperamos que esta pesquisa forneça apontamentos que permitam nortear o uso das ferramentas analisadas em situações semelhantes.

Referências

BERTKA, B. T. An introduction to bezier curves, b-splines, and tensor product surfaces with history and applications. *University of California Santa Cruz, May 30th*, 2008.

BURDEN, R. L.; FAIRES, J. D.; TASKS, A. *Análise numérica*. [S.l.]: Cengage Learning, 2008.

BURRUS, C. S.; GOPINATH, R. A.; GUO, H. Introduction to wavelets and wavelet transforms: a primer. Prentice-Hall, Inc., 1997.

DAUBECHIES, I. Where do wavelets come from? a personal point of view. *Proceedings of the IEEE*, IEEE, v. 84, n. 4, p. 510–513, 1996.

DAUBECHIES, I. et al. *Ten lectures on wavelets*. [S.l.]: SIAM, 1992. v. 61.

EVES, H. W. *Introdução à história da matemática*. [S.l.]: Unicamp, 1995.

JACQUES, L. et al. *The YAWTb Toolbox: Yet Another Wavelet Toolbox*. 2001. Disponível em: <http://sites.uclouvain.be/ispgroup/yawtb/>. Acesso em: 19 Jan 2017.

MADUREIRA, R. L.; RINCON, M. A.; SCHECHTER, L. M. Algoritmos de interseções de curvas de bézier com uma aplicação à localização de raízes de equações. 2013.

MAGRINI, L. A.; DOMINGUES, M. O.; MENDES, O. Análise tempo-escala de séries temporais de geofísica espacial com lacunas: estudo de caso. In: *Congresso Nacional de Matemática Aplicada e Computacional*. [S.l.: s.n.], 2016.

MEYER, Y. Wavelets-algorithms and applications. *Wavelets-Algorithms and applications Society for Industrial and Applied Mathematics Translation., 142 p.*, v. 1, 1993.

MORETTIN, P. A.; TOLOI, C. M. de C. *Modelos para previsão de séries temporais*. [S.l.]: Instituto de matematica pura e aplicada, 1981. v. 1.

POLIKAR, R. The story of wavelets. *Physics and modern topics in mechanical and electrical engineering*, World Scientific and Engineering Society Press: Wisconsin, USA, p. 192–197, 1999.

RESTIVO, F. Processamento digital de sinal. 2011.

RUGGIERO, M. A. G.; LOPES, V. L. d. R. *Cálculo numérico: aspectos teóricos e computacionais*. [S.l.]: Makron Books do Brasil, 1997.

SOUZA, F. de. *Transformada de Fourier*. 2009. Disponível em: <http://webx.ubi.pt/~felippe/main_pgs/mat_didp.htm>. Acesso em: 19 Jan 2017.

VALENS, C. A really friendly guide to wavelets. *ed. Clemens Valens*, 1999.

WEEKS, M. Processamento digital de sinais utilizando matlab e wavelets". *2^a edição, LTD*, 2012.

Apêndice A

Polinômio interpolador de Lagrange

Um dos tipos mais comuns de interpolação polinomial é por meio do polinômio interpolador de Lagrange. Seja f uma função conhecida nos $n+1$ pontos $x_0, x_1..., x_n$ de modo que:

$$f(x_0) = y_0, f(x_1) = y_1, ..., f(x_n) = y_n$$

Procuramos, assim, um polinômio $P(x)$ que interpole $f(x)$ nos pontos $x_0, ..., x_n$. Definimos esse polinômio da seguinte forma:

$$P_n(x_i) = y_0 \cdot L_{n,0}(x_i) + y_1 \cdot L_{n,1}(x_i) + ... + y + n \cdot L_{n,n}(x_i) = \sum_{k=0}^{n} f(x_k) \cdot L_{n,k}(x) = y_i$$

Para que obter o polinômio acima, devemos ter $L_{n,i}(x_k) = 0$, para $i \neq k$, e $L_{n,i}(x_k) = 1$, se $i = k$. Para construir cada um dos polinômios $L_{n,i}(x)$, devemos considerar que:

1. Para que tenhamos $L_{n,i}(x_k) = 0$ para $i \neq k$, devemos ter no numerador o termo $(x - x_0) \cdot (x - x_1)...(x - x_{k-1}) \cdot (x - x_{k+1})...(x - x_{n-1}) \cdot (x - x_n)$. De fato, para cada valor de $x_i \neq x_k$, temos um fator neste termo que é igual a 0, anulando o numerador;

2. Para que $L_{n,i}(x_i)$ seja igual a 1 para todo i, devemos ter o denominador igual ao numerador dado em i) calculado em cada um desses valores. O numerador indicado em i), calculado em x_i, é igual a $(x_i - x_0) \cdot (x_i - x_1)...(x_i - x_{i-1}) \cdot (x_i - x_{i+1})...(x_i - x_{n-1})(x_i - x_n)$.

Deste modo, usando as informações de i) e ii), temos que:

$$L_{n,i}(x) = \frac{(x - x_0) \cdot (x - x_1)...(x - x_{k-1}) \cdot (x - x_{k+1})...(x - x_{n-1}) \cdot (x - x_n)}{(x_i - x_0) \cdot (x_i - x_1)...(x_i - x_{k-1}) \cdot (x_i - x_{k+1})...(x_i - x_{n-1}) \cdot (x_i - x_n)}$$

Assim, o polinômio $P(x)$ está definido e respeita as condições dadas.

O método das diferenças divididas de Newton

O operador diferenças divididas

Para construir o polinômio inteprolador de Newton, precisamos definir o operador diferenças divididas como segue:

$$f[x_0] = f(x_0)$$

$$f[x_0, x_1] = \frac{f[x_1] - f[x_0]}{x_1 - x_0} = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

$$f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} = \frac{f[x_0, x_2] - f[x_0, x_1]}{x_2 - x_1}$$

...

$$f[x_0, x_1, x_2, ..., x_n] = \frac{f[x_1, x_2, ..., x_n] - f[x_0, x_1, ..., x_{n-1}]}{x_n - x_0}$$

O polinômio interpolador de Newton

O polinômio interpolador de Newton assume a seguinte forma:

$$P_n(x) = a_0 + a_1 \cdot (x - x_0) + a_2 \cdot (x - x_0) \cdot (x - x_1) + ... + a_n \cdot (x - x_0) \cdot (x - x_1)...(x - x_{n-1})$$

Observemos que $P_n(x_0) = a_0$ e, pela forma como é definido o polinômio $P_n(x)$, temos que:

$$a_0 = f(x_0) \tag{5.1}$$

Por definição, temos que $P_n(x_1) = a_0 + a_1 \cdot (x_1 - x_0)$. De modo análogo e utilizando (5.1), temos que:

$$P_n(x_1) = f(x_0) + a_1(x_1 - x_0) = f(x_1)$$

$$f(x_0) + a_1(x_1 - x_0) = f(x_1)$$

$$a_1 \cdot (x_1 - x_0) = f(x_1) - f(x_0)$$

$$a_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = f[x_0, x_1] \tag{5.2}$$

Calculando, ainda, a_2, temos:

$$P_n(x_2) = a_0 + a_1 \cdot (x_2 - x_0) + a_2 \cdot (x_2 - x_0) \cdot (x_2 - x_1) = f(x_2)$$

$$a_2 \cdot (x_2 - x_0) \cdot (x_2 - x_1) = f(x_2) - \frac{f(x_1) - f(x_0)}{x_1 - x_0} \cdot (x_2 - x_0) - f(x_0)$$

$$a_2 = \frac{f(x_2) - f(x_0)}{(x_2 - x_0) \cdot (x_2 - x_1)} - \frac{f(x_1) - f(x_0)}{(x_1 - x_0) \cdot (x_2 - x_1)} = \frac{\frac{f(x_2)-f(x_0)}{x_2-x_0} - \frac{f(x_1)-f(x_0)}{x_1-x_0}}{x_2 - x_1}$$

$$a_2 = \frac{f[x_0, x_2] - f[x_0, x_1]}{x_2 - x_1} = f[x_0, x_1, x_2]$$

Dessa forma, espera-se que, para $0 \leq k \leq n$, tenhamos:

$$a_k = f[x_0, x_1, ..., x_k]$$

E, assim, o polinômio $P_n(x)$ pode ser reescrito como:

$$P_n(x) = f(x_0) + \sum_{i=1}^{n} f[x_0, x_1, ..., x_i] \cdot (x - x_0) \cdot (x - x_1)...(x - x_{i-1})$$

Apêndice B

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                                 %
% CODIGO PARA PLOTAR O PRIMEIRO EXEMPLO DA TRANSFORMADA DE FOURIER                %
%                                                                                 %
% Escrito por: Luciano Aparecido Magrini, Paula Neves de Araujo                   %
% Data: 19/11/2016                                                                %
%                                                                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all % Remove objetos definidos previamente
clc % Limpa a tela de comandos do MATLAB

t = linspace(0,4*pi,1024); % Define um vetor de tempo
f = cos(t); % Define a funcao f(t)

fft_f = fft(f); % Aplica a Transformada Rapida de Fourier (FFT)
fft_f = fftshift(fft_f); % Desloca a FFT

t_aux = linspace(-pi, pi, 1024); % Cria um vetor auxiliar de tempo

aux1 = length(t_aux); % Cria uma variavel que armazena o tamanho do vetor t_aux
aux2 = length(fft_f); % Cria uma variavel que armazena o tamanho do vetor fft_f

figure(01) % Cria uma nova figura
subplot(2, 1, 2) % Comando para plotagem da funcaoo e da FFT na mesma janela
plot(t_aux,abs(fft_f), 'k') % Comando para plotagem do valor absoluto da FFT
axis tight; % Ajusta o tamanho dos eixos
xlabel ('frequencia'); % Define o nome do eixo das abscissas
ylabel ('amplitude'); % Define o nome do eixo das ordenadas
subplot(2,1,1)
plot(t, f, 'k'); % Comando para plotagem da funcao f
axis tight; % Ajusta o tamanho dos eixos
xlabel ('t'); % Define o nome do eixo das abscissas
ylabel ('f(t)'); % Define o nome do eixo das ordenadas


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                                 %
% CODIGO PARA PLOTAR O SEGUNDO EXEMPLO DA TRANSFORMADA DE FOURIER                 %
%                                                                                 %
% Escrito por: Luciano Aparecido Magrini, Paula Neves de Araujo                   %
% Data: 19/11/2016                                                                %
```

```
%                                                                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all % Remove objetos definidos previamente
clc % Limpa a tela de comandos do MATLAB

t=linspace(0,3*pi,1024); % Define um vetor de tempo
f=cos(6*pi*t).*exp(-t); % Define a funcao f(t)

fft_f = fft(f); % Aplica a Transformada Rapida de Fourier (FFT)
fft_f = fftshift(fft_f); % Desloca a FFT

t_aux = linspace(-pi, pi, 1024); % Cria um vetor auxiliar de tempo

aux1 = length(t_aux); % Cria uma variavel que armazena o tamanho do vetor t_aux
aux2 = length(fft_f); % Cria uma variavel que armazena o tamanho do vetor fft_f

figure(01) % Cria uma nova figura
subplot(2,1,1)
plot(t, f, 'k'); % Comando para plotagem da funcao f
axis tight; % Ajusta o tamanho dos eixos
xlabel ('t'); % Define o nome do eixo das abscissas
ylabel ('f(t)'); % Define o nome do eixo das ordenadas
subplot (2, 1, 2)
plot(t_aux(aux1/2+1:aux1),abs(fft_f(aux2/2+1:aux2)), 'k');
% Plotagem das frequencias positivas
xlabel('frequencia'); % Define o nome do eixo das abscissas
ylabel('amplitude'); % Define o nome do eixo das ordenadas
axis tight; % Ajusta o tamanho dos eixos


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                                 %
% CODIGO PARA PLOTAR O TERCEIRO EXEMPLO DA TRANSFORMADA DE FOURIER                %
%                                                                                 %
% Escrito por: Luciano Aparecido Magrini, Paula Neves de Araujo                   %
% Data: 19/11/2016                                                                %
%                                                                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all % Remove objetos definidos previamente
clc % Limpa a tela de comandos do MATLAB

tempo=linspace(0,10,512); % Define um vetor de tempo
f=exp(tempo); % Define a funcao f

fft_f = fft(f); % Aplica a Transformada Rapida de Fourier (FFT)
fft_f = fftshift(fft_f); % Desloca a FFT
```

```
t_aux = linspace(-pi, pi, 512); % Cria um vetor auxiliar de tempo

aux1 = length(t_aux); % Cria uma variavel que armazena o tamanho do vetor t_aux
aux2 = length(fft_f); % Cria uma variavel que armazena o tamanho do vetor fft_f

figure(01) % Cria uma nova figura
subplot(2,1,1)
plot(tempo, f, 'k'); % Comando para plotagem da funcao f
axis tight; % Ajusta o tamanho dos eixos
xlabel ('t'); % Define o nome do eixo das abscissas
ylabel ('f(t)'); % Define o nome do eixo das ordenadas
subplot (2, 1, 2)
plot(t_aux(aux1/2+1:aux1),abs(fft_f(aux2/2+1:aux2)), 'k');
% Plotagem das frequencias positivas
xlabel('frequencia'); % Define o nome do eixo das abscissas
ylabel('amplitude'); % Define o nome do eixo das ordenadas
axis tight; % Ajusta o tamanho dos eixos


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
% CODIGO PARA PLOTAR O QUARTO EXEMPLO DA TRANSFORMADA DE FOURIER               %
%                                                                              %
% Escrito por: Luciano Aparecido Magrini, Paula Neves de Araujo                %
% Data: 19/11/2016                                                             %
%                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all % Remove objetos definidos previamente
clc % Limpa a tela de comandos do MATLAB
tempo=linspace(0,10,512); % Cria um vetor de tempo
pulso=zeros(1,512); % Cria um vetor de 512 entradas iguais a 0
pulso(100)=1; % Atribui o valor 1 para t=100
pulso(380)=5; % Atribui o valor 5 para t=380
fft_f = fft(pulso); % Aplica a Transformada Rapida de Fourier (FFT)
fft_f = fftshift(fft_f); % Desloca a FFT

t_aux = linspace(-pi, pi, 512); % Cria um vetor auxiliar de tempo

aux1 = length(t_aux); % Cria uma variavel que armazena o tamanho do vetor t_aux
aux2 = length(fft_f);  % Cria uma variavel que armazena o tamanho do vetor fft_f

figure(01) % Cria uma nova figura
subplot(2,1,1)
plot(tempo, pulso, 'k'); % Comando para plotagem da funcao pulso
axis tight; % Ajusta o tamanho dos eixos
xlabel ('t');  % Define o nome do eixo das abscissas
```

```
ylabel ('f(t)'); % Define o nome do eixo das ordenadas
subplot (2, 1, 2)
plot(t_aux(aux1/2+1:aux1),abs(fft_f(aux2/2+1:aux2)), 'k');
% Plotagem das frequencias positivas
xlabel('frequencia'); % Define o nome do eixo das abscissas
ylabel('amplitude'); % Define o nome do eixo das ordenadas
axis tight; % Ajusta o tamanho dos eixos



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
% CODIGO PARA PLOTAR AS WAVELETS CHAPEU MEXICANO E DE MORLET                   %
%                                                                              %
% Escrito por: Luciano Aparecido Magrini, Paula Neves de Araujo                %
% Data: 19/11/2016                                                             %
%                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



clear all % Remove objetos definidos previamente
clc % Limpa a tela de comandos do MATLAB
t=linspace(-20,20,512); % Cria um vetor de tempo

f=2/(3^0.5*pi^0.25)*exp(-t.^2/2).*(1-t.^2); % Define a funcao Chapeu Mexicano

g=exp(i*pi/2*t)*1/((pi*5^2)^(1/4)).*exp(-t.^2/(2*5^2));
% Define a funcao wavelet de Morlet

figure(01) % Cria uma nova figura
subplot(1,2,1) % Comando para plotagem das duas funcoes na mesma janela
plot(t,f) % Comando para plotagem da funcao Chapeu Mexicano
axis tight % Ajusta o tamanho dos eixos
xlabel('tempo (t)') % Define o nome do eixo das abscissas
ylim([-1 1] % Define um valor minimo e um maximo para o eixo das ordenadas
ylabel('amplitude') % Define o nome do eixo das ordenadas
subplot(1,2,2)
plot(t,real(g),'b',t,imag(g),'r')
% Comando para plotagem das partes real e imaginaria da wavelet de Morlet
legend('Re','Im'); % Define legendas para cada uma destas partes
axis tight % Ajusta o tamanho dos eixos
xlabel('tempo (t)') % Define o nome do eixo das abscissas
ylabel('amplitude') % Define o nome do eixo das ordenadas



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
% CODIGO PARA PLOTAR O PRIMEIRO EXEMPLO DA TRANSFORMADA WAVELET CONTINUA       %
```

```
%                                                                                %
% Escrito por: Luciano Aparecido Magrini, Paula Neves de Araujo                  %
% Data: 19/11/2016                                                               %
%                                                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all % Remove objetos definidos previamente
clc % Limpa a tela de comandos do MATLAB
tempo=linspace(0.25,10,512); % Define o vetor de tempo
pulso=zeros(1,512); % Cria um vetor de 512 entradas iguais a 0
pulso(100)=3; % Atribui o valor 3 a t=100
pulso(380)=5; % Atribui o valor 5 a t=380

figure(01) % Cria uma nova figura
plot(pulso) % Comando para plotagem do vetor pulso
axis tight % Ajusta o tamanho dos eixos

s=linspace(1, 50, 512); % Cria um vetor de 512 escalas
wav=cwt1d(fft(pulso), 'mexican', s, ['export']); % Aplica a TWC na FFT do sinal pulso

figure(02) % Cria uma nova figura
yashow(wav); % Comando para gerar o escalograma

escalograma=abs(wav.*wav); % Cria a matriz dos modulos dos quadrados dos coeficientes

for i=1:length(s) % Comando usado para somar as linhas da matriz anterior
    espectro_global(i)=sum(escalograma(i,:)); % Armazena as somas em um vetor
end

figure(03) % Cria uma nova figura
plot (espectro_global, 1:1:512) % Comando para plotagem do espectro global
set(gca,'xdir', 'reverse'); % Comando para inversao do eixo das abscissas
axis tight % Ajusta o tamanho dos eixos


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                                %
% CODIGO PARA PLOTAR O SEGUNDO EXEMPLO DA TRANSFORMADA WAVELET CONTINUA          %
%                                                                                %
% Escrito por: Luciano Aparecido Magrini, Paula Neves de Araujo                  %
% Data: 19/11/2016                                                               %
%                                                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all % Remove objetos definidos previamente
clc % Limpa a tela de comandos do MATLAB
tempo=linspace(0,10,1024); % Define o vetor de tempo
for i=1:length(tempo) % Comando utilizado para definir a funcao g
```

```
if tempo(i)<=4 % Comando para mudar o comportamento da funcao em t>4
   g(i)=2*i; % Associa, a cada valor t menor ou igual a 4, o dobro de seu indice
   else
   g(i)=150; % Associa aos demais valores de t a imagem 150
end
end

figure(01) % Cria uma nova figura
plot(g) % Comando para plotagem da funcao g
axis tight % Ajusta o tamanho dos eixos

s=linspace(1, 25, 1024); % Cria um vetor de 1024 escalas
wav=cwt1d(fft(g), 'mexican', s, ['export']); % Aplica a TWC na FFT da funcao g

figure(02) % Cria uma nova figura
yashow(wav); % Comando para gerar o escalograma

escalograma=abs(wav.*wav); % Cria a matriz dos modulos dos quadrados dos coeficientes

for i=1:length(s) % Comando usado para somar as linhas da matriz anterior
    espectro_global(i)=sum(escalograma(i,:)); % Armazena as somas em um vetor
end

figure(03) % Cria uma nova figura
plot (espectro_global, 1:1:1024)% Comando para plotagem do espectro global
set(gca,'xdir', 'reverse'); % Comando para inversao do eixo das abscissas
axis tight % Ajusta o tamanho dos eixos


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
% CODIGO PARA PLOTAR O TERCEIRO EXEMPLO DA TRANSFORMADA WAVELET CONTINUA       %
%                                                                              %
% Escrito por: Luciano Aparecido Magrini, Paula Neves de Araujo                %
% Data: 19/11/2016                                                             %
%                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clear all % Remove objetos definidos previamente
clc % Limpa a tela de comandos do MATLAB
tempo=linspace(0,4.45*pi,512); % Define o vetor de tempo
h=sin(2*pi*tempo) + sin(7*pi*tempo); % Define a funcao h(t)

figure(01) % Cria uma nova figura
plot(tempo,h) % Comando para plotagem da funcao h
axis tight % Ajusta o tamanho dos eixos
```

```
s=linspace(1, 100, 256); % Cria um vetor de 256 escalas
wav=cwt1d(fft(h), 'morlet', s, ['export']); % Aplica a TWC na FFT da funcao h

figure(02)  % Cria uma nova figura
yashow(wav); % Comando para gerar o escalograma

escalograma=abs(wav.*wav); % Cria a matriz dos modulos dos quadrados dos coeficientes

for i=1:length(s) % Comando usado para somar as linhas da matriz anterior
    espectro_global(i)=sum(escalograma(i,:)); % Armazena as somas em um vetor
end

figure(03) % Cria uma nova figura
plot (espectro_global, 1:1:256) % Comando para plotagem do espectro global
set(gca,'xdir', 'reverse'); % Comando para inversao do eixo das abscissas
axis tight % Ajusta o tamanho dos eixos


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
% CODIGO PARA PLOTAR O QUARTO EXEMPLO DA TRANSFORMADA WAVELET CONTINUA         %
%                                                                              %
% Escrito por: Luciano Aparecido Magrini, Paula Neves de Araujo                %
% Data: 19/11/2016                                                             %
%                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clear all % Remove objetos definidos previamente
clc % Limpa a tela de comandos do MATLAB
tempo=linspace(0,10.55*pi,512); % Cria um vetor de tempo
for j=1:256 % Comando para gerar as 256 primeiras entradas da funcao x
x(j)=cos(2*pi*tempo(j)); % Define valores para estas entradas
end
for j=256:512 % Comando para gerar as demais entradas
x(j)=sin(5*pi*tempo(j)); % Define valores para estas entradas
end

figure(01) % Cria uma nova figura
plot(tempo, x); % Comando para plotagem da funcao x
axis tight % Ajusta o tamanho dos eixos

s=linspace(0, 25, 512); % Cria um vetor de 512 escalas
wav=cwt1d(fft(x), 'morlet', s, ['export']); % Aplica a TWC na FFT da funcao x

figure(02) % Cria uma nova figura
yashow(wav); % Comando para gerar o escalograma
```

```
escalograma=abs(wav.*wav); % Cria a matriz dos modulos dos quadrados dos coeficientes

for i=1:length(s) % Comando usado para somar as linhas da matriz anterior
    espectro_global(i)=sum(escalograma(i,:)); % Armazena as somas em um vetor
end

figure(03) % Cria uma nova figura
plot (espectro_global, 1:1:512) % Comando para plotagem do espectro global
set(gca,'xdir', 'reverse'); % Comando para inversao do eixo das abscissas
axis tight % Ajusta o tamanho dos eixos


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
% CODIGO PARA GERAR OS TESTES COM FALHAS DO TIPO I                             %
%                                                                              %
% Escrito por: Luciano Aparecido Magrini, Paula Neves de Araujo                %
% Data: 19/11/2016                                                             %
%                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all % Remove objetos definidos previamente
clc % Limpa a tela de comandos do MATLAB

A=load('dados.txt'); % Leitura do arquivo de dados.
A_copia=A; % Copia para armazenar os dados originais
A_aux=A; % Copia do arquivo de dados.

% Os seguintes comandos inserem falhas no vetor A_aux
A_aux(201:225)=nan;
A_aux(526:550)=nan;
A_aux(701:725)=nan;
A_aux(1101:1125)=nan;
A_aux(1326:1350)=nan;
A_aux(1886:1910)=nan;
A_aux(2001:2025)=nan;
A_aux(2101:2125)=nan;
A_aux(2301:2325)=nan;
A_aux(2416:2440)=nan;

nan_aux=isnan(A_aux); % A funcao ''isnan'' identifica as posicoes de falha
% e gera um novo vetor com zeros e valor 1 nas falhas

h_max=max(A_aux); % Encontrando o valor maximo da serie
h_min=min(A_aux); % Encontrando o valor minimo da serie

% Encontrando os indices que correspondem a intervalos com falha
% e os indices que correspondem a dados
```

```
indices_com_falha=find(nan_aux==1);
indices_com_falha=indices_com_falha';
indices_com_dados=find(nan_aux==0);
indices_com_dados= indices_com_dados';

% Criacao de um vetor auxiliar cujas entradas
% sao as posicoes conhecidas do arquivo de texto lido
% Sera usado no processo de interpolacao por splines cubicos
A_aux1=A_aux;
cte_aux=length(indices_com_dados);
f=zeros(1,cte_aux);
for d=1:cte_aux
    f(d)=A_aux1(indices_com_dados(d));
end

% Criacao de um vetor auxiliar cujas entradas
% sao as posicoes conhecidas do arquivo de texto lido
% Sera usado no processo de interpolacao por polinomio cubico de Hermite
A_aux2=A_aux;
cte_aux=length(indices_com_dados);
g=zeros(1,cte_aux);
for d=1:cte_aux
    g(d)=A_aux1(indices_com_dados(d));
end

% Usando spline cubica
aprox_spline=spline(indices_com_dados,f,indices_com_falha);
for m=1:length(aprox_spline)
    A_aux1(indices_com_falha(m))=aprox_spline(m);
end

% Usando polinomio cubico de Hermite
aprox_hermite=pchip(indices_com_dados,f,indices_com_falha);
for m=1:length(aprox_hermite)
    A_aux2(indices_com_falha(m))=aprox_hermite(m);
end

% Criacao de um vetor auxiliar,
% que sera usado no processo de interpolacao pela curva de Bezier
A_aux3=A_aux;

n=1; % Inicializa o contador que sera utilizado para gerar a curva
y(1)=1; % Define o valor de y(1), que tambem sera utilizado para gerar a curva
teste=isnan(A_aux3); % Identifica as posicoes com falha na serie

for i=2:11 % Aqui, geraremos um vetor x com 10 entradas,
% que marcam o inicio de cada intervalo com falha
    n=y(i-1);
```

```
    while teste(n)==0
        x(i)=n;
        n=n+1;
    end
    m=n;
    if m<2500
    while teste(m)==1 % Geramos tambem um vetor y com 10 entradas,
    % que marcam o fim dos intervalos com falha
        m=m+1;
        y(i)=m;
    end
        else
            y(i)=y(i-1);
    end

p0 = [x(i) A_aux3(x(i))]';  % p0 e a primeira extremidade
p1 = [(y(i)-x(i))/2 h_min]'; % p1 e o primeiro ponto de controle
p2 = [(y(i)-x(i))/2 h_max]'; % p2 e o segundo ponto de controle
p3 = [y(i) A_aux3(y(i))]'; % p3 e a segunda extremidade

B = [p0 p1 p2 p3]'; % geramos uma matriz cujas colunas
% correspondem aos vetores encontrados anteriormente

M = [-1 3 -3 1; 3 -6 3 0; -3 3 0 0; 1 0 0 0];
% M e a matriz correspondente
% a curva cubica de Bezier

A = M*B; % Multiplicamos a matriz M da curva pela matriz B dos pontos

u = linspace(0,1,y(i)-x(i)+1); % Geramos um vetor u, com entradas de 0 a 1,
% na mesma quantidade de valores com falha no intervalo observado
u = u';
unit = ones(size(u));
U = [u.^3 u.^2 u unit];

Pu = U*A; % Produto da matriz A pela matriz U

int=Pu(:, 2)'; % Os valores de interesse nesse caso
% sao os valores da segunda coluna da matriz Pu

A_aux3(x(i):y(i))=int; % Substituimos o intervalo com falha
% pelos valores do vetor int
end


% Criacao de um vetor auxiliar,
% que sera usado no processo de interpolacao com o polinomio cubico de Chebyshev
```

```
A_aux4=A_aux;

teste=isnan(A_aux4); % Encontra os valores de A_aux4 com falha

for i=2:11
    n=y(i-1);
    while teste(n)==0
        x(i)=n;
        n=n+1;
    end
    m=n;
    if m<2500
    while teste(m)==1
        m=m+1;
        y(i)=m;
    end
        else
            y(i)=y(i-1);
    end
    v=linspace(-1, 1, y(i)-x(i)+1); % Geramos um vetor com o numero de componentes
    % igual ao numero de entradas com falhas no intervalo observado
    chebyshev=4*(v.^3)-3*v; % Aplicamos o polinomio cubico de Chebyshev no vetor v,
    % obtendo uma curva com oscilacao controlada;
    chebyshev=1/2*((A_aux4(y(i))-A_aux4(x(i)))*chebyshev+A_aux4(x(i))+A_aux(y(i)));
    A_aux4(x(i):y(i))=chebyshev; % Substituimos o intervalo com falha
    % pelo vetor obtido
end

% Criacao de um vetor auxiliar,
% que sera usado no processo de interpolacao linear
A_aux5=A_aux;


for i=2:11
    n=y(i-1);
    while teste(n)==0
        x(i)=n;
        n=n+1;
    end
    m=n;
    if m<2500
    while teste(m)==1
        m=m+1;
        y(i)=m;
    end
        else
            y(i)=y(i-1);
    end
```

```
    m=(A_aux5(y(i))-A_aux5(x(i)))/(y(i)-x(i));
    % Encontra o coeficiente angular da reta
    n=A_aux5(y(i))-m*y(i);  % Encontra o coeficiente linear da reta

    for j=x(i):y(i)
    A_aux5(j)=m*j+n; % Substitui o intervalo com falha pela reta correspondente
    end
end

% Plotagem das series obtidas

figure(01)
plot(A_copia,'k')
title('Serie original')
axis tight
xlabel('Tempo');
ylabel('Magnitude');

figure(02)
plot(A_aux1,'k')
title('Interpolacao por splines cubicos')
axis tight
xlabel('Tempo');
ylabel('Magnitude');

figure(03)
plot(A_aux2,'k')
title('Interpolacao por polinomio cubico de Hermite')
axis tight
xlabel('Tempo');
ylabel('Magnitude');

figure(04)
plot(A_aux3,'k')
title('Interpolacao por curva cubica de Bezier')
axis tight
xlabel('Tempo');
ylabel('Magnitude');

figure(05)
plot(A_aux4,'k')
title('Interpolacao por polinomio cubico de Chebyshev')
axis tight
xlabel('Tempo');
ylabel('Magnitude');

figure(06)
plot(A_aux5,'k')
```

```
title('Interpolacao linear')
axis tight
xlabel('Tempo');
ylabel('Magnitude');

% Aplicando a TWC na serie original e nas series interpoladas
coefs01=cwt1d(fft(A_copia), 'morlet', 1:1:2000, [,'export']);
coefs02=cwt1d(fft(A_aux1), 'morlet', 1:1:2000, [,'export']);
coefs03=cwt1d(fft(A_aux2), 'morlet', 1:1:2000, [,'export']);
coefs04=cwt1d(fft(A_aux3), 'morlet', 1:1:2000, [,'export']);
coefs05=cwt1d(fft(A_aux4), 'morlet', 1:1:2000, [,'export']);
coefs06=cwt1d(fft(A_aux5), 'morlet', 1:1:2000, [,'export']);

% Plotando os escalogramas de cada serie
figure(07)
yashow(coefs01);
xlabel('Tempo b')
ylabel('Escala a')
title('Escalograma - Serie original')

figure(08)
yashow(coefs02);
xlabel('Tempo b')
ylabel('Escala a')
title('Escalograma - Interpolacao por spline cubico')

figure(09)
yashow(coefs03);
xlabel('Tempo b')
ylabel('Escala a')
title('Escalograma - Interpolacao por polinomio cubico de Hermite')

figure(10)
yashow(coefs04);
xlabel('Tempo b')
ylabel('Escala a')
title('Escalograma - Interpolacao por curva cubica de Bezier')

figure(11)
yashow(coefs05);
xlabel('Tempo b')
ylabel('Escala a')
title('Escalograma - Interpolacao por polinomio cubico de Chebyshev')

figure(12)
yashow(coefs06);
xlabel('Tempo b')
ylabel('Escala a')
```

```
title('Escalograma - Interpolacao linear')

% Obtendo os valores para o espectro global
escalograma01=abs(coefs01.*coefs01);
for i=1:2000
    espectro_global01(i)=sum(escalograma01(i,:));
end

escalograma02=abs(coefs02.*coefs02);
for i=1:2000
    espectro_global02(i)=sum(escalograma02(i,:));
end

escalograma03=abs(coefs03.*coefs03);
for i=1:2000
    espectro_global03(i)=sum(escalograma03(i,:));
end

escalograma04=abs(coefs04.*coefs04);
for i=1:2000
    espectro_global04(i)=sum(escalograma04(i,:));
end

escalograma05=abs(coefs05.*coefs05);
for i=1:2000
    espectro_global05(i)=sum(escalograma05(i,:));
end

escalograma06=abs(coefs06.*coefs06);
for i=1:2000
    espectro_global06(i)=sum(escalograma06(i,:));
end

% Plotagem dos espectros globais
figure(13)
plot(espectro_global01, 1:1:2000, 'k')
set(gca, 'xdir','reverse')
set(gca, 'ydir','reverse')
axis tight
xlabel('Energia')
title('Espectro global - Serie original')

figure(14)
plot(espectro_global02, 1:1:2000, 'k')
set(gca, 'xdir','reverse')
set(gca, 'ydir','reverse')
axis tight
xlabel('Energia')
```

```
title('Espectro global - Interpolacao por splines cubicos')

figure(15)
plot(espectro_global03, 1:1:2000, 'k')
set(gca, 'xdir','reverse')
set(gca, 'ydir','reverse')
axis tight
xlabel('Energia')
title('Espectro global - Interpolacao por polinomio cubico de Hermite')

figure(16)
plot(espectro_global04, 1:1:2000, 'k')
set(gca, 'xdir','reverse')
set(gca, 'ydir','reverse')
axis tight
xlabel('Energia')
title('Espectro global - Interpolacao por curva cubica de Bezier')

figure(17)
plot(espectro_global05, 1:1:2000, 'k')
set(gca, 'xdir','reverse')
set(gca, 'ydir','reverse')
axis tight
xlabel('Energia')
title('Espectro global - Interpolacao por polinomio cubico de Chebyshev')

figure(18)
plot(espectro_global06, 1:1:2000, 'k')
set(gca, 'xdir','reverse')
set(gca, 'ydir','reverse')
axis tight
xlabel('Energia')
title('Espectro global - Interpolacao linear')

% Obtencao da energia total das series
energia_serieoriginal=sum(sum(escalograma01))
energia_spline=sum(sum(escalograma02))
energia_hermite=sum(sum(escalograma03))
energia_bezier=sum(sum(escalograma04))
energia_chebyshev=sum(sum(escalograma05))
energia_linear=sum(sum(escalograma06))

% Obtencao da porcentagem da energia de cada serie interpolada em relacao a original
porcentagem_spline=(energia_spline)/(energia_serieoriginal)
porcentagem_hermite=(energia_hermite)/(energia_serieoriginal)
porcentagem_bezier=(energia_bezier)/(energia_serieoriginal)
porcentagem_chebyshev=(energia_chebyshev)/(energia_serieoriginal)
porcentagem_linear=(energia_linear)/(energia_serieoriginal)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                                %
% CODIGO PARA GERAR OS TESTES COM FALHAS DO TIPO II                              %
%                                                                                %
% Escrito por: Luciano Aparecido Magrini, Paula Neves de Araujo                  %
% Data: 19/11/2016                                                               %
%                                                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clear all
clc

A=load('dados.txt'); % Leitura do arquivo de dados
A_copia=A; % Copia para armazenamento dos dados originais
A_aux=A; % Copia do arquivo de dados.


% Os seguintes comandos inserem falhas no vetor A_aux
A_aux(201:325)=nan;
A_aux(526:650)=nan;
A_aux(701:825)=nan;
A_aux(1101:1225)=nan;
A_aux(1501:1625)=nan;
A_aux(1886:2010)=nan;
A_aux(2101:2225)=nan;
A_aux(2316:2440)=nan;

nan_aux=isnan(A_aux); % A funcao ''isnan'' identifica as posicoes de falha
%e gera um novo vetor com zeros e valor 1 nas falhas

h_max=max(A_aux); % Encontrando o valor maximo da serie
h_min=min(A_aux); % Encontrando o valor minimo da serie

% Encontrando os indices que correspondem a intervalos com falha
% e os indices que correspondem a dados
indices_com_falha=find(nan_aux==1);
indices_com_falha=indices_com_falha';
indices_com_dados=find(nan_aux==0);
indices_com_dados= indices_com_dados';

% Criacao de um vetor auxiliar cujas entradas
% sao as posicoes conhecidas do arquivo de texto lido
% Sera usado no processo de interpolacao por splines cubicos
A_aux1=A_aux;
cte_aux=length(indices_com_dados);
f=zeros(1,cte_aux);
for d=1:cte_aux
```

```
    f(d)=A_aux1(indices_com_dados(d));
end

% Criacao de um vetor auxiliar cujas entradas
% sao as posicoes conhecidas do arquivo de texto lido
% Sera usado no processo de interpolacao por polinomio cubico de Hermite
A_aux2=A_aux;
cte_aux=length(indices_com_dados);
g=zeros(1,cte_aux);
for d=1:cte_aux
    g(d)=A_aux1(indices_com_dados(d));
end

% Usando spline cubica
aprox_spline=spline(indices_com_dados,f,indices_com_falha);
for m=1:length(aprox_spline)
    A_aux1(indices_com_falha(m))=aprox_spline(m);
end

% Usando polinomio cubico de Hermite
aprox_hermite=pchip(indices_com_dados,f,indices_com_falha);
for m=1:length(aprox_hermite)
    A_aux2(indices_com_falha(m))=aprox_hermite(m);
end

% Criacao de um vetor auxiliar cujas entradas
% sao as posicoes conhecidas do arquivo de texto lido
% Sera usado no processo de interpolacao pela curva de Bezier
A_aux3=A_aux;

n=1; % Inicializa o contador que sera utilizado para gerar a curva
y(1)=1; % Define o valor de y(1), que tambem sera utilizado para gerar a curva
teste=isnan(A_aux3); % Identifica as posicoes com falha na serie

for i=2:9 % Aqui, geraremos um vetor x com 8 entradas,
%que marcam o inicio de cada intervalo com falha
    n=y(i-1);
    while teste(n)==0
        x(i)=n;
        n=n+1;
    end
    m=n;
    if m<2500
    while teste(m)==1  % geramos tambem um vetor y com 8 entradas,
    % que marcam o fim dos intervalos com falha
        m=m+1;
        y(i)=m;
    end
```

```
        else
            y(i)=y(i-1);
    end

p0 = [x(i) A_aux3(x(i))]';  % p0 e a primeira extremidade
p1 = [(y(i)-x(i))/2 h_min]'; % p1 e o primeiro ponto de controle
p2 = [(y(i)-x(i))/2 h_max]'; % p2 e o segundo ponto de controle
p3 = [y(i) A_aux3(y(i))]'; % p3 e a segunda extremidade

B = [p0 p1 p2 p3]'; % geramos uma matriz cujas colunas
% correspondem aos vetores encontrados anteriormente

M = [-1 3 -3 1; 3 -6 3 0; -3 3 0 0; 1 0 0 0];       % M e a matriz correspondente
% a curva cubica de Bezier

A = M*B; % Multiplicamos a matriz M da curva pela matriz B dos pontos

u = linspace(0,1,y(i)-x(i)+1); % Geramos um vetor u, com entradas de 0 a 1,
% na mesma quantidade de valores com falha no intervalo observado
u = u';
unit = ones(size(u));
U = [u.^3 u.^2 u unit];

Pu = U*A; % Produto da matriz A pela matriz U

int=Pu(:, 2)'; % Os valores de interesse nesse caso
% sao os valores da segunda coluna da matriz Pu

A_aux3(x(i):y(i))=int; % Substituimos o intervalo com falha
% pelos valores do vetor int
end

A_aux4=A_aux; % Serie na qual realizaremos a interpolacao por polinomios de Chebyshev
teste=isnan(A_aux4); % Encontra os valores de A_aux4 com falha

for i=2:9
    n=y(i-1);
    while teste(n)==0
        x(i)=n;
        n=n+1;
    end
    m=n;
    if m<2500
    while teste(m)==1
        m=m+1;
        y(i)=m;
    end
        else
```

```
            y(i)=y(i-1);
    end
    v=linspace(-1, 1, y(i)-x(i)+1); % Geramos um vetor com o numero de componentes
    % igual ao numero de entradas com falhas no intervalo observado
    chebyshev=4*(v.^3)-3*v; % Aplicamos o polinomio cubico de Chebyshev no vetor v,
    % obtendo uma curva com oscilacao controlada;
    chebyshev=1/2*((A_aux4(y(i))-A_aux4(x(i)))*chebyshev+A_aux4(x(i))+A_aux(y(i)));
    A_aux4(x(i):y(i))=chebyshev; % Substituimos o intervalo com falha
    % pelo vetor obtido
end

A_aux5=A_aux;
for i=2:9
    n=y(i-1);
    while teste(n)==0
        x(i)=n;
        n=n+1;
    end
    m=n;
    if m<2500
    while teste(m)==1
        m=m+1;
        y(i)=m;
    end
        else
            y(i)=y(i-1);
    end
    m=(A_aux5(y(i))-A_aux5(x(i)))/(y(i)-x(i));
    n=A_aux5(y(i))-m*y(i);
    for j=x(i):y(i)
    A_aux5(j)=m*j+n;
    end
end

% Plotagem das series obtidas

figure(01)
plot(A_copia,'k')
title('Serie original')
axis tight
xlabel('Tempo');
ylabel('Magnitude');

figure(02)
plot(A_aux1,'k')
title('Interpolacao por splines cubicos')
axis tight
xlabel('Tempo');
```

```
ylabel('Magnitude');

figure(03)
plot(A_aux2,'k')
title('Interpolacao por polinomio cubico de Hermite')
axis tight
xlabel('Tempo');
ylabel('Magnitude');

figure(04)
plot(A_aux3,'k')
title('Interpolacao por curva cubica de Bezier')
axis tight
xlabel('Tempo');
ylabel('Magnitude');

figure(05)
plot(A_aux4,'k')
title('Interpolacao por polinomio cubico de Chebyshev')
axis tight
xlabel('Tempo');
ylabel('Magnitude');

figure(06)
plot(A_aux5,'k')
title('Interpolacao linear')
axis tight
xlabel('Tempo');
ylabel('Magnitude');

% Aplicando a TWC na serie original e nas series interpoladas
coefs01=cwt1d(fft(A_copia), 'morlet', 1:1:2000, ['export']);
coefs02=cwt1d(fft(A_aux1), 'morlet', 1:1:2000, ['export']);
coefs03=cwt1d(fft(A_aux2), 'morlet', 1:1:2000, ['export']);
coefs04=cwt1d(fft(A_aux3), 'morlet', 1:1:2000, ['export']);
coefs05=cwt1d(fft(A_aux4), 'morlet', 1:1:2000, ['export']);
coefs06=cwt1d(fft(A_aux5), 'morlet', 1:1:2000, ['export']);

% Plotando os escalogramas de cada serie
figure(07)
yashow(coefs01);
xlabel('Tempo b')
ylabel('Escala a')
title('Escalograma - Serie original')

figure(08)
yashow(coefs02);
xlabel('Tempo b')
```

```
ylabel('Escala a')
title('Escalograma - Interpolacao por spline cubico')

figure(09)
yashow(coefs03);
xlabel('Tempo b')
ylabel('Escala a')
title('Escalograma - Interpolacao por polinomio cubico de Hermite')

figure(10)
yashow(coefs04);
xlabel('Tempo b')
ylabel('Escala a')
title('Escalograma - Interpolacao por curva cubica de Bezier')

figure(11)
yashow(coefs05);
xlabel('Tempo b')
ylabel('Escala a')
title('Escalograma - Interpolacao por polinomio cubico de Chebyshev')

figure(12)
yashow(coefs06);
xlabel('Tempo b')
ylabel('Escala a')
title('Escalograma - Interpolacao linear')

% Obtendo os valores para o espectro global
escalograma01=abs(coefs01.*coefs01);
for i=1:2000
    espectro_global01(i)=sum(escalograma01(i,:));
end

escalograma02=abs(coefs02.*coefs02);
for i=1:2000
    espectro_global02(i)=sum(escalograma02(i,:));
end

escalograma03=abs(coefs03.*coefs03);
for i=1:2000
    espectro_global03(i)=sum(escalograma03(i,:));
end

escalograma04=abs(coefs04.*coefs04);
for i=1:2000
    espectro_global04(i)=sum(escalograma04(i,:));
end
```

```
escalograma05=abs(coefs05.*coefs05);
for i=1:2000
    espectro_global05(i)=sum(escalograma05(i,:));
end

escalograma06=abs(coefs06.*coefs06);
for i=1:2000
    espectro_global06(i)=sum(escalograma06(i,:));
end

% Plotagem dos espectros globais
figure(13)
plot(espectro_global01, 1:1:2000, 'k')
set(gca, 'xdir','reverse')
set(gca, 'ydir','reverse')
axis tight
xlabel('Energia')
title('Espectro global - Serie original')

figure(14)
plot(espectro_global02, 1:1:2000, 'k')
set(gca, 'xdir','reverse')
set(gca, 'ydir','reverse')
axis tight
xlabel('Energia')
title('Espectro global - Interpolacao por splines cubicos')

figure(15)
plot(espectro_global03, 1:1:2000, 'k')
set(gca, 'xdir','reverse')
set(gca, 'ydir','reverse')
axis tight
xlabel('Energia')
title('Espectro global - Interpolacao por polinomio cubico de Hermite')

figure(16)
plot(espectro_global04, 1:1:2000, 'k')
set(gca, 'xdir','reverse')
set(gca, 'ydir','reverse')
axis tight
xlabel('Energia')
title('Espectro global - Interpolacao por curva cubica de Bezier')

figure(17)
plot(espectro_global05, 1:1:2000, 'k')
set(gca, 'xdir','reverse')
set(gca, 'ydir','reverse')
axis tight
```

```
xlabel('Energia')
title('Espectro global - Interpolacao por polinomio cubico de Chebyshev')

figure(18)
plot(espectro_global06, 1:1:2000, 'k')
set(gca, 'xdir','reverse')
set(gca, 'ydir','reverse')
axis tight
xlabel('Energia')
title('Espectro global - Interpolacao linear')

% Obtencao da energia total das series
energia_serieoriginal=sum(sum(escalograma01))
energia_spline=sum(sum(escalograma02))
energia_hermite=sum(sum(escalograma03))
energia_bezier=sum(sum(escalograma04))
energia_chebyshev=sum(sum(escalograma05))
energia_linear=sum(sum(escalograma06))

% Obtencao da porcentagem da energia de cada serie interpolada em relacao a original
porcentagem_spline=(energia_spline)/(energia_serieoriginal)
porcentagem_hermite=(energia_hermite)/(energia_serieoriginal)
porcentagem_bezier=(energia_bezier)/(energia_serieoriginal)
porcentagem_chebyshev=(energia_chebyshev)/(energia_serieoriginal)
porcentagem_linear=(energia_linear)/(energia_serieoriginal)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
% CODIGO PARA GERAR OS TESTES COM FALHAS DO TIPO III                           %
%                                                                              %
% Escrito por: Luciano Aparecido Magrini, Paula Neves de Araujo                %
% Data: 19/11/2016                                                             %
%                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all % Remove objetos definidos previamente
clc % Limpa a tela de comandos do MATLAB

A=load('dados.txt'); % Leitura do arquivo de dados.
A_copia=A; % Copia para armazenamento dos dados originais
A_aux=A; % Copia do arquivo de dados.

% Comandos para inserir as falhas
A_aux(751:1000)=nan;
A_aux(2201:2450)=nan;

nan_aux=isnan(A_aux); % A funcao ''isnan'' identifica as posicoes de falha
```

```
%e gera um novo vetor com zeros e valor 1 nas falhas

h_max=max(A_aux); % Encontrando o valor maximo da serie
h_min=min(A_aux); % Encontrando o valor minimo da serie

% Encontrando os indices que correspondem a intervalos com falha
% e os indices que correspondem a dados
indices_com_falha=find(nan_aux==1);
indices_com_falha=indices_com_falha';
indices_com_dados=find(nan_aux==0);
indices_com_dados= indices_com_dados';

% Criacao de um vetor auxiliar cujas entradas
% sao as posicoes conhecidas do arquivo de texto lido
% Sera usado no processo de interpolacao por splines cubicos
A_aux1=A_aux;
cte_aux=length(indices_com_dados);
f=zeros(1,cte_aux);
for d=1:cte_aux
    f(d)=A_aux1(indices_com_dados(d));
end

% Criacao de um vetor auxiliar cujas entradas
% sao as posicoes conhecidas do arquivo de texto lido
% Sera usado no processo de interpolacao por polinomio cubico de Hermite
A_aux2=A_aux;
cte_aux=length(indices_com_dados);
g=zeros(1,cte_aux);
for d=1:cte_aux
    g(d)=A_aux1(indices_com_dados(d));
end

% Usando spline cubica
aprox_spline=spline(indices_com_dados,f,indices_com_falha);
for m=1:length(aprox_spline)
    A_aux1(indices_com_falha(m))=aprox_spline(m);
end

% Usando polinomio cubico de Hermite
aprox_hermite=pchip(indices_com_dados,f,indices_com_falha);
for m=1:length(aprox_hermite)
    A_aux2(indices_com_falha(m))=aprox_hermite(m);
end

% Criacao de um vetor auxiliar cujas entradas
% sao as posicoes conhecidas do arquivo de texto lido
% Sera usado no processo de interpolacao pela curva de Bezier
A_aux3=A_aux;
```

```
n=1; % Inicializa o contador que sera utilizado para gerar a curva
y(1)=1; % Define o valor de y(1), que tambem sera utilizado para gerar a curva
teste=isnan(A_aux3); % Identifica as posicoes com falha na serie

for i=2:3 % Aqui, geraremos um vetor x com 2 entradas,
%que marcam o inicio de cada intervalo com falha
    n=y(i-1);
    while teste(n)==0
        x(i)=n;
        n=n+1;
    end
    m=n;
    if m<2500
    while teste(m)==1  % Geramos tambem um vetor y com 2 entradas,
    % que marcam o fim dos intervalos com falha
        m=m+1;
        y(i)=m;
    end
        else
            y(i)=y(i-1);
    end

p0 = [x(i) A_aux3(x(i))]';  % p0 e a primeira extremidade
p1 = [(y(i)-x(i))/2 h_min]'; % p1 e o primeiro ponto de controle
p2 = [(y(i)-x(i))/2 h_max]'; % p2 e o segundo ponto de controle
p3 = [y(i) A_aux3(y(i))]'; % p3 e a segunda extremidade

B = [p0 p1 p2 p3]'; % geramos uma matriz cujas colunas
% correspondem aos vetores encontrados anteriormente

M = [-1 3 -3 1; 3 -6 3 0; -3 3 0 0; 1 0 0 0];       % M e a matriz correspondente
% a curva cubica de Bezier

A = M*B; % Multiplicamos a matriz M da curva pela matriz B dos pontos

u = linspace(0,1,y(i)-x(i)+1); % Geramos um vetor u, com entradas de 0 a 1,
% na mesma quantidade de valores com falha no intervalo observado
u = u';
unit = ones(size(u));
U = [u.^3 u.^2 u unit];

Pu = U*A; % Produto da matriz A pela matriz U

int=Pu(:, 2)'; % Os valores de interesse nesse caso
% sao os valores da segunda coluna da matriz Pu

A_aux3(x(i):y(i))=int; % Substituimos o intervalo com falha
```

```
% pelos valores do vetor int
end

A_aux4=A_aux; % Serie na qual realizaremos a interpolacao por polinomios de Chebyshev
teste=isnan(A_aux4); % Encontra os valores de A_aux4 com falha

for i=2:3
    n=y(i-1);
    while teste(n)==0
        x(i)=n;
        n=n+1;
    end
    m=n;
    if m<2500
    while teste(m)==1
        m=m+1;
        y(i)=m;
    end
        else
            y(i)=y(i-1);
    end
    v=linspace(-1, 1, y(i)-x(i)+1); % Geramos um vetor com o numero de componentes
    % igual ao numero de entradas com falhas no intervalo observado
    chebyshev=4*(v.^3)-3*v; % Aplicamos o polinomio cubico de Chebyshev no vetor v,
    % obtendo uma curva com oscilacao controlada;
    chebyshev=1/2*((A_aux4(y(i))-A_aux4(x(i)))*chebyshev+A_aux4(x(i))+A_aux(y(i)));
    A_aux4(x(i):y(i))=chebyshev; % Substituimos o intervalo com falha
    % pelo vetor obtido
end

A_aux5=A_aux;
for i=2:3
    n=y(i-1);
    while teste(n)==0
        x(i)=n;
        n=n+1;
    end
    m=n;
    if m<2500
    while teste(m)==1
        m=m+1;
        y(i)=m;
    end
        else
            y(i)=y(i-1);
    end
    m=(A_aux5(y(i))-A_aux5(x(i)))/(y(i)-x(i));
    n=A_aux5(y(i))-m*y(i);
```

```
    for j=x(i):y(i)
    A_aux5(j)=m*j+n;
    end
end

% Plotagem das series obtidas

figure(01)
plot(A_copia,'k')
title('Serie original')
axis tight
xlabel('Tempo');
ylabel('Magnitude');

figure(02)
plot(A_aux1,'k')
title('Interpolacao por splines cubicos')
axis tight
xlabel('Tempo');
ylabel('Magnitude');

figure(03)
plot(A_aux2,'k')
title('Interpolacao por polinomio cubico de Hermite')
axis tight
xlabel('Tempo');
ylabel('Magnitude');

figure(04)
plot(A_aux3,'k')
title('Interpolacao por curva cubica de Bezier')
axis tight
xlabel('Tempo');
ylabel('Magnitude');

figure(05)
plot(A_aux4,'k')
title('Interpolacao por polinomio cubico de Chebyshev')
axis tight
xlabel('Tempo');
ylabel('Magnitude');

figure(06)
plot(A_aux5,'k')
title('Interpolacao linear')
axis tight
xlabel('Tempo');
ylabel('Magnitude');
```

```
% Aplicando a TWC na serie original e nas series interpoladas
coefs01=cwt1d(fft(A_copia), 'morlet', 1:1:2000, ['export']);
coefs02=cwt1d(fft(A_aux1), 'morlet', 1:1:2000, ['export']);
coefs03=cwt1d(fft(A_aux2), 'morlet', 1:1:2000, ['export']);
coefs04=cwt1d(fft(A_aux3), 'morlet', 1:1:2000, ['export']);
coefs05=cwt1d(fft(A_aux4), 'morlet', 1:1:2000, ['export']);
coefs06=cwt1d(fft(A_aux5), 'morlet', 1:1:2000, ['export']);

% Plotando os escalogramas de cada serie
figure(07)
yashow(coefs01);
xlabel('Tempo b')
ylabel('Escala a')
title('Escalograma - Serie original')

figure(08)
yashow(coefs02);
xlabel('Tempo b')
ylabel('Escala a')
title('Escalograma - Interpolacao por spline cubico')

figure(09)
yashow(coefs03);
xlabel('Tempo b')
ylabel('Escala a')
title('Escalograma - Interpolacao por polinomio cubico de Hermite')

figure(10)
yashow(coefs04);
xlabel('Tempo b')
ylabel('Escala a')
title('Escalograma - Interpolacao por curva cubica de Bezier')

figure(11)
yashow(coefs05);
xlabel('Tempo b')
ylabel('Escala a')
title('Escalograma - Interpolacao por polinomio cubico de Chebyshev')

figure(12)
yashow(coefs06);
xlabel('Tempo b')
ylabel('Escala a')
title('Escalograma - Interpolacao linear')

% Obtendo os valores para o espectro global
escalograma01=abs(coefs01.*coefs01);
```

```
for i=1:2000
    espectro_global01(i)=sum(escalograma01(i,:));
end

escalograma02=abs(coefs02.*coefs02);
for i=1:2000
    espectro_global02(i)=sum(escalograma02(i,:));
end

escalograma03=abs(coefs03.*coefs03);
for i=1:2000
    espectro_global03(i)=sum(escalograma03(i,:));
end

escalograma04=abs(coefs04.*coefs04);
for i=1:2000
    espectro_global04(i)=sum(escalograma04(i,:));
end

escalograma05=abs(coefs05.*coefs05);
for i=1:2000
    espectro_global05(i)=sum(escalograma05(i,:));
end

escalograma06=abs(coefs06.*coefs06);
for i=1:2000
    espectro_global06(i)=sum(escalograma06(i,:));
end

% Plotagem dos espectros globais
figure(13)
plot(espectro_global01, 1:1:2000, 'k')
set(gca, 'xdir','reverse')
set(gca, 'ydir','reverse')
axis tight
xlabel('Energia')
title('Espectro global - Serie original')

figure(14)
plot(espectro_global02, 1:1:2000, 'k')
set(gca, 'xdir','reverse')
set(gca, 'ydir','reverse')
axis tight
xlabel('Energia')
title('Espectro global - Interpolacao por splines cubicos')

figure(15)
plot(espectro_global03, 1:1:2000, 'k')
```

```
set(gca, 'xdir','reverse')
set(gca, 'ydir','reverse')
axis tight
xlabel('Energia')
title('Espectro global - Interpolacao por polinomio cubico de Hermite')

figure(16)
plot(espectro_global04, 1:1:2000, 'k')
set(gca, 'xdir','reverse')
set(gca, 'ydir','reverse')
axis tight
xlabel('Energia')
title('Espectro global - Interpolacao por curva cubica de Bezier')

figure(17)
plot(espectro_global05, 1:1:2000, 'k')
set(gca, 'xdir','reverse')
set(gca, 'ydir','reverse')
axis tight
xlabel('Energia')
title('Espectro global - Interpolacao por polinomio cubico de Chebyshev')

figure(18)
plot(espectro_global06, 1:1:2000, 'k')
set(gca, 'xdir','reverse')
set(gca, 'ydir','reverse')
axis tight
xlabel('Energia')
title('Espectro global - Interpolacao linear')

% Obtencao da energia total das series
energia_serieoriginal=sum(sum(escalograma01))
energia_spline=sum(sum(escalograma02))
energia_hermite=sum(sum(escalograma03))
energia_bezier=sum(sum(escalograma04))
energia_chebyshev=sum(sum(escalograma05))
energia_linear=sum(sum(escalograma06))

% Obtencao da porcentagem da energia de cada serie interpolada em relacao a original
porcentagem_spline=(energia_spline)/(energia_serieoriginal)
porcentagem_hermite=(energia_hermite)/(energia_serieoriginal)
porcentagem_bezier=(energia_bezier)/(energia_serieoriginal)
porcentagem_chebyshev=(energia_chebyshev)/(energia_serieoriginal)
porcentagem_linear=(energia_linear)/(energia_serieoriginal)
```