Universidade Feral de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

Execuções de operações e algoritmos em dados no domínio comprimido

Gustavo Simões Carnivali

JUIZ DE FORA
DEZEMBRO, 2016

Execuções de operações e algoritmos em dados no domínio comprimido

Gustavo Simões Carnivali

Universidade Feral de Juiz de Fora
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Orientador: Alex Borges Vieira
Coorientador: Orestes Piermatei Filho

JUIZ DE FORA
DEZEMBRO, 2016

EXECUÇÕES DE OPERAÇÕES E ALGORITMOS EM DADOS NO DOMÍNIO COMPRIMIDO

Gustavo Simões Carnivali

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Alex Borges Vieira
Doutor em Ciência da Computação (UFMG)

Orestes Piermatei Filho
Doutor em Matemática (UFRJ)

Heder Soares Bernardino
Doutor em Modelagem Computacional (LNCC/MCTI)

Francisco Henrique Cerdeira Ferreira
Mestre em Ciência da Computação (UFJF)

JUIZ DE FORA

16 DE DEZEMBRO, 2016

Aos meus pais e irmão, pelo apoio.

Resumo

Dados em larga escala são cada dia mais comuns para a computação. Tais dados geram uma classe de serviço que impõe fortes requisitos na infraestrutura e no ambiente de execução. Nesse trabalho estudamos uma representação de grandes volumes de dados a partir da Transformada de Fourier. Apresentamos características que tornam a transformada útil na representação de séries temporais, operações que podem ser feitas com os dados representados pela transformada e exemplos de preditores e geradores de carga implementado a partir desses estudos. Nos experimentos, se alcançou aumentos na velocidade do algoritmo sem uma perda substancial da qualidade dos seus resultados.

Palavras-chave: Processamento, Comprimido, Domínio, Transformada, Fourier, Previsão.

Abstract

Big Data are increasingly more common for computation, Those datas generate the class of service that require strong requisits on infrastructure and at the execution ambient. Here is studied the representation of big data volumes with the Fourier Transform. We present some caracteristics that make the Fourier Transform usefull at functions representation, operations that can be made with represented datas by the transformed and exemples of predicts and implemented charge generators from those studies, with the objective to rise up the algorithm velocity without a substancial loss quality of results.

Keywords: Processing, Compressed, Domain, Transform, Fourier, Prediction.

Agradecimentos

Aos meus pais por me apoiarem em todas as minhas escolhas e por serem parte integral dessa conquista.

Ao meu irmão por sempre ser uma fonte de inspiração.

A todos meus familiares e amigos pelo apoio constante.

Ao meu orientador Alex Vieira por me trilhar pelo caminho da Ciência.

Ao meu co-orientador Orestes Piermatei por tantos ensinamentos.

Ao meu ex-orientador Marcelo Bernardes por me iniciar no caminho da Ciência.

A todos os professores, estudantes e membros da UFJF por me permitir tão nobre formação.

A todos que de alguma forma contribuíram com este projeto.

A UFJF, Fapemig e Fadepe pelos apoios.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

DCC	Departamento de Ciência da Computação
UFJF	Universidade Federal de Juiz de Fora
DCT	Discrete Cosine Transform
JPEG	Joint Photographic Experts Group

1 Introdução

Com a evolução dos sistemas computacionais, a tendência de mídias ou outras formas de representações de informação que são baseadas em grandes volumes de dados se tornaram algo comum no cotidiano. Em geral, esses dados são muito utilizada por empresas prestadoras de serviço como jornais, revistas e rádios, ou por uma necessidade e interesses pessoais, como ocorre em redes sociais, onde é exigido sempre maiores demandas de informação (Stone et al, 2014).

Portanto, se torna notório a existência da necessidade desses setores oferecerem ou armazenarem essas informações da melhor forma. Sob essa necessidade, estudos sobre a eficiência na compactação envio ou manipulação de mídias se tornam uma realidade do meio acadêmico. Em particular, para a economizar no custo de armazenamento e envio, a maioria das mídias são guardadas em formato compactado (Dugad et al, 2001). Neste contexto, a compactação pode ser vista como um ramo da teoria da informação na qual se objetiva e estuda a transmissão de menores quantidades de dados (Lelewe et al, 1987).

O processamento em um domínio comprimido ocorre quando algum dado, em seu estado comprimido, recebe alguma manipulação. Historicamente, vários processos são usados para compactação de dados, em específico, o uso de transformadas é muito comum. O padrão de imagens JPEG, por exemplo, em suas várias implementações, já usou para compactação a DCT (Transformada Discreta do Cosseno) um caso específico da Transformada de Fourier (Hu et al, 2010) e sua última versão JPEG 2000 utiliza principalmente Wavelets para o processo de compactação (Marcellin et al, 2000). No caso de sons, temos o exemplo dos formatos MP1, MP2 e MP3 do padrão MPEG áudio e MPEG Video que utilizam variações da Transformada de Fourier também para a compactação (Mitchell et al, 1997).

Em Santos et al (2004), por exemplo, tem-se que cada coeficiente da Transformada de Fourier integra a representação de um harmônico fundamental da função. Pode-se interpretar a transformada como uma "soma de infinitos harmônicos cujas frequências são múltiplos inteiros de sua frequência fundamental". Cada coeficiente na Transformada de

Fourier representa uma curva periódica da função. Essa propriedade, de evidenciar curvas periódicas, é um dos motivos pelos quais a Transformada de Fourier e suas derivadas são muito utilizadas para compressão nos principais formatos de mídia. Ao evidenciar partes periódicas, partes aperiódicas ou aleatórias da função, que em geral são menos importantes, se tornam mais facilmente identificadas e ignoradas. Assim, pode-se representar a função com menos dados ou realizar uma compressão com perdas (Hu et al, 2010). Nesse caso ocorre uma compactação com perdas pois, é a partir da retirada de informações menos importantes que consegue-se uma representação com menores quantidades de dados, ou seja, um dado comprimido.

Transformadas como essas, em geral, são aplicações que levam vetores de um espaço vetorial em outro e nesse novo espaço de vetores os dados são apresentados de uma nova forma e podem representar a informação original com menores quantidades de dados. Com isso algoritmos desenvolvidos em cima desses domínios, por conterem menores dados, necessitam percorrer menores conjuntos para realizar seus objetivos.

Portanto, sob o efeito dessas transformadas, novas formas de manipulação desses dados são necessárias. Com a Transformada de Fourier, pode-se considerar que passa-se a manipular frequências fundamentais. Porém, estudos que analisam e propõem algoritmos de manipulação desses dados em sua forma comprimida já são comuns com exemplos facilmente encontrados na literatura. Alguns exemplos que utilizam a DCT nesse processo podem ser citados como um estudo das mudanças no tamanho de imagens (Dugad et al, 2001), incorporação de marcas d'água em vídeos (Meng et al, 1998), detecção e rastreamento de sons (Wang et al, 2001) e sumarização de músicas (Shao et al, 2004).

Ainda é muito comum encontrar livros que trazem várias aplicações onde transformadas como a DCT são utilizadas. Em Rao et al (2014), por exemplo, encontram-se aplicações da DCT como filtragens e codificações de imagens, compressão de dados, reconhecimento de padrões, entre outros.

O problema, então, se compõe em tentar aplicar essas possíveis melhorias geradas pelas transformadas em diversas áreas, no caso de Big Data, mesmo processamentos simples podem ser custosos pelo tamanho dos dados utilizados. Nesse contexto, viu-se que algumas transformadas são muito utilizadas para compactação e, em outros momentos,

para realização de algum processamento. Portanto o problema do trabalho consiste em unir esses dois problemas afim de atacar a lentidão que pode existir em processamentos em Big Data. Em um único contexto, com o apoio de uma transformada, pretende-se compactar os dados e, depois, com os dados compactados, realizar algum processamento neles. Para isso, inicialmente tem-se que considerar uma forma de representar os dados do problema no domínio comprimido; pensar em quais propriedades que se quer considerar das informações disponíveis; e selecionar um conjunto de transformadas que aplicadas a esses dados mantenha suas informações importantes e torne sua manipulação fácil.

Após isso, é considerado também quais operações que podem ser realizadas nesse novo domínio, afim de aplicá-las em processamentos futuros. Precisa-se permitir que operações realizadas no domínio original dos dados sejam possíveis de ser realizadas no domínio comprimido mantendo sua eficiência computacional ou a melhorando. Tal condição é necessária pois, a aplicação do processamento em uma nova representação de dados, não faria sentido se não obtivesse melhorias em relação a forma original.

A aplicação de transformadas, objetivando a compactação, pode gerar erros para os dados. É necessário analisar o quanto de erro se produz enquanto mais se compacta os dados. No contexto de problemas da área de Big Data. Tem-se, o problema do alto número de dados que dificulta a análise necessária para o processo, necessitando de uma forma diferenciada que análise esses dados de forma mais eficiente. A partir dessa dificuldade surge a motivação, ao considerarmos que a Big Data se torna uma realidade, soluções criativas devem ser geradas para esses problemas. Considerando que a Transformada de Fourier foi criada em 1822, que em 1975 N. Ahmed, T. Natarajan and K. R. Rao já criara um algoritmo rápido para realizá-la (Ahmed et al, 1974). Em 1965, Cooley e Tukey(Cooley et al, 1965) desenvolveram a DCT, uma clássica modificação da Transformada de Fourier. Nota-se que o desenvolvimento gerado a partir da transformada é alto e rápido, o que estimula a acreditar na sua alta eficiência na solução de vários problemas, incluindo problemas da área de Big Data.

Veja que um termo que será bastante utilizado nesse trabalho é o termo Big Data, porém, Maçada et al (2015) considera que ele está sujeito a algumas caraterizações que são descritas por 5 V's: Volume, Velocidade, Variedade, Veracidade e Valor. Onde

o Volume representa os altos volumes de dados. Velocidade que implica na velocidade em que tais dados devem ser tratados. Variedade que implica que os dados podem ser compostos por vários tipos com caracteristicas diferentes. Veracidade que implica que sua base de dados deva ser confiável e, por fim, Valor que traz a importância de aplicar trabalho a um conjunto de dados. Dos 5 V's, o Big Data usado neste trabalho requer altos volumes para implicar melhorias na velocidade de compactação, velocidade por objetivar aumentar a velocidade de tratamento desses dados. Veracidade, em considerar que a maior parte dos dados representam informações reais e valor, já que o trabalho é feito em cima de problemas reais. O único não representativo no trabalho é a Variedade já que, em geral, o trabalho é totalmente feito com funções unidimensionais.

Considerando toda teoria explicitada acima, acredita-se que é possível definir um conjunto mínimo de operações atuando em domínios comprimidos conhecidos, ao qual dados compactados podem sofrer qualquer tipo de alteração. Assim, o objetivo é a criação de um conjunto de operações que permitam a execução de diversos algoritmos em dados compactados.

Para isso, esse trabalho apresenta os seguintes objetivos. Inicialmente, quer-se indicar representações de dados que consigam modificações de forma mais rápida e eficiente. Assim como analisar formas com as quais essas manipulações, sobre os dados, poderão ser feitas. Na análise do erro gerado na utilização da transformada objetiva-se criar padrões que evidenciem a qualidade que determinada operação pode ter em um problema.

Para isso, inicia-se esse trabalho contextualizando a principal transformada utilizada, a Transformada de Fourier, será apresentada sua fórmula e uma análise sobre ela. Será mostrado o funcionamento da compactação com mudança de domínio, por que ela é possível e quando ela é indicada, serão mostrados casos bons e ruins onde a compactação gera resultados próximos ou muito distantes das função original .

Será percorrida a literatura em busca de leis que regem as transformadas e que possam ser úteis futuramente. A partir dessas leis e de implicações que surgem das fórmulas, é desenvolvido operações que podem ser realizadas no domínio comprimido. São analisadas essas operações em diversos contextos, tentando evidenciar momentos em

que podem ou não ser úteis, abrindo um leque de possíveis algoritmos que poderiam ser otimizados.

Por fim, após toda definição gerada, é aplicado às operações criadas em problemas reais a fim de possibilitar sua análise. É aplicada a transformada em algoritmos de previsão clássicos na literatura, aplicando o algoritmo modificado pela transformada num conjunto de dados diferentes a fim de analisar propriedades e a eficiência da aplicação da transformada nesse problema.

Por fim, é criado e estudado um algoritmo completo de geração de carga de uma série temporal onde, a partir das operações demonstradas ao longo do trabalho, é gerado um algoritmo de previsão completo, funcionando totalmente no domínio comprimido. É analisada e estudada a eficiência temporal do algoritmo, especificando a complexidade de todas as operações que os compõem e é analisada a qualidade dos resultados obtidos.

2 Transformada de Fourier

A Transformada de Fourier pode ser interpretada como uma função que identifica diferentes frequências senoidais (e suas respectivas amplitudes) que se combinam de forma arbitrária para definir uma onda Brigham et al (1974). Oppenheim et al (2010) define a Transformada Discreta de Fourier(TDF) como,

$$F(n) = \sum_{x=0}^{L} f(x) exp\left(-i\frac{2\pi x.n}{L}\right) \tag{2.1}$$

Expandindo a exponencial complexo a transformada também pode ser escrita como segue abaixo:

$$F(n) = \frac{1}{2\pi}\sum_{x=0}^{L} f(x).cos\left(\frac{x.n.\pi}{L}\right) - i.f(x).sen\left(\frac{x.n.\pi}{L}\right) \tag{2.2}$$

Onde L é o número de pontos de f e N o número de coeficientes da Transformada de Fourier e L o número de pontos da função. Devido ao número complexo presente na Transformada a fórmula é comumente separada em dois termos, $C(n)$ e $S(n)$ onde $C(n)$ representa a parte real da fórmula ou do cosseno e $S(n)$ a parte imaginária da fórmula ou do seno. Apresentadas nas fórmulas 2.3 e 2.4.

$$C(n) = \frac{1}{2\pi}\sum_{x=0}^{L} f(l).cos\left(\frac{x.n.\pi}{L}\right) \tag{2.3}$$

$$S(n) = \frac{1}{2\pi}\sum_{x=0}^{L} .F(x).sen\left(\frac{x.n.\pi}{L}\right) \tag{2.4}$$

A forma discreta inversa da Transformada de Fourier é dada pela fórmula 2.5 onde, $C(n)$ e $S(n)$ são respectivamente a parcela do cosseno e do seno gerados pela Transformada.

$$f(x) = \sum_{n=0}^{N} C(n).cos\left(\frac{x.n.\pi}{L}\right) + i.S(n).sen\left(\frac{x.n.\pi}{L}\right) \tag{2.5}$$

Neste trabalho será usado principalmente as fórmulas dadas por 2.2 e 2.5.

2.1 Convergência da Transformada de Fourier

A partir da convergência da Transformada de Fourier temos o controle de até que ponto se pode usar a Transformada de Fourier para comprimir dados tendo uma perda aceitável. A principal motivação de se estudar a convergência é que, futuramente, a qualidade da transformação sobre uma função com um determinado índice de compactação, aplicará diretamente na qualidade de operações ou algoritmos que utilizarão a Transformada de Fourier.

Em Gonçalves et al (2004) é explicitado que se f(x) for absolutamente aditiva, em outras palavras a propriedade $\sum_{n=-\infty}^{\infty} |f(x))| < \infty$, então a série é dita absolutamente convergente. Esse resultado é de grande interesse, pois, a partir dele, se tem que a representação de qualquer função pela Transformada de Fourier pode, com um N devidamente grande, representar a função original com perdas despreziveis. Ou seja, o resultado apresentado na função abaixo é válido.

$$\lim_{N\to\infty} (f(x) - F_N(x)) = 0, \tag{2.6}$$

onde f é a função original e F_n é a função f reconstruída pela Transformada de Fourier com N coeficientes.

Porém, para este trabalho, se tem que N tendendo ao infinito não é de grande interesse, pois isso se torna um resultado intratável para a computação. Portanto, analisar-se-á o resultado para $\lim_{N\to N'}$ onde N' é uma constante conhecida e $N' < \infty$.

Será estudado como como $f(x) - F_N(x)$ se comporta para variados N. Para isso apresentar-se-á uma função de exemplo, a figura 2.1 apresenta o número de espectadores durante uma partida de futebol durante a copa do mundo de 2014. Usa-se a Transformada de Fourier para reconstruir a função e varia-se o valor de N nessas representações.

Enquanto os dados originais têm mais de uma centena de milhão de informações, uma representação da reconstrução com Transformada de Fourier com apenas 3 coeficientes (N=3) já é visto uma boa aproximação da integral total da função original. Com N = 5 já se é capaz de verificar variações como primeiro e segundo tempo (1800s a 4500s e 5400s a 8100s respectivamente); com N = 10, especificidades como o começo da função já são consideradas e, com N = 100, se tem uma caracterização visualmente muito boa da

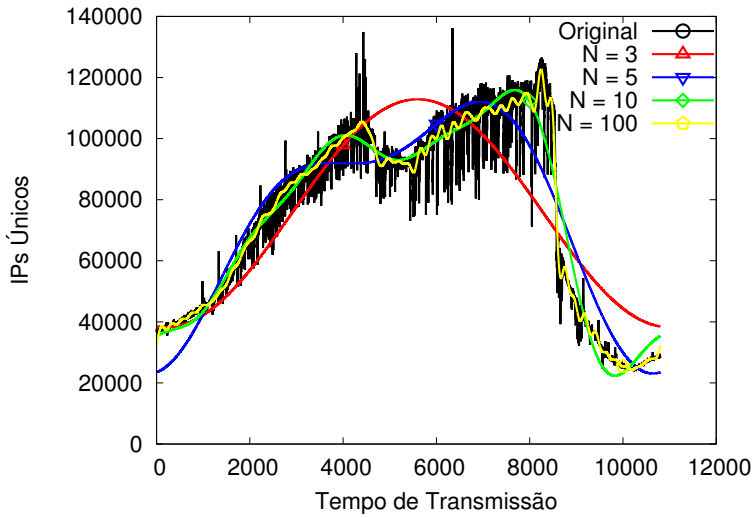

Figura 2.1: Gráfico do jogo Brasil e Alemanha com a reconstrução da função pela Transformada de Fourier.

função.

Isso sugere que a tendência da função $f(x) - F_N(x)$ é realmente convergir para 0 com o aumento do valor de N. Essa tendência pode ser mais percebida na figura 2.2 onde é mostrado o erro gerado com a variação de N na função anterior, é calculado a média do modulo da diferença entre cada ponto da função original com cada ponto da função reconstruída para cada N. Percebe-se que há uma convergência rápida para baixos valores de erro relativo. É possível ver que, com N = 20, já se obtém um erro em torno de apenas 4%.

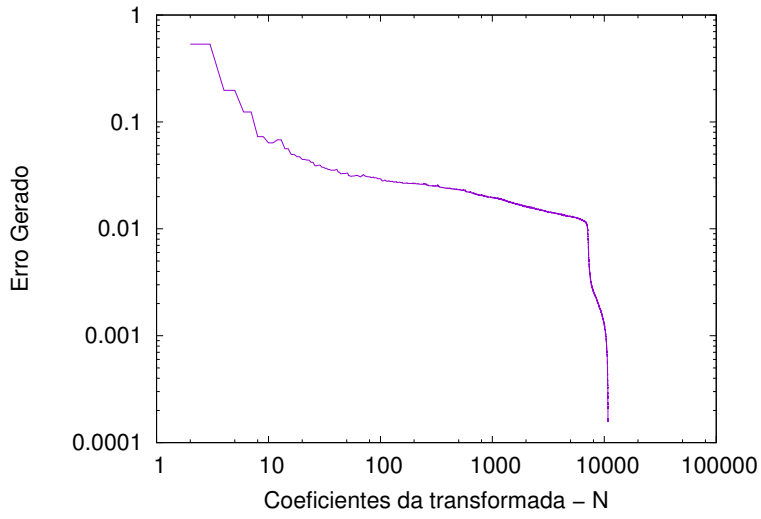

Figura 2.2: Erro relativo entre os dados originais e a representação de Fourier com N coeficientes de F1.

Até aqui, se mostrou, uma possível tendência de convergência da Transformada

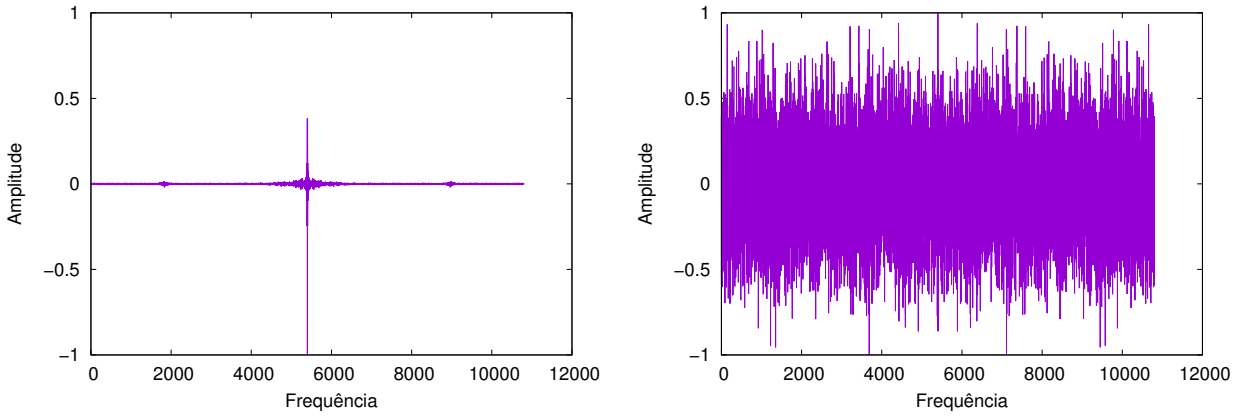

Figura 2.3: Representação do cosseno normalizado da Transformada de Fourier para as funções F1 e F2

de Fourier. Para um exemplo de uma série temporal real se viu que a transformada converge rapidamente para um erro próximo de 0 assim como se pode supor do limite 2.6.

2.2 Convergência da Transformada de Fourier relativo a Função

Mostrou-se que para N tendendo a infinito foi mostrado que o erro gerado pela Transformada de Fourier tende a 0. Fora isso se observou que para um exemplo específico o erro gerado também se aproxima de 0 com o aumento de N. Mostrar-se-á agora que esse erro se comporta de forma diferente para variadas funções. Para isso comparar-se-á duas funções, a função especificada na figura 2.1 que se chamará de F1 e uma nova função F2 definida também no domínio de 0 a 10800, onde os pontos são escolhidos aleatoriamente durante todo o domínio.

Pela característica aleatória dos pontos de F2 é esperado que sua derivada média seja muito maior que de F1. Isso é percebido na imagem 2.3 que apresenta a representação da parte do cosseno das funções F1 e F2 onde se pode ver que para F1 a Transformada de Fourier tem grande representação nos coeficientes de índices baixos enquanto F2 é determinada tanto por coeficientes de índices baixos quanto altos.

Essa característica torna F2 pouco eficiente de ser trabalhada pela Transformada de Fourier. Em F1 tem-se apenas que utilizar a parte central da função cosseno para ter ampla representação, podendo ignorar frequências altas. Porém para F2, caso se ignore

frequências altas, é perceptível que se terá alta perda de informação. Isso é visto na figura 2.4 que mostra o erro gerado pela representação pela Transformada de Fourier variando o valor de N como foi feita na seção anterior.

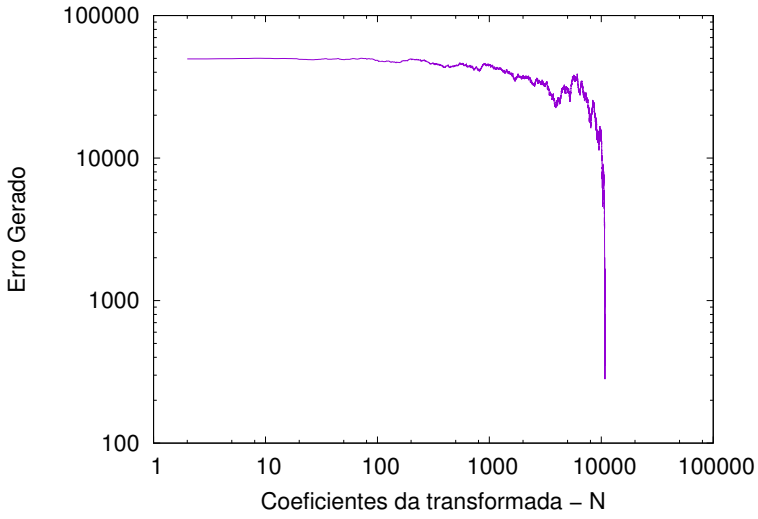

Figura 2.4: Erro relativo entre os dados originais e a representação de Fourier com N coeficientes de F2.

Comparando as imagens 2.2 e 2.4 é visto que além do erro gerado pela função F2 ser maior que para F1, mesmo em valores de N relativamente altos, a convergência do erro para 0 em F2 acontece bem mais lentamente que para F1.

Um caso interessante ocorreu na Figura 2.4. A função possui um caimento quase contínuo porem próximo de N=10000 coeficientes ocorre uma elevação incomun. Isso é possivelmente gerado por um fenômeno chamado Fenômeno de Gibbs. O Fenômeno de Gibbs é caracterizado como a geração de grandes oscilações próximas a pontos de descontinuidade na função, onde tais variações não desaparecem mesmo com um alto número de coeficientes (Gottlieb et al, 1997). Veja que F2, pela característica randômica de seus pontos, possui varias áreas de descontinuidade o que poderia gerar tal fenômeno.

A conclusão que se pode tirar dessa comparação é que a Transformada de Fourier possui índices diferentes de convergência do erro para diferentes funções. Portanto, mesmo que F2 tenha sido um caso extremo onde todos os pontos são gerados aleatoriamente, é interessante que haja uma análise prévia das funções utilizadas pela transformada para determinar a eficiência da sua utilização.

2.3 Convergência da Transformada de Fourier relativo a Primeira Derivada

Foi possível ver na seção 2.2, que a convergência do erro da Transformada de Fourier varia de acordo com a função. Para isso, se comparou o erro gerado por duas funções F1 e F2. Uma análise interessante vem a partir desse ponto, o que faz as funções F1 e F2 terem comportamentos tão diferentes em relação a convergência da transformada? Vários fatores podem contribuir para isso, um deles, que se analisará nessa seção, é a derivada média da função que é pequena em F1 e grande em F2.

Para isso, se criará um conjunto de 500 funções $L_1 ... L_{500}$ onde cada L_n é definida como na regra 2.7.

$$\begin{cases} L_n(0) = 1 \\ L'_n(x) = n \end{cases} \tag{2.7}$$

Assim como foi feito nas seções anteriores, se avaliará o erro gerado por cada L_n variando o valor de N, para isso, para cada n se calculará a variação média gerada pela função reconstruída com N coeficientes e a função original. O resultado é apresentado na figura 2.5.

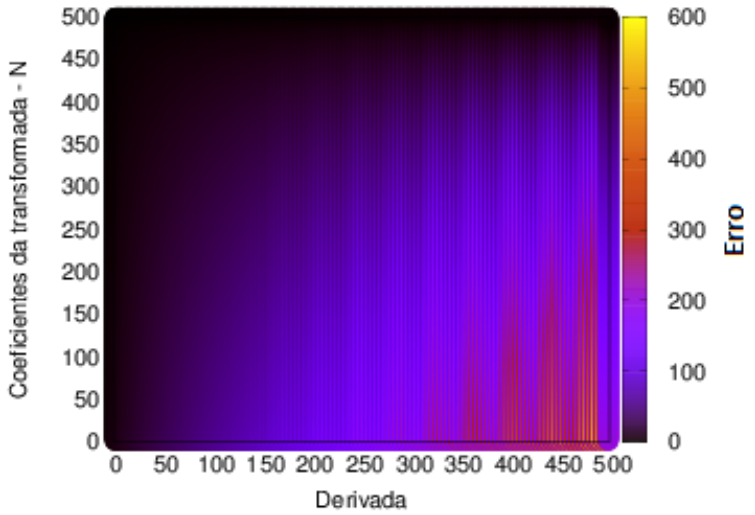

Figura 2.5: Erro relativo entre os dados originais e a representação de Fourier variando o número de coeficientes da Transformada e a derivada da função.

Pode-se concluir pela imagem que, de fato, a derivada média da função altera a convergência da Transformada nesse exemplo. É possível perceber que, mesmo ignorando

o aumento da derivada, caso possua-se um número suficientemente grande de coeficientes o erro tende a zero, mas que a velocidade de convergência é menor para valores diferentes de derivada. Por exemplo, para 200 coeficientes, tem-se um erro mínimo para baixas derivadas, mas erros relativamente grandes para derivadas altas.

Conclui-se nessa seção uma possível tendência da Transformada de Fourier onde a derivada média da função varia a convergência total da Transformada, e que o incremento do erro com o aumento da derivada pode ser realmente expressivo, o que mostra a importância de fazer uma análise da derivada da função antes de utilizá-la em conjunto com a Transformada de Fourier.

2.4 Convergência da Transformada de Fourier relativo a Segunda Derivada

Se fará um estudo análogo ao apresentado na seção anterior, para a primeira derivada, com a segunda derivada. Uma hipótese interessante é que se a primeira derivada pode variar tanto a convergência, a segunda derivada também poderá variar a convergência calculada.

Para analisar esse resultado usar-se-á uma metodologia semelhante à exposta na seção anterior, criando um conjunto de 500 funções $L_1 ... L_{500}$ onde cada L_n é definida como na regra 2.8.

$$\begin{cases} L_n(0) = 1 \\ L'_n(0) = 1 \\ L''_n(x) = n \end{cases} \tag{2.8}$$

A mesma análise de erro será feita sobre essas operações, onde serão reconstruidas 500 funções variando o valor total de frequências N utilizadas em cada reconstrução e se calculará o erro médio gerado em cada função para cada N. O resultado dessa análise pode ser visto na imagem 2.6.

A imagem apresenta um novo resultado com uma conclusão semelhante a anterior. É possível ver que, com o aumento da segunda derivada da função, o erro total gerado por cada N também aumenta. Além disso, é visto que o erro gerado em segundas derivadas

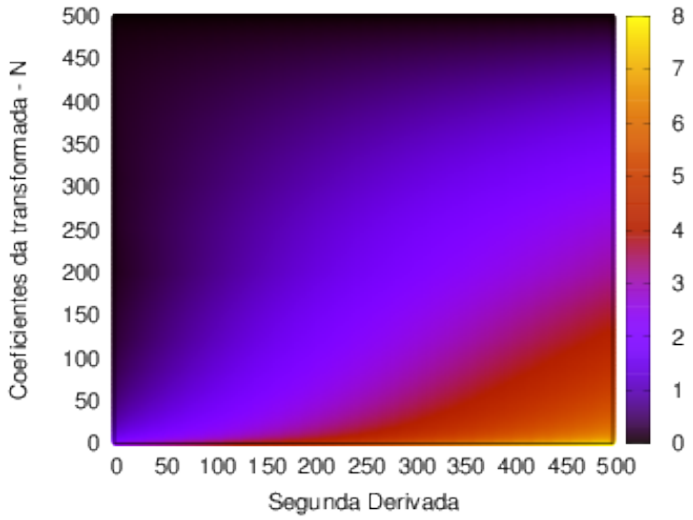

Figura 2.6: Erro relativo entre os dados originais e a representação de Fourier variando o número de coeficientes da Transformada e a derivada da função.

maiores demora mais para convergir para 0 ou para pequenos erros. Isso mostra outra possível tendência relativa à convergência da Transformada de Fourier, onde a segunda derivada média de uma função varia o erro produzido na reconstrução da função pela transformada e na sua convergência para uma reconstrução sem erros.

2.5 Convergência da Transformada de Fourier relativo a Derivada por Ponto

Se observou nas seções anteriores uma possível tendência entre a derivada média de uma função e a convergência da Transformada de Fourier. Tentando encontrar ainda mais essas propriedades que relacionam a convergência se analisará um detalhamento da seção anterior.

Será usado um conjunto de funções com uma derivada igual em todos os pontos da função. Naturalmente essa não é uma propriedade comum, por isso nessa seção se estudará outra série temporal que representa novamente os acessos a uma exibição online do jogo de futebol entre Equador e Franca. A função é especificada na figura 2.7.

Usar-se-á essa nova função para analisar a convergência pontual que ocorre na função, a fim de obter uma relação entre a convergência encontrada e a derivada média da função. Assim, analisar-se-á se isto implicará em outra possível tendência entre a derivada

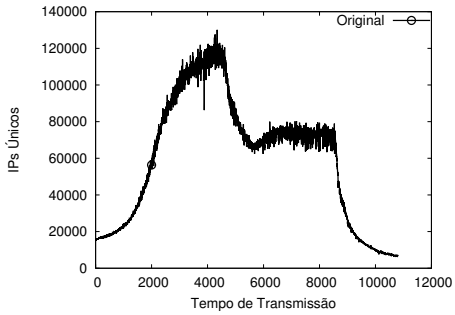

Figura 2.7: Gráfico do jogo Equador e França.

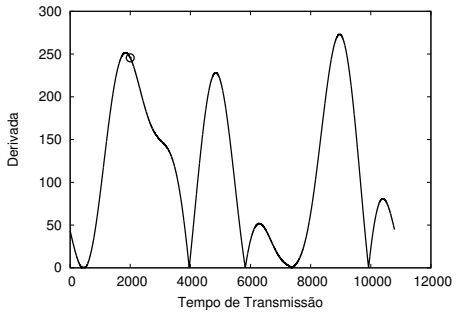

Figura 2.8: Gráfico da derivada da função do jogo Equador e França.

pontual e a convergência presente naquele ponto.

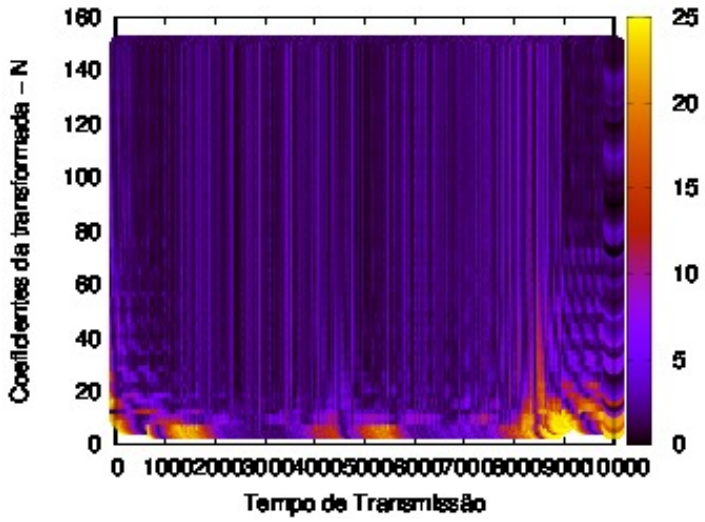

Figura 2.9: Erro relativo entre os dados originais e a representação de Fourier variando o número de coeficientes da Transformada e a derivada da função.

Para isso será analisados outros dois gráficos, o gráfico da imagem 2.8 apresenta a derivada pontual do jogo mostrado na imagem 2.7. A imagem 2.9 apresenta o erro gerado pela variação de N na reconstrução da função com a Transformada de Fourier em cada ponto de 0 a 10800 da função original.

É possível perceber que, apesar do aumento de N, o erro tende a valores baixos. Erros altos para valores baixos de N acontecem principalmente onde a derivada da função é maior. Por exemplo, no tempo 9000s se tem a maior derivada da função do jogo e para N baixo se tem os maiores erros da função.

Aqui se mostrou outra possível tendência da convergência da Transformada de Fourier onde, para esse exemplo, se encontra uma relação entre a derivada pontual e o erro gerado na reconstrução com a transformada.

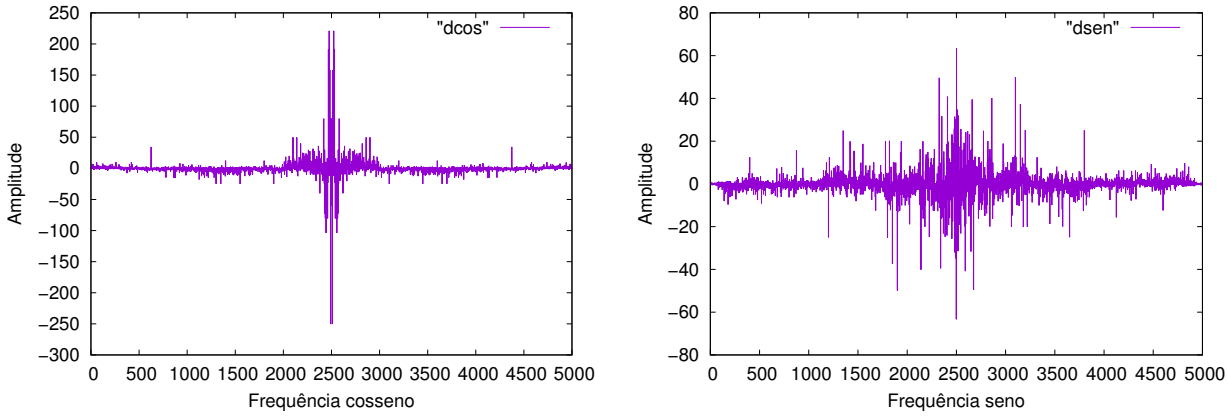

Figura 2.10: Representaçao dos coeficientes da função constante, respectivamente, parcela do cosseno e do seno.

2.6 Convergência da Transformada de Fourier para a Função Constante

Como visto na seção 2.3 e 2.4 é esperado que uma função que apresente uma baixa primeira derivada e segunda derivada, deveria convergir rápido para erros próximos de 0 na reconstrução com a Transformada de Fourier. Mas isso, em geral, não ocorre para uma função que tenha tanto a primeira quanto à segunda derivada zerada por todo seu percurso, ou seja, uma reta.

Dada uma função f definida no domínio [0,5000] dada por $f(x) = K$ sendo K uma constante, a figura 2.10 representa os coeficientes da Transformada de Fourier sobre a função f, respectivamente, a parcela do cosseno e do seno da função f.

É possível perceber na imagem 2.10 que de fato há uma maior concentração da carga total das funções no centro delas, porém ainda se tem uma carga relativamente alta quando se afasta do centro. Na função cosseno, apenas antes de 2000 e após 3000 que se tem uma representação realmente próxima de 0, ou seja, precisar-se-ia de mil frequências para uma representação razoável. Essa situação é ainda pior no seno, que, não apresenta uma alta convergência em nenhuma área da sua função. Essa análise mostra que caso se queira reconstruir a função constante iria precisar de um alto número de frequências.

Perceba que nesse exemplo, uma função que possui derivada nula gerou um erro relativamente alto. Isso mostra que, os resultados mostrados aqui realmente representam tendências. Em um estudo mais específico sobre uma função será interessante analisar sua

primeira e segunda derivada, mas o resultado obtido nessa análise não deve gerar certezas sobre a convergência da função.

2.7 Convergência da Transformada de Fourier Irrestrita a Propriedades

Viu-se nas seções anteriores que a convergência da Transformada de Fourier pode depender de fatores como a derivada pontual da função, mas é esperado que a convergência dependa de vários outros fatores que definem uma função além dos analisados até aqui. Embasado nisso, se irá analisar o erro gerado no decorrer da convergência para entender os fatores que contribuem para esse erro. Será usado a própria formula da Transformada de Fourier para saber com exatidão a formula que mostra o erro final gerado pela transformada para, assim, ser possível definir propriedades concretas que definem esses erros.

Considerando F_N a função f reconstruída com N coeficientes da Transformada de Fourier, o erro pontual $E(x)$ gerado pela retirada de um coeficiente I dessa reconstrução é dado por $E(x) = F_N(x) - F_{N-1}(x)$. Substituindo a expressão gerada com a Transformada Inversa de Fourier obtém-se:

$$F_m(x) - F_{m-1}(x) = \sum_{n=0}^{N} C(n).cos\left(\frac{n.x.\pi}{L}\right) + i.S(n).sen\left(\frac{n.x.\pi}{L}\right) - \sum_{n=0}^{N-1} C(n).cos\left(\frac{n.x.\pi}{L}\right) + i.S(n).sen\left(\frac{n.x.\pi}{L}\right) \tag{2.9}$$

Que é equivalente a:

$$F_N(x) - F_{N-1}(x) = C(I).cos\left(\frac{x.I.\pi}{L}\right) + i.S(I).sen\left(\frac{x.I.\pi}{L}\right) \tag{2.10}$$

Expandindo $C(I)$ e $S(I)$ obtém-se:

$$F_N(x) - F_{N-1}(x) = \frac{1}{2\pi}\left(\sum_{l=0}^{L} f(l).cos\left(\frac{I.l.\pi}{L}\right)\right).cos\left(\frac{x.I.\pi}{L}\right) + \left(\sum_{l=0}^{L} f(l).sen\left(\frac{I.l.\pi}{L}\right)\right).sen\left(\frac{x.I.\pi}{L}\right) \tag{2.11}$$

Conclui-se desse resultado que o erro gerado em um ponto x qualquer da reconstrução, com a retirada de uma frequência I qualquer, depende de todos os pontos $f(n)$ da

função, da posição n que o ponto esta e do tamanho L da função. Isso implica que o erro gerado em um ponto n com a retirada de uma frequência I depende de todo o conjunto de pontos da função tornando imprecisa qualquer análise de convergência sobre uma única propriedade.

3 Operações com a Transformada de Fourier

O objetivo do trabalho é tratar grandes volumes de dados, usando a compactação possibilitada pela Transformada de Fourier vista no capítulo 2, mas, para isso, é preciso antes saber como tratar esses dados no domínio comprimido, já que operações realizadas no domínio comprimido não geram o mesmo efeito sobre os dados que as mesmas operações realizadas no domínio original. Será reafirmada essa frase que compõem o principal problema do trabalho com um exemplo.

Deseja-se somar à função mostrada na figura 3.1 um valor constante de 10 em todos os pontos da função. Isso será feito de duas formas, a primeira fazendo $f(x) = f(x)+10$ para todos os valores de x da função, a segunda forma é equivalente, mas realizar-se-á essa soma para todas as frequências da função e não para seus pontos diretamente. Se uma operação feita no domínio original gerasse um resultado semelhante à realizada no domínio da frequência, então essas duas formas dariam o mesmo resultado.

Para isso, dado a função da imagem 3.1 será feito o cálculo $f(x) = f(x) + 10$ gerando a função mostrada na imagem 3.2, que, como pode ser visto, possui um deslocamento no eixo Y. Agora é feita a mesma coisa no domínio da frequência, inicialmente é retirado as frequências da nossa função, o resultado é mostrado em 3.3, nessas frequências, somaremos 10 em todos os seus pontos, o resultado pode ser visto na figura 3.4, com um pequeno deslocamento no eixo Y. Obviamente, não é possível comparar os resultados até aqui, portanto é aplicado a transformada inversa na função gerada obtendo a função mostrada na figura 3.5. Comparando 3.2 com 3.5, que são os dois resultados gerados pelos dois métodos, pode-se perceber que as duas funções não são equivalentes, demonstrando a afirmação de que operações realizadas no domínio comprimido podem não gerar o mesmo efeito sobre os dados do que as mesmas operações realizadas no domínio original. Essa operação será corretamente realizada no domínio comprimido ainda nesse capítulo.

Tem-se, a partir de agora, um novo problema, caso se queira realizar operações nos dados de entrada modificando suas frequências, deve-se especificar uma nova forma de realizar isso. Para isso será mostrado nesse capítulo um conjunto de operações que

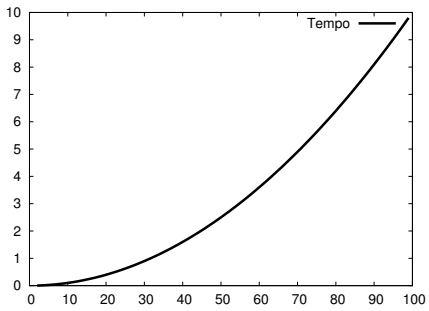

Figura 3.1: Função de entrada.

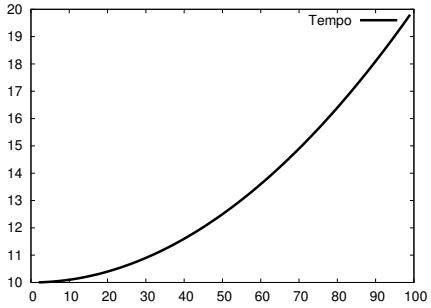

Figura 3.2: Função com todos os valores somados a 10.

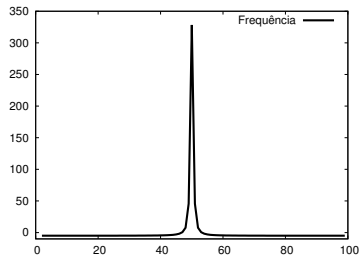

Figura 3.3: Frequências da Função.

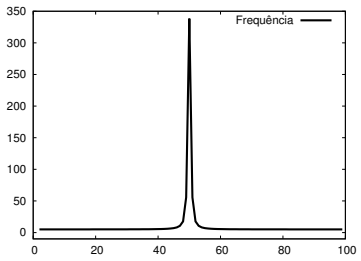

Figura 3.4: Frequências da função somada a 10.

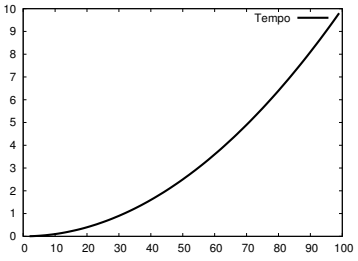

Figura 3.5: Função reconstruida com as frequências somadas a 10.

conseguem modificações nesses dados e que futuramente poderão compor algoritmos. Das 7 operações apresentadas duas são implicações diretas de teoremas conhecidos e uma foi baseada em um desses teoremas, todas as outras foram criadas objetivando sua utilização em exemplos futuros. Os teoremas usados podem ser encontrados em Bracewell et al (1965) e serão citados futuramente. Todos os testes desse capítulo foram feitos com a função F1 dada no capítulo 2.

3.1 Operações: Multiplicação por Escalar

Inicialmente, tratar-se-á da função de multiplicação. Objetiva-se, a partir de modificações nos coeficientes da Transformada de Fourier, multiplicar todos os pontos da função original por uma constante K.

A operação de multiplicação pode ser realizada utilizando uma ideia similar ao Teorema do Produto de Convolução que é mostrado na equação 3.2 e pode ser visto em detalhes em Bracewell et al (1965). Para uma análise da função, mostrar-se-á como ela é gerada.

Inicialmente será considerado que todos os pontos da função original já estão multiplicados pela constante K. Assim, tem-se a transformada dessa função dada pela equação 3.1:

$$F(n) = \sum_{x=0}^{L}(K.f(x)).cos\left(\frac{x.n.\pi}{L}\right) + \sum_{x=0}^{L} i.(K.f(x)).sen\left(\frac{x.n.\pi}{L}\right) \tag{3.1}$$

Sendo K constante, pode-se colocá-la em evidência e retirá-la do somatório:

$$F(n) = K.\left(\sum_{x=0}^{L} f(x).cos\left(\frac{x.n.\pi}{L}\right) + \sum_{x=0}^{L} i.f(x).sen\left(\frac{x.n.\pi}{L}\right)\right) \tag{3.2}$$

Portanto, deve-se apenas multiplicar todas as frequências por K, para fazer a multiplicação de K no domínio original.

Lembrando que, o principal objetivo é compactar grandes volumes de dados, é interessante conhecer o erro gerado pela multiplicação com valores diferentes de N, para saber até que ponto pode se ignorar informações sem ter perdas significativas. Para isso, é calculado a partir da função dada na figura 2.1 a diferença da função multiplicação feita no domínio original, fazendo $f(x) = f(x) * K$ e a operação de multiplicação apresentada pela equação 3.2, com N variando de 2 a 10800, o resultado pode ser visto na figura 3.6. Percebe-se que a função converge bem rápido para 0, a uma velocidade próxima do erro da própria Transformada.

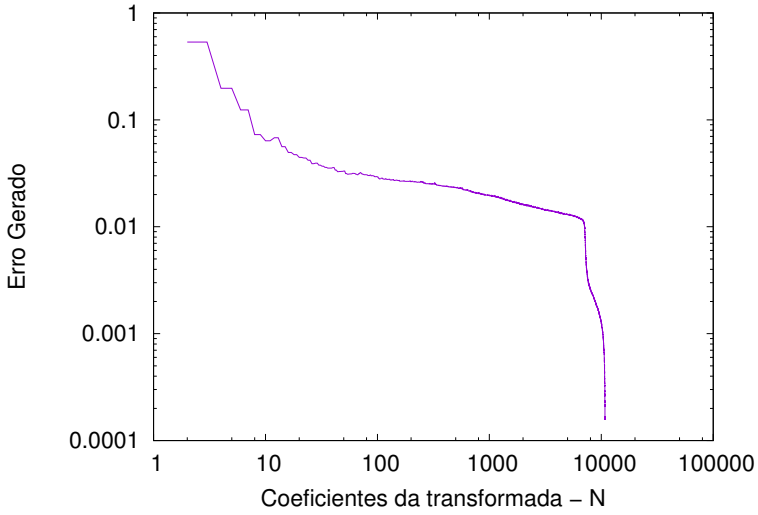

Figura 3.6: Erro relativo a função de multiplicação.

Naturalmente esse resultado encontrado para convergência da operação depende totalmente da função utilizada, como foi visto na seção anterior. A análise acima mostra

apenas uma tendência de como a convergência se comporta, e servirá como comparativo para futuras observações. Análises que estudem a convergência de data operação vinculada à função podem ser feitas futuramente.

É interessante perceber que, caso essa operação seja implementada em algoritmo, sua ordem de complexidade seria equivalente a ordem de complexidade da operação original, feita no seu domínio original, equivalente a O(n), onde a operação no domínio original seria O(L) onde L é o número de pontos, e no domínio comprimido complexidade O(N) onde N é o número de frequências. Não perdendo na complexidade, poder-se-ia obter uma melhoria temporal já que é possível reduzir o número de dados utilizados como mostrado acima.

3.2 Operações: Soma por Escalar

A próxima operação é a operação apresentada no início deste capítulo. Ela consiste da soma de um valor constante K sobre todos os pontos da função de entrada, mas, assim como na operação anterior, isso será feito modificando apenas os coeficientes da Transformada de Fourier. Para isso, utiliza-se uma característica da transformada Inversa. Dada à transformada Inversa de Fourier verifica-se como a frequência 0 se comporta.

$$f(x) = C(0).cos\left(\frac{x.0.\pi}{L}\right) + i.S(0).sen\left(\frac{x.0.\pi}{L}\right) \tag{3.3}$$

$$f(x) = C(0).1 + i.S(0).0 = C(0) \tag{3.4}$$

Portanto, a frequência 0 representa uma soma de $C(0)$ em todos os pontos. Assim, caso seja feito $C(0) \rightarrow (0) + K$ obtém-se:

$$f(x) = C(0) + K \tag{3.5}$$

A partir disso, é realizada uma soma de K em todos os pontos do domínio original. É interessante perceber que a operação de soma altera apenas uma frequência, portanto, ela não depende de N(número de frequências). Por esse motivo, a análise da convergência, como foi feita na seção anterior, é desnecessária, já que a operação apresentaria erro

constante para qualquer tamanho de N e uma complexidade computacional igual a O(1).

3.3 Operações: Deslocamento no eixo X

A última operação definida na seção anterior pode ser interpretada como um deslocamento da função no eixo Y. Quer-se agora uma aplicação semelhante, serão deslocados k pontos no eixo X a função e, como de costume, isso é feito apenas modificando os coeficientes gerados da Transformada de Fourier.

Para gerar essa operação se manipula a Transformada de Fourier. Será considerado que a função f(x) está no ponto l+k. Assim:

$$f(x) = \frac{1}{2\pi}\sum_{n=0}^{N} C(n).cos\left(\frac{(n).x.\pi}{L}\right) - i.S(n).sen\left(\frac{(n).x.\pi}{L}\right) \tag{3.6}$$

Procurando a equação do seno e cosseno tem-se as equações abaixo:

$$C(n) = \frac{1}{2\pi}\sum_{l=0}^{L} f(l).cos\left(\frac{l.n.\pi}{L} + \frac{k.n.\pi}{L}\right) \tag{3.7}$$

$$S(n) = \frac{1}{2\pi}\sum_{l=0}^{L} -i.f(l).sen\left(\frac{l.n.\pi}{L} + \frac{k.n.\pi}{L}\right) \tag{3.8}$$

Usando as regras de soma de seno e cosseno temos:

$$C(n) = \frac{1}{2\pi}\sum_{l=0}^{L} f(l).cos\left(\frac{l.n.\pi}{L}\right).cos\left(\frac{k.n.\pi}{L}\right) - f(l).sen\left(\frac{l.n.\pi}{L}\right).sen\left(\frac{k.n.\pi}{L}\right) \tag{3.9}$$

$$S(n) = \frac{1}{2\pi}\sum_{l=0}^{L} f(l).sen\left(\frac{l.n.\pi}{L}\right).cos\left(\frac{k.n.\pi}{L}\right) + f(l).cos\left(\frac{l.n.\pi}{L}\right).sen\left(\frac{k.n.\pi}{L}\right) \tag{3.10}$$

Que pode ser resumido em:

$$C(n) = C(n).cos\left(\frac{k.n.\pi}{L}\right) - S(n).sen\left(\frac{k.n.\pi}{L}\right) \tag{3.11}$$

$$S(n) = S(n).cos\left(\frac{k.n.\pi}{L}\right) + C(n).sen\left(\frac{k.n.\pi}{L}\right) \tag{3.12}$$

Ou seja, o deslocamento da função no eixo x pode ser feito com uma atualização direta dos coeficientes gerando um custo computacional de n, um custo equivalente à mesma operação feita no domínio original. Note-se que, para cada C(x) usou-se S(x) e que para cada S(x) usou-se C(x), o que não é um grande problema já que se têm esses valores.

3.4 Operações: Soma de Funções

Agora será analisado como é possível realizar a soma dos pontos de várias funções utilizando o apoio da Transformada de Fourier. Em linhas gerais, se quer calcular o valor de f(x) para todo x onde $f(x) = (f_1(x) + f_2(x) + ... + f_n(x))$.

Pode-se, assim como foi feito nas seções anteriores, encontrar meios de substituir as operações presentes na fórmula e, assim, realizá-la totalmente no domínio comprimido gerando o mesmo resultado da fórmula realizada no seu domínio original de pontos. Porém, para uma análise diferente da aplicação, é realizado essa operação no domínio da frequência na forma literal que ela se apresenta e uma análise dos resultados gerados por ela será feita. Portanto, será analisado o resultado de $g(x) = (g_1(x) + g_2(x) + ... + g_n(x))$ onde $g_m(x)$ é o coeficiente x da Transformada de Fourier sobre f_m.

A partir de operações sobre a Transformada Inversa de Fourier, veremos que os resultados das duas expressões são iguais. Considerando a transformada inversa abaixo:

$$f(x) = \sum_{n=0}^{N} C(n).cos\left(\frac{x.n.\pi}{L}\right) + i.S(n).sen\left(\frac{x.n.\pi}{L}\right) \tag{3.13}$$

Fazendo para o somatório tem-se:

$$f_1(x)+...+f_m(x) = \left(\sum_{n=0}^{N} C_1(n).cos\left(\frac{x.n.\pi}{L}\right) + i.S_1(n).sen\left(\frac{x.n.\pi}{L}\right) + ... + \sum_{n=0}^{N} C_m(n).cos\left(\frac{x.n.\pi}{L}\right) + i.S_m(n).sen\left(\frac{x.n.\pi}{L}\right)\right) \tag{3.14}$$

Juntando o somatório:

$$f_1(x)+...+f_m(x) = \sum_{n=0}^{N} \left(C_1(n).cos\left(\frac{x.n.\pi}{L}\right) + i.S_1(n).sen\left(\frac{x.n.\pi}{L}\right) + ... + C_m(n).cos\left(\frac{x.n.\pi}{L}\right) + i.S_m(n).sen\left(\frac{x.n.\pi}{L}\right)\right) \tag{3.15}$$

Agora pode-se colocar a parte constante em evidência:

$$f_1(x)+...+f_m(x)=\sum_{n=0}^{N}(C_1(n)+...+C_m(n)).cos\left(\frac{x.n.\pi}{L}\right)+i.(S_1(n)+...+S_m(n)).sen\left(\frac{x.n.\pi}{L}\right) \quad (3.16)$$

Portanto, pode-se afirmar que realizar a soma dos coeficientes, gera na Transformada inversa, um resultado semelhante à soma dos pontos das funções no domínio original da função.

Assim como na operação de multiplicação, a operação apresentada não gera maiores complexidades temporais para a função, mantendo sua complexidade na ordem de O(L.m) onde L é o número de coeficientes e m o número de funções. Portanto, a melhoria no processamento dessa operação também poderia ocorrer numa menor utilização de dados. A convergência dessa operação, ao contrário da multiplicação, depende mais fortemente das funções dadas, pois ela apresenta um conjunto de funções sendo utilizado. Nesse contexto, optou-se por não mostrar a convergência já que ela poderia concluir erros a aplicação. A convergência deve ser estudada após se definir o problema.

3.5 Operações: Multiplicação de Funções

O objetivo agora é multiplicar duas funções: dado f(x) e g(x) se quer encontrar I(x) que seja, para todo x, I(x)=f(x).g(x). Para isso, pode ser usado o Produto de Convolução. O produto de convolução definido por F(x)*g(x) nos da um resultado equivalente a f(x).g(x) sendo F a função correspondente aos coeficientes da Transformada de Fourier de f.

Um estudo detalhado da convolução pode ser visto em Bracewell et al (1965), mas para uma compreensão rápida dessa operação e de como podemos realiza-la, mostraremos uma geração dela agora. Considere a Transformada Inversa dada por 3.17:

$$f(x)=\sum_{n=0}^{N}C(n).cos\left(\frac{x.n.\pi}{L}\right)+i.S(n).sen\left(\frac{x.n.\pi}{L}\right) \quad (3.17)$$

Deseja-se saber quanto é f(x).g(x). Portanto fazendo:

$$f(x).g(x)=\left(\sum_{n=0}^{N}C(n).cos\left(\frac{x.n.\pi}{L}\right)+i.S(n).sen\left(\frac{x.n.\pi}{L}\right)\right).g(x) \quad (3.18)$$

$$f(x).g(x) = \sum_{n=0}^{N} C(n).cos\left(\frac{x.n.\pi}{L}\right).g(x) + i.S(n).sen\left(\frac{x.n.\pi}{L}\right).g(x) \tag{3.19}$$

$$f(x).g(x) = \sum_{n=0}^{N} (C(n).g(x)).cos\left(\frac{x.n.\pi}{L}\right) + i.(S(n).g(x)).sen\left(\frac{x.n.\pi}{L}\right) \tag{3.20}$$

Logo, de acordo com a equação 3.20, para obter f(x).g(x). Deve-se multiplicar cada C(n) e S(n) por g(x).

Um fato importante sobre esse resultado é que o produto de convolução, se executado computacionalmente, teria ordem de complexidade de $O(L.N)$, onde se teria uma multiplicação entre cada frequência com cada ponto da função. Porém, dependendo da aplicação que utilizará essa operação, o produto de convolução pode, como mostrado na fórmula 3.20, ser acoplado a própria transformada inversa de Fourier sem trazer novos custos ao problema, já que a transformada inversa, é um procedimento esperado de acontecer em uma aplicação que use Transformada de Fourier.

3.6 Operações: Somatório

A próxima operação é o somatório, quer-se conhecer qual é a soma de todos os pontos de uma função utilizando apenas os coeficientes da transformada. Para essa função pode-se implementar uma equação gerada a partir de um teorema da Transformada de Fourier. O Teorema de Parseval Bracewell et al (1965) nós dá uma relação entre o somatório dos pontos da função original e o somatório das suas frequências definido pela equação abaixo:

$$\sum_{x=0}^{L} |f(x)|^2 = \frac{1}{N} \sum_{n=0}^{N} |F(n)|^2 \tag{3.21}$$

Sendo $f(x)$ a função original e $F(n)$ os coeficientes de Fourier. Portanto, tem-se apenas que usar a primeira parte da igualdade para o conjunto de tempos e a segunda parte da igualdade para as frequências.

O que se quer agora, depois de demonstrada a validade da operação é saber como ela se comporta para valores diferentes de N, para novamente conhecer seus possíveis índices de compactação. Para tal, foi utilizada a mesma análise feita para a multiplicação por escalar, foi calculado a variação que a equação 3.21 gera na função do gráfico 2.1

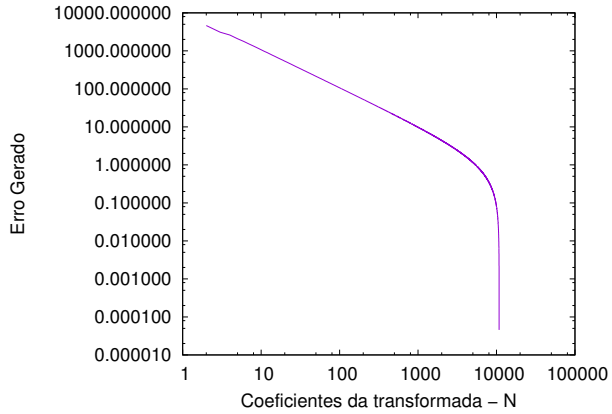

Figura 3.7: Erro relativo a função Somatório 1.

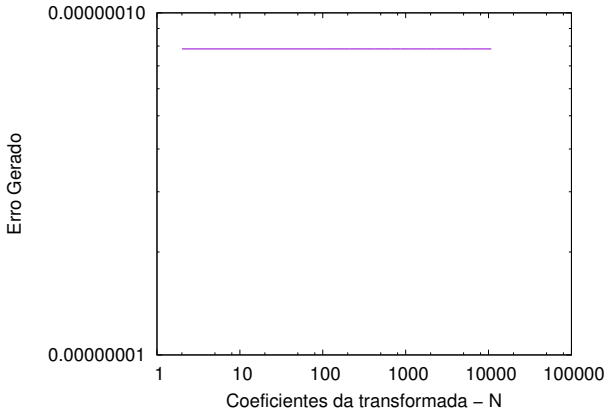

Figura 3.8: Erro relativo a função Somatório 2.

quando é alterado o valor de N para as frequências. A figura 3.7 apresenta o resultado.

Com a figura 3.7, pode-se perceber que o erro gerado é muito grande para valores pequenos de N e ainda, que a convergência da função é muito lenta. A fim de melhorar esse resultado de erro, será apresentada outra forma de calcular o somatório. Para isso, se verifica qual a soma que uma única frequência gera, esse valor pode ser encontrado a partir da equação 3.23. A equação 3.23 representa a Transformada Discreta Inversa de Fourier modificada, onde cada $f_2(n)$ representa o somatório de cada frequência n que compõe cada ponto $f(n)$ da função original. $C(n)$ e $S(n)$ representam respectivamente a parcela do cosseno e seno gerados pela Transformada de Fourier. A função 3.22 representa o somatório final gerado com a soma de cada parcela de frequência.

$$E_2 = \sum_{n=0}^{N} f_2(n) \tag{3.22}$$

$$f_2(n) = \sum_{x=0}^{N} C(n).cos\left(\frac{n.x.\pi}{L}\right) + i.S(n).sen\left(\frac{n.x.\pi}{L}\right) \tag{3.23}$$

Veja que a única diferença com a formula da transformada original é que, nesse caso, tem-se o somatório em função dos coeficientes (x) e, C(n) e S(n) em função dos pontos da função (n). Será Mostrado que essa função é equivalente a soma de todos os pontos gerados pela Transformada Inversa de Fourier que é dada na equação 3.24

$$E = \sum_{n=0}^{N} f(n) = \sum_{n=0}^{N}\sum_{x=0}^{N} C(x).cos\left(\frac{n.x.\pi}{L}\right) + i.S(x).sen\left(\frac{n.x.\pi}{L}\right) \tag{3.24}$$

Para isso, é substituída a equação 3.23 em 3.22 obtendo 3.25 e invertendo os somatórios chega-se a 3.26 que é equivalente a 3.24

$$E_2 = \sum_{n=0}^{N} f_2(n) = \sum_{n=0}^{N}\sum_{x=0}^{N} C(n).cos\left(\frac{n.x.\pi}{L}\right) + i.S(n).sen\left(\frac{n.x.\pi}{L}\right) \tag{3.25}$$

$$E_2 = \sum_{x=0}^{N}\sum_{n=0}^{N} C(n).cos\left(\frac{n.x.\pi}{L}\right) + i.S(n).sen\left(\frac{n.x.\pi}{L}\right) \tag{3.26}$$

Portanto, para ter o somatório final, pode ser utilizada a função 3.22 que possui um custo computacional de $O(N.L)$(número de frequências*número de pontos). Seria interessante uma operação que não dependesse do número de pontos da função para assim obter um tempo de processamento constante e irrelevante ao tamanho dos nossos dados. Para gerar uma função com custo O(N), começa-se separando e tirando do somatório $C(n)$ e $S(n)$.

$$f_2(n) = C(n).\sum_{x=0}^{N} cos\left(\frac{n.x.\pi}{L}\right) + S(n).\sum_{x=0}^{N} i.sen\left(\frac{n.x.\pi}{L}\right) \tag{3.27}$$

E é guardado o resultado do somatório em um vetor:

$$f_2(n) = C(n).C'(n) + S(n).S'(n) \tag{3.28}$$

$$C'(n) = \sum_{n=0}^{N} cos\left(\frac{n.x.\pi}{L}\right) \tag{3.29}$$

$$S'(n) = \sum_{x=0}^{N} sen\left(\frac{n.x.\pi}{L}\right) \tag{3.30}$$

$C'(n)$ e $S'(n)$ são constantes e irrelevantes a função original, são derivadas diretamente da Transformada Inversa. O somatório final se torna:

$$E = \sum_{n=0}^{N} C(n).C'(n) + S(n).S'(n) \tag{3.31}$$

Gerado esse novo resultado, será analisado a sua convergência como foi feito para a primeira proposta do somatório, queremos que ela obtenha um resultado melhor do que a primeira proposta afim de obter melhorias em relação a velocidade de processamento. Como pode ser visto na figura 3.8 o erro gerado pela nova expressão se mantém perto de

0 para todos os valores de N. Portanto, a proposta conseguiu um resultado excelente em relação ao resultado da literatura em razão à convergência, mantendo a mesma complexidade temporal. Isso gera um novo método que conseguiria resultados semelhantes com um número menor de coeficientes (N), aumentando a possibilidade de compactação dos dados.

3.7 Operações: Get(x) e Set(x)

Será mostrada a operação Get(x), a função que, dados os coeficientes gerados da Transformada de Fourier retorna um único valor na posição x da função original. Utilizando a mesma lógica é definida a função Set(x) que soma um valor ao ponto x da função.

Fica explícito na função 2.2 que cada coeficiente F(x) da transformada é definido por um somatório de todos os pontos da função. Isso implica que para definir um coeficiente da Transformada de Fourier com total exatidão é necessário percorrer todos os pontos da função.

O inverso também é visto na função 2.5. Onde, para definir um ponto da nossa função original, novamente com total exatidão, é preciso percorrer todos os N coeficientes gerados pela transformada.

Por fim, Get(x) é definido pela implementação direta da função 3.32

$$Get(f(x)) = \sum_{n=0}^{N} C(n).cos\left(\frac{x.n.\pi}{L}\right) + i.S(n).sen\left(\frac{x.n.\pi}{L}\right) \tag{3.32}$$

Conclui-se que a função Get(x) pode ser realizada com a utilização direta da Transformada Inversa de Fourier. Considerando que a função Set(x) possui a mesma lógica funcional, se tem que ela pode ser facilmente realizada com uma implementação direta da Transformada de Fourier.

A análise da convergência da função Get e Set se assemelha a toda à análise feita para a convergência da Transformada de Fourier. Além de a fórmula ser equivalente ela depende totalmente de um único ponto da função que pode possuir propriedades bem diferentes de todos os outros pontos da mesma função tornando sua convergência complexa de ser definida.

Perceba que, a complexidade dessas operações na forma como foram apresentadas é de O(N). Esse é possivelmente o pior resultado que pode-se apresentar com o uso da transformada, já que a mesma operação poderia ser facilmente feita em tempo O(1) no domínio original. De fato, por características da transformada, funções pontuais como o Get e o Set, que alterariam apenas um ponto da função, não são interessantes de serem feitas nesse domínio, o que apresenta um grande desafio para qualquer algoritmo.

3.8 Resultados: Média de Funções

O primeiro resultado, que utiliza as operações vistas acima, é uma operação bem simples e por isso, ela é apresentada nesse capítulo, pois pode ser vista como uma nova operação. Será definida uma função f(x) que seja a média de n funções $f_m(x)$ com $n, m \in \mathbb{N}$ e $0 \leq m \leq n$, definidas como $f(x) = (f_1(x) + f_2(x) + ... + f_n(x))/n$ para cada x.

Perceba que essa função pode ser dividida em duas operações, um somatório de n funções mais uma divisão por n ou uma multiplicação por $1/n$. Formalmente, essa função poderia ser escrita como $f(x) = s(x) * m(x)$ onde $s(x) = f_1(x) + f_2(x) + ... + f_n(x)$ e $m(x) = 1/n$.

Analisando cada termo da fórmula, tem-se que o resultado de s(x) é um somatório de funções que foi estudado na seção 3.4 e pode ser realizado totalmente no domínio comprimido. Já m(x) é uma multiplicação da função por um escalar, que também foi estudada nesse trabalho e pode ser visto na seção 3.1.

O resultado final da operação pode ser visto na fórmula 3.33, onde para gerar o resultado de $f(x) = (f_1(x) + f_2(x) + ... + f_n(x))/n$ tem-se que manipular os coeficientes da Transformada como especificado nas fórmulas 3.34 e 3.35.

$$\frac{f_1(n) + ... + f_m(n)}{m} = \sum_{x=0}^{N} \frac{(C_1(x) + ... + C_m(x))}{m} . cos\left(\frac{n.x.\pi}{L}\right) + i. \frac{(S_1(x) + ... + S_m(x))}{m} . sen\left(\frac{n.x.\pi}{L}\right) \tag{3.33}$$

$$C(x) = \frac{(C_1(x) + C_2(x) + ... + C_m(x))}{m} \tag{3.34}$$

$$S(x) = \frac{(S_1(x) + S_2(x) + ... + S_m(x))}{m} \tag{3.35}$$

Conclui-se então que, a média de várias funções poderia ser totalmente realizada no domínio comprimido podendo, com isso, ter ganhos na quantidade de dados utilizados. Ainda, não são geradas novas perdas na complexidade de tempo de processamento total já que as duas operações apresentadas para realizar a média podem ser implementadas na mesma ordem de complexidade que suas funções originais.

3.9 Resultados: Média de Pontos

O próximo resultado é também uma operação bem simples que se utiliza das operações vista neste capítulo. É estudada a aplicação de uma média, porém, ao contrário da média anterior, quer-se gerar um valor M que seja a média de todos os pontos de uma função. A fórmula da expressão pode ser vista em 3.36.

$$M = \frac{\sum_{x=0}^{N} f(x)}{N} \tag{3.36}$$

Assim como feito anteriormente, essa expressão pode também ser dividida em duas funções $M = s(x) * m(x)$ onde $s(x)$ é dado pelo somatório dos N $f(x)$ e $m(x)$ é dado pela constante $1/N$.

Dividido M em duas expressões $s(x)$ e $m(x)$ temos que $s(x)$ é o somatório dos pontos da função $f(x)$ que foi explicitado e analisado na seção 3.6 e pode ser realizada totalmente no domínio comprimido. Igualmente à seção anterior, $m(x)$ é uma multiplicação por uma constante que também pode ser realizada totalmente no domínio comprimido como foi visto na seção 3.1.

A conclusão se assemelha a média anterior, tem-se uma nova expressão que pode ser composta por expressões que podem ser realizadas totalmente no domínio comprimido e, ainda, a utilização dessas operações não traz maiores custos a complexidade computacional da expressão final, podendo ter um ganho na velocidade de processamento ao se utilizar um conjunto menor de dados.

3.10 Conclusão

A conclusão que se pode tirar desse capítulo é que a Transformada de Fourier pode ser utilizada para a composição de várias operações que podem servir de base para vários algoritmos futuros. Foi produzido um conjunto de 8 operações e 2 resultados. O grande ganho deste capítulo pode ser visto na tabela 3.10 onde a complexidade das operações feitas no domínio original são comparadas com a sua complexidade no domínio comprimido. Veja que apenas 4 casos apresentaram mudanças, 3 de piora com um caimento de O(L) para O(L.N) e O(1) para O(N) e, um caso de melhora, indo de O(L) para O(1). Porém, a manutenção da complexidade, nesse caso, pode apresentar uma melhoria na velocidade final do algoritmo, já que, os algoritmos apresentados no domínio comprimido trabalham em cima de um domínio de dados menor, compactado, ou seja $N \leq L$.

Algoritmos	Domínio Original	Domínio Comprimido
Multiplicação por Escalar	O(L)	O(N)
Soma por Escalar	O(L)	O(1)
Deslocamento no Eixo X	O(L)	O(N)
Soma de Funções	O(L)	O(N)
Multiplicação de Funções	O(L)	O(L.N)
Somatório	O(L)	O(N)
Get	O(1)	O(N)
Set	O(1)	O(N)
Média de Funções	O(L.M)	O(N.M)
Média de Pontos	O(L)	O(N)

Tabela 3.1: Complexidade das operações apresentadas para o domínio original e comprimido. Onde L é o número de pontos, N o número de coeficientes e M o número de funções utilizadas na média.

4 Algoritmos de Previsão

Até agora foi mostrada uma análise sobre a Transformada de Fourier, foi analisada como ela se comporta para diferentes funções e evidenciadas propriedades das funções que podem ser utilizadas como indicativo para a eficiência da sua utilização com a transformada. A seguir é estabelecido um conjunto de operações que mostram como se pode utilizar a transformada para resolver pequenos problemas que podem compor problemas mais complexos. Agora se quer testar o que foi realizado até agora, são apresentados problemas recorrentes da computação e mostrado como a Transformada, utilizando o que foi visto até aqui, pode ajudar nesses problemas.

Especificamente, serão analisados problemas de previsão. Morettin et al (1981) diz que, etimologicamente, a palavra previsão sugere que se quer conhecer algo antes que ela exista, o termo indica algo que pode acontecer no futuro. Matematicamente pode-se supor que existe uma série temporal conhecida até o instante t e deseja-se saber o valor dessa série no instante $t + h$. Basicamente os procedimentos para isso conduzem à uma equação gerada a partir dos pontos conhecidos da série temporal (Morettin et al, 1981).

Definido o problema, serão utilizados dois algoritmos mostrados em Morettin et al (1981) que já são muito conhecidos. Será mostrado de forma rápida seu funcionamento e depois serão reescritos utilizando a Transformada de Fourier. Os algoritmos utilizados serão a Média Móvel Simples (MMS) e o Alisamento Exponencial Simples (AES).

4.1 Médias Móveis Simples (MMS)

A Média Móvel Simples (MMS) consiste em calcular os L pontos conhecidos mais recentes da função, quer-se encontrar M tal que:

$$M = \frac{x_i + x_{i+1} + x_{i+2} + ... + x_{i+L}}{L} \tag{4.1}$$

Objetiva-se, como foi falado acima, reconstruir essa função utilizando a Transformada de Fourier e obtendo possíveis ganhos na velocidade total do algoritmo a partir de

um ganho com a diminuição da quantidade de dados totais utilizados.

Inicialmente pode-se considerar que temos um vetor de L números que precisa ser transformado para o domínio comprimido. A transformada utilizada possui uma complexidade total de O(LlogL). Após os L números transformados tem-se que realizar uma média simples para obter o valor de M, porém, a média já foi estudada e implementada na seção 3.9.

Conclui-se até agora que o primeiro algoritmo de previsão apresentado a Médias Móveis Simples(MMS) conseguiu, além do tempo gasto com a Transformada de Fourier, um tempo total de O(N) que seria a mesma complexidade que sua utilização no domínio original considerando L igual a N. Podendo ter um ganho com a quantidade de dados utilizados onde N seria menor que L.

Uma complicação da solução pode ocorrer quando se quer prever um valor diferente de x_{i+L+1}. Para obter, por exemplo, x_{i+L+2} em geral esses algoritmos utilizam um novo conjunto de dados que vai de i a $i+L+1$ ou de $i+1$ a $i+L+1$ utilizando o valor encontrado ou o valor real recém descoberto. Porém, a concatenação de um novo valor a uma função não foi apresentada e talvez não possua ordem de complexidade N. Para resolver esse problema mostraremos duas formas que podem ser utilizadas.

Inicialmente pode-se utilizar a solução mais óbvia que é realizar direto a transformada sobre o novo conjunto de pontos. O problema dessa solução é que, para cada novo ponto a ser previsto, será necessário um custo de O(NlogN) para organizar os dados. Ou seja, apesar de simples a solução, ela pode gerar um custo computacional alto.

A segunda solução envolve a utilização de duas operações vistas no capítulo anterior. Inicialmente é feito a operação de deslocamento mostrado na seção 3.3 com $k = -1$. Como o deslocamento apresentado é circular temos que na função original f, $f(L) := f(1)$, considerando que objetiva-se um vetor que represente os $i+1$ a $i+L+1$ pontos da função, tudo que tem-se que fazer é $f(L) = f(L+1)$ para isso pode-se utilizar a função $Set(x)$ apresentada na seção 3.7 para subtrair o valor de $f(1)$ e somar o valor de $f(L+1)$.

Perceba que o segundo método apresentado, mesmo utilizando duas funções, apresenta um tempo total de $O(L)$. Portanto, pode-se considerar que a atualização do vetor para a previsão de vários pontos da função não traz maiores custos a complexidade original

que a própria função de previsão possui.

4.2 Alisamento Exponencial Simples(AES)

O Alisamento Exponencial Simples pode ser visto como uma melhoria do método de média Móvel Simples. Todas as funções explicadas na seção anterior irão compor esse novo algoritmo. Em linhas gerais seu modelo matemático é expresso abaixo:

$$M = ax_{i+L} + a(1-a)x_{i+L-1} + a(1-a)^2 x_{i+L-2} + ... + a(1-a)^L x_i \tag{4.2}$$

Sendo a uma constante de alisamento onde $0 < a < 1$. Para gerar esse algoritmo precisa-se também de duas expressões, inicialmente uma semelhante a do algoritmo anterior, onde se tem que realizar um somatório dos L termos de uma função, essa expressão pode ser feita facilmente utilizando a expressão apresentada na seção 3.6. Porém, o somatório necessário aqui é um somatório ponderado por uma expressão que utiliza a como constante, mas, essa expressão pode também ser realizada com as expressões já mostradas nesse trabalho.

Para fazer a ponderação da função é inicialmente criada uma nova função k(n) definida no domínio de [0,L] e dada abaixo:

$$k(n) = a(1-a)^{n-1} \tag{4.3}$$

Portanto, sendo f(n) a função original, tem-se apenas que fazer f(n).k(n) e assim obter a equação dada por 4.2. Essa nova expressão pode ser realizada com a operação apresentada na seção 3.5 pelo produto de convolução.

A aplicação do produto de convolução ao método traz um problema a ela. Como explicado na seção 3.5, o produto entre funções possui custo computacional na ordem de O(N.L) ou de O(LlogN) caso seja incorporado a transformada. Porém, em ambos os casos, temos um custo maior em relação ao algoritmo original.

Usando o produto de funções acoplado a transformada, tem-se o custo de fazer a transformada em cada novo ponto a ser previsto. Usando o produto diretamente nas

funções pode-se utilizar o deslocamento da função mostrado na seção anterior, mas mesmo assim teríamos um custo mais elevado.

Esse resultado gera um impasse, utilizando as operações apresentadas se aumenta a ordem de complexidade total da função, o que em geral, aumenta muito o tempo de processamento. Por outro lado, utilizando a transformada pode-se, com a compactação dos dados, ter um gasto menor de espaço e uma melhora no tempo já que se pode reduzir a utilização de dados. Portanto, a eficiência desse método está ligada ao quanto se pode compactar os dados tendo uma perda razoável e assim nivelar os prós e contras da sua utilização.

4.3 Conjunto de Dados

A fim de experimentar a eficiência dos algoritmos mostrados acima e da sua aplicação com a Transformada de Fourier, serão apresentadas três séries temporais que serão usadas para a realização dos testes. Cada série temporal possui propriedades importantes que permitirão visualizar características tanto do algoritmo quanto da Transformada de Fourier.

A primeira série temporal que será utilizada pode ser encontrada em Morettin et al (1981) é chamada de Série do Leite e representa a produção de leite no estado de São Paulo(em milhões de litros) retirada do Instituto de Economia Agrícola(IEA) nos períodos de dezembro de 1975 a novembro de 1980. O gráfico da série pode ser visto na Figura 4.1.

A segunda série utilizada foi retirada do repositório de séries estatísticas do IBGE. É apresentado o custo e índices da construção civil por mês, são investigados a partir do levantamento de preços de materiais e salários pagos na construção civil, para o setor habitação pelo Sistema Nacional de Pesquisa de Custos e Índices da Construção Civil IBGE (2016). A série é definida no período de Julho de 1994 até Janeiro de 2016. O gráfico da série pode ser visto na Figura 4.2.

A terceira série utilizada foi retirada do mesmo repositório que a série anterior. O IBGE (2016) define a série como o Sistema Nacional de Pesquisa de Custos e Índices da Construção Civil efetua a produção de custos e índices da construção civil, a partir

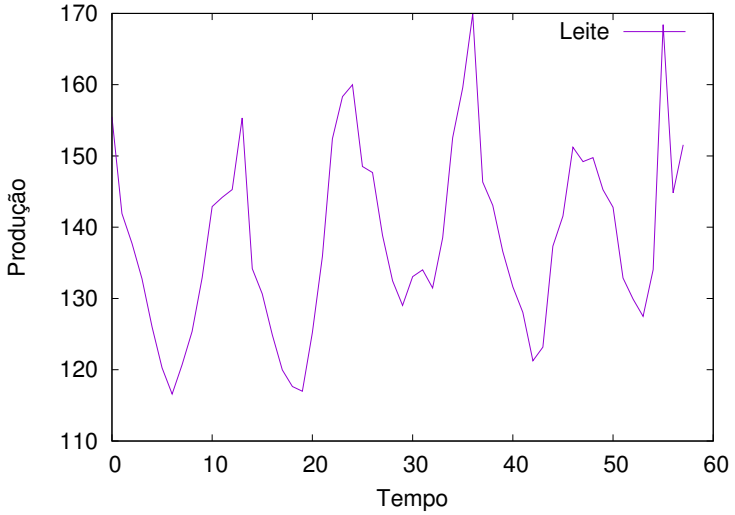

Figura 4.1: Produção de leite no estado de São Paulo(em milhões de litros) por mês. Morettin et al (1981) Instituto de Economia Agrícola(IEA)

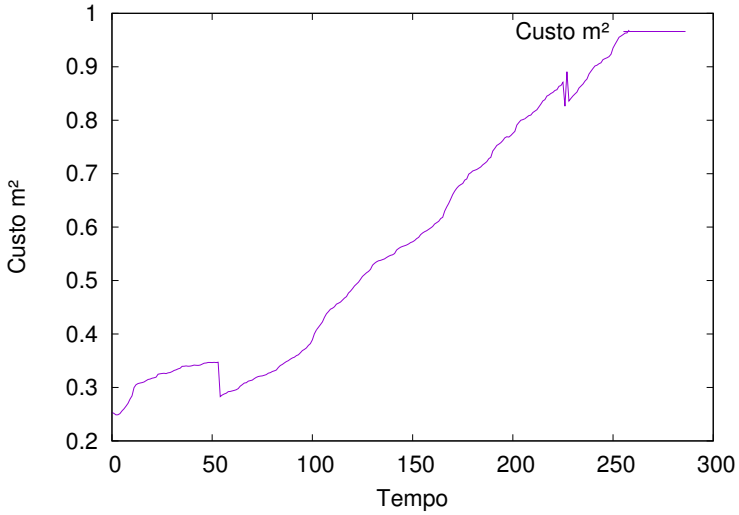

Figura 4.2: Custos e índices da construção civil por mês. IBGE (2016) Repositório de séries estatísticas do IBGE

do levantamento de preços de materiais e salários pagos na construção civil, para o setor habitação. Também é uma série mensal definida nos períodos de Julho de 1994 a Janeiro de 2016. O gráfico da série pode ser visto em 4.3.

Para uma referência mais fácil às séries, elas serão chamadas de S1, S2 e S3 respectivamente a ordem que foram apresentadas.

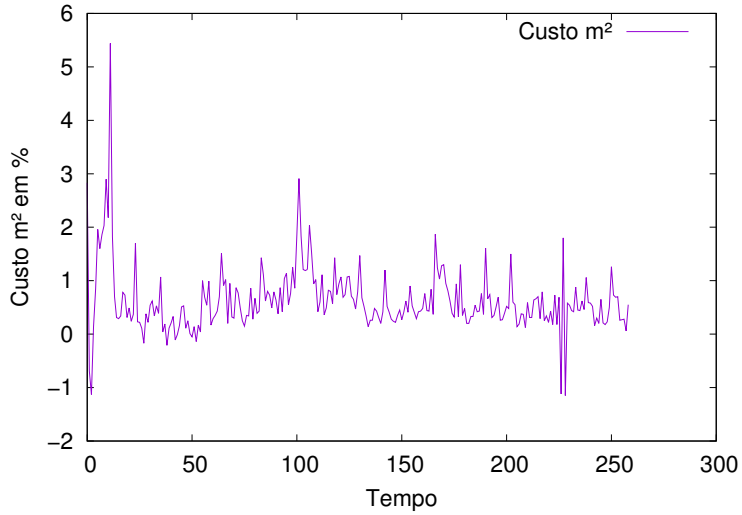

Figura 4.3: Custo médio m em moeda corrente - variação (% no mês). IBGE (2016) Custos e Índices da Construção Civil

4.4 Testes e Resultados

Com os dois algoritmos mostrados e com uma forma de reconstrui-los utilizando a Transformada de Fourier e com três séries temporais definidas, pode-se agora testar o funcionamento dos algoritmos. Para isso, o algoritmo MMS e o algoritmo AES serão aplicados as três séries apresentadas(S1, S2 e S3). Inicialmente é estudada a aplicação do método MMS nas três séries, posteriormente é testado o método AES.

Para a realização dos testes será escolhido um intervalo L que define o número de pontos que será usado para realizar a previsão. O algoritmo de previsão terá L pontos conhecidos e tentará prever o ponto L+1. Chamando X de L+1, a cada iteração incrementa-se X e, para prever X+1, usa-se os L pontos anteriores a X. Termina-se o algoritmo quando X for o último ponto da função.

4.4.1 MMS: Série S1

Inicialmente será testada a técnica MMS utilizada na série S1, o esquema para teste citado acima será utilizado. A partir desse ponto algumas características interessantes tanto do algoritmo quanto da série poderão ser vistas. Alguns resultados são mostrados na Figura 4.4. Veja que, como diz o método, ate os L pontos do gráfico a previsão é perfeita, isso ocorre porque esses pontos são considerados como conhecidos e deseja-se prever apenas os pontos alem destes.

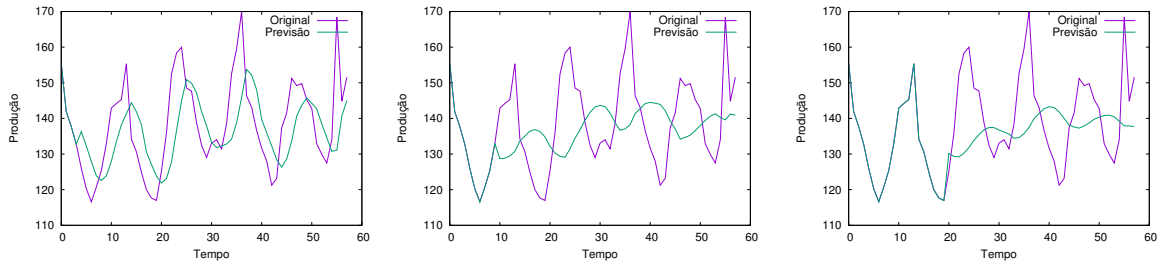

Figura 4.4: Algoritmo MMS utilizado na série S1 para L=4, L=10 e L=20 respectivamente.

Pode-se perceber que a série possui uma alta variação, mesmo apresentando uma periodicidade com picos de aproximadamente 12 em 12 meses, o algoritmo não é capaz de captar isso. Além disso, o algoritmo apresentado utiliza principalmente a média entre os pontos. A média possui uma natural característica de suavizar a função. Explicando com um exemplo simples, considere dois pontos A e B com A>B, a média entre A e B chamada de C necessariamente estará no meio dos dois pontos, ou seja, A>C>B. A característica de suavização pode ser vista nos exemplos gerados, é visto que aumentando o L(o número de pontos usados na média) a resposta gerada perde seus picos e se torna mais suave.

O objetivo com os testes, além de apresentar características dos algoritmos e das séries, é tentar utilizar a Transformada de Fourier para reconstruir o algoritmo utilizando um número menor de dados porém, neste caso, tem-se um empecilho. Considerando que a melhor resposta utiliza apenas 4 pontos no algoritmo, a compactação que poderia ser utilizada, seria muito pequena e pouco útil já que o algoritmo, utilizando apenas 4 pontos, já seria muito rápido. As aplicabilidades da transformada assim como suas características poderão ser melhores vistas nos exemplos seguintes.

4.4.2 MMS: Série S2

Será testado agora a série S2. Ao contrário da série S1, a nova série apresenta pequenas variações no seu decorrer, tendo apenas um significativo aumento que perdura por quase todo seu domínio. Testes semelhantes aos da série S1 foram feitos e são apresentados na Imagem 4.5. Perceba que foi utilizado valores de N maiores exatamente pela baixa variação da função.

A característica de suavização, marcante no resultado da série S1, também é visto

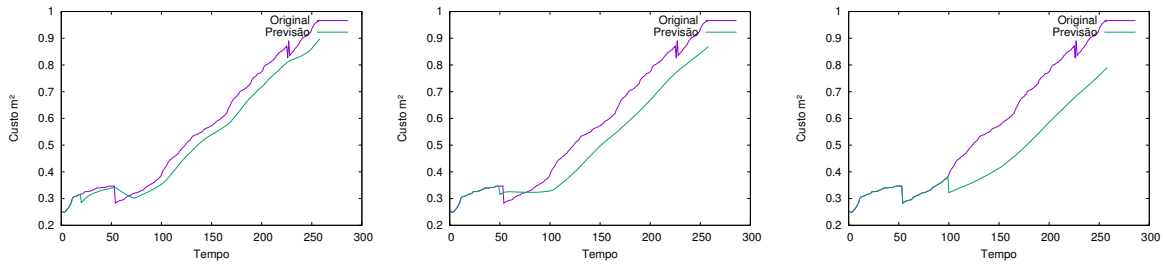

Figura 4.5: Algoritmo MMS utilizado na série S2 para L=20, L=50 e L=100 respectivamente.

aqui, mas com menos intensidade porém, é possível perceber que com L=100 tem-se uma curva mais suavizada que com L=20. Uma outra característica que pode ser vista é um deslocamento no eixo Y com o aumento de L. Isso se deve também a característica da média e nessa caso ao crescimento continuo da função.

Para esse caso pode-se considerar que o melhor caso é apresentado na função com L=20. Apensar de ainda, nesse caso, não poder-se considera-lo Big Data, uma análise da transformada já é possível. O resultado pode ser visto na imagem 4.6.

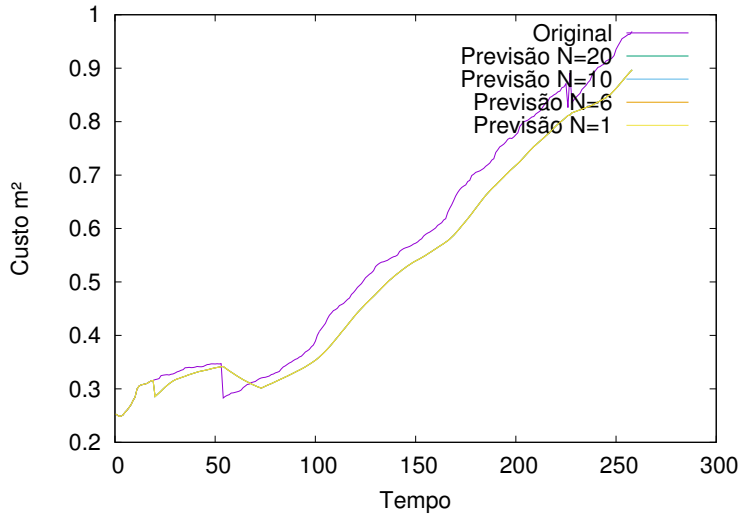

Figura 4.6: Algoritmo MMS utilizado na série S2 para L=20 e variando o número de frequências N.

O resultado mostrado na imagem pode parecer inicialmente estranho, mas apenas revela o poder da Transformada de Fourier. O resultado mostra que a utilização da Transformada com apenas um coeficiente não traz nenhum erro visível à resposta. Dentre os possíveis motivos desse resultado podemos considerar a baixa derivada da função, que como já visto nesse trabalho pode influenciar a geração de erros e, o pequeno domínio

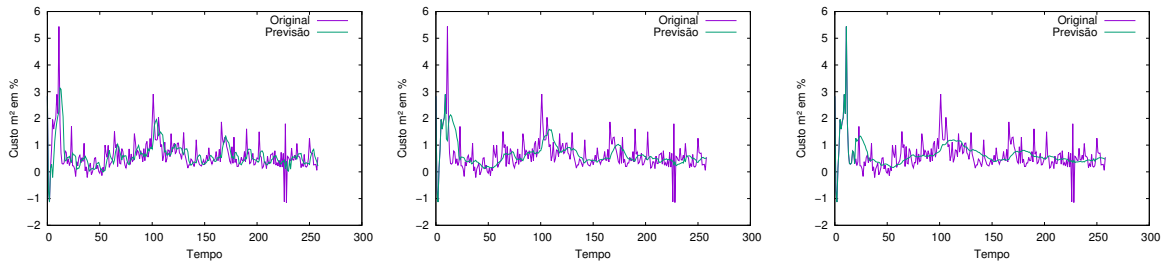

Figura 4.7: Algoritmo MMS utilizado na série S3 para L=4, L=10 e L=20 respectivamente.

usado pra transformada que poderia concentrar mais ainda as baixas frequências no ponto central.

Uma curiosidade que foi vista durante os testes é em relação à operação de deslocamento no eixo X. Foi visto que ela gera um crescimento significativo nos coeficientes da transformada de forma proporcional, apesar de manter a forma da função, a amplitude dos coeficientes é aumentada. Isso pode trazer, dependendo da aplicação, um problema de overflow, portanto, deve-se ter cuidado ao utilizar essa operação.

4.4.3 MMS: Série S3

Como último teste do método MMS, é testado sua atuação na série S3. A série S3 apresenta uma alta variação durante todo seu percurso. Os resultados gerados no teste podem ser vistos na imagem 4.7.

Percebe-se que o algoritmo conseguiu uma boa aproximação em todo percurso, Como nos outros casos ouve uma suavização da função resposta com o aumento do valor de N, mas, mesmo assim, o algoritmo ainda obteve resultados razoáveis.

Será feito o mesmo teste realizado anteriormente para testar a Transformada de Fourier, utilizando o algoritmo com N=20, por possuir uma qualidade de resposta e um N de tamanho razoáveis. O resultado é apresentado na imagem 4.8.

O mesmo resultado obtido anteriormente aconteceu. Porém, nesse caso, tem-se uma função com uma alta variação e, mesmo assim, isso não gerou alterações visíveis na qualidade da função. Um possível fator que explica esse resultado pode ser visto numa convergência já estudada. A principal operação que compõem o algoritmo MMS reconstruído é a função de somatório, já que a operação de divisão pode ser feita diretamente

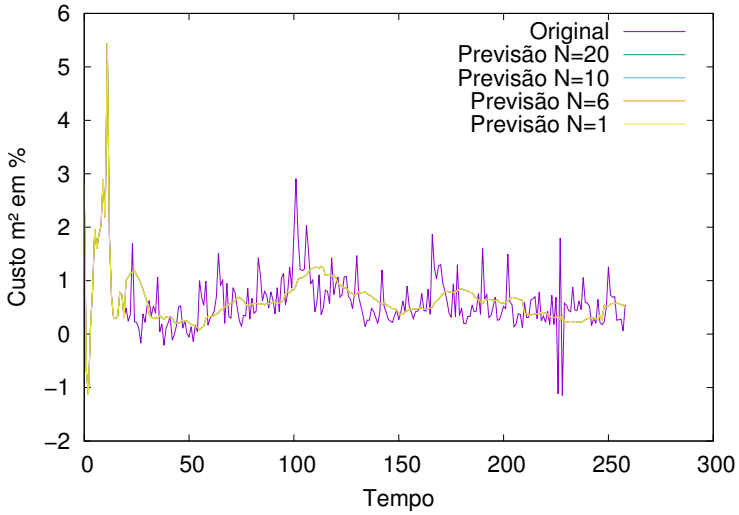

Figura 4.8: Algoritmo MMS utilizado na série S3 para L=20 e variando o número de frequências N.

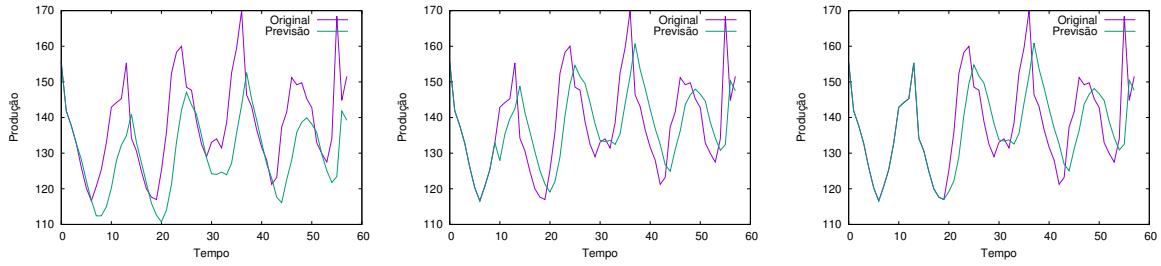

Figura 4.9: Algoritmo AES utilizado na série S1 para L=4, L=10 e L=20 respectivamente.

sobre o resultado da função de somatório. Observando a convergência encontrada na seção 3.6 para a operação, vê-se que ela é próxima de constante, portanto, isso explicaria o não acréscimo de erro com o decréscimo do número de coeficientes da transformada.

4.4.4 AES: Série S1, S2 e S3

O algoritmo AES foi utilizado nas três séries temporais já apresentadas. Os resultados gerados apresentam características muito semelhantes e por isso serão apresentados em uma mesma seção. Os resultados são apresentados nas imagens 4.9, 4.10 e 4.11

O problema da média, encontrado no algoritmo de MMS não se apresenta aqui, irrelevante ao número de pontos utilizados, a curva não é suavizada. Também foi percebido que a ponderação utilizada gera uma alta prioridade para pontos mais próximos do valor que se quer prever e gera uma baixa variação quando aumentamos L para um L já suficientemente grande. Isso também aperfeiçoa a representação de curvas com alta

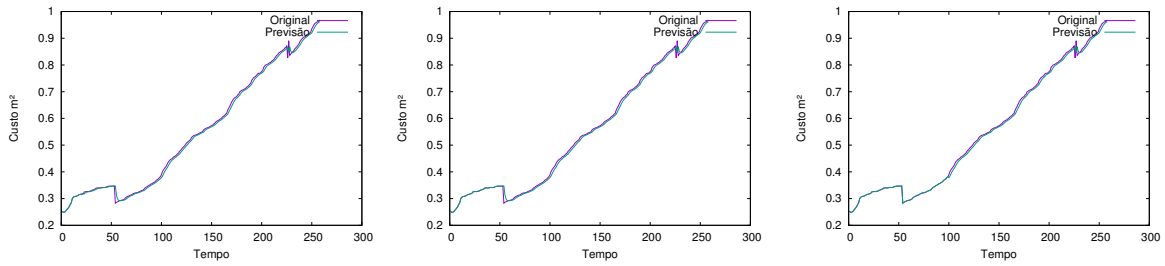

Figura 4.10: Algoritmo AES utilizado na série S2 para L=20, L=50 e L=100 respectivamente.

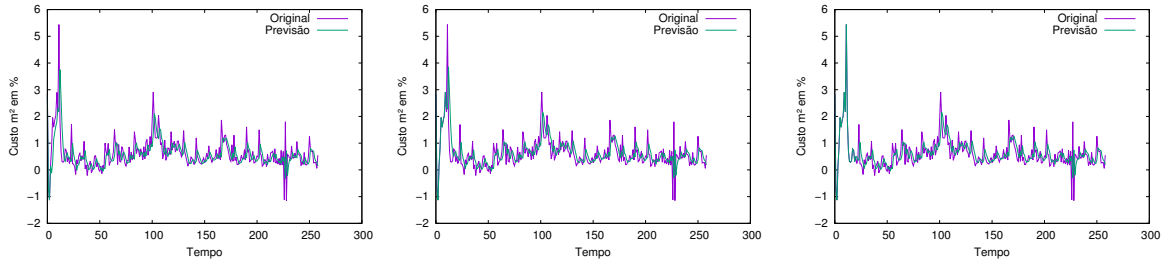

Figura 4.11: Algoritmo AES utilizado na série S3 para L=4, L=10 e L=20 respectivamente.

frequência como S1 e S3 e curvas com uma variação contínua, como o caso de S2, já que essas variações são ponderadas em relação à distância do objetivo.

Uma analise dos resultados do algoritmo variando N assim como foi feito para os exemplos do algoritmo MMS não sera feito. O motivo pelo qual é estudado o resultado do algoritmo com a diminuição do numero de coeficientes utilizados é a realização de um algoritmo que consiga gerar um resultado com pouca perda de qualidade de resposta e com um tempo de processamento menor. Porém, para o caso do AES, tem-se a aplicação da convolução que é de ordem $O(n^2)$ que se da por n pontos aplicados a n frequências. Assim, mesmo que essa operação seja realizada utilizando apenas 1 frequência sem perdas consideráveis, o que seria o melhor resultado possível, sua complexidade seria O(n) que é a complexidade da operação no domínio original. Isso implica que, mesmo obtendo o melhor resultado possível utilizando essa operação, ainda não seria obtida melhorias em relação a sua realização habitual.

A conclusão da aplicação da Transformada de Fourier no algoritmo AES é que, apesar de se conseguir realizá-lo totalmente no domínio comprimido, isso não traz melhorias para o algoritmo por aumentar sua complexidade temporal. Isso mostra que a aplicação da Transformada deve ser usada com cautela.

4.5 Conclusão

O capitulo mostrou uma forma de, utilizando as operações vistas ate aqui, remodelar dois algoritmos famosos de previsão utilizando a Transformada de Fourier. Aplicou-se o método remodelado a um conjunto de 3 funções de exemplos e percebemos com seus resultados que é possível remodelar todas as operações necessárias com a transformada. Em relação a eficiência do método foi visto que o caimento de coeficientes no algoritmo altera muito pouco a eficiência do método e, com isso, é possível melhorar bastante o numero de dados utilizados no algoritmo e consequentemente a velocidade do mesmo. Porém, foi visto que, para um dos algoritmos, não conseguimos um conjunto de operações que o realiza-se de forma mais rápida que sua formulação habitual.

5 Geração de Carga

Nos capítulos 2 e 3 foram estudadas algumas características da Transformada de Fourier, algumas propriedades relacionadas as funções utilizadas e outras relacionadas as possibilidades com a transformada. No capítulo 4 foi apresentado um primeiro exemplo completo de uma aplicação da Transformada de Fourier. Foi mostrado um conjunto de séries temporais que foram utilizadas, algoritmos famosos e os resultados da aplicação da transformada sobre essas estruturas remodelando o algoritmo e o aplicando as séries temporais de teste.

Neste capítulo, será apresentada a aplicação da Transformada de Fourier, em outro problema, quer-se representar a carga de acessos de um evento esportivo de larga escala na Internet. O objetivo em gerar modelos de carga é que, sua geração pode resultar no controle do sub ou super-dimensionamento da capacidade alocada de um sistema, o que pode aplicar em custos adicionais a essas alocações (Pedroso et al, 2006). Para isso, inicialmente é apresentado o conjunto de dados utilizados, Será criado um algoritmo que realize a função de geração de carga com as operações apresentadas no capítulo 3, por fim serão analisados os resultados encontrados.

5.1 Problema de Geração de Carga

Eventos de larga escala na Internet são cada dia mais comuns. Todos os grandes eventos recentes, como a copa do mundo de futebol ou os atuais jogos olímpicos, contam com a Internet como uma de suas principais plataformas de disseminação. Tais eventos geram uma classe de serviço que impõem fortes requisitos na infraestrutura e no ambiente de execução.

Diversos estudos caracterizam eventos deste tipo (Erman et al (2013); Almeida et al (2015); Santos et al (2004)). Estes eventos, em sua maior parte, se preocupam em gerar modelos que descrevam como será o comportamento dos clientes que acessam serviços relacionados a eventos de grande escala, ou como será o tráfego de rede observado durante

um grande evento na Internet. Um dos principais problemas para se caracterizar e gerar modelos precisos é o grande volume de dados associados a um grande evento na Internet. Em geral, são diversos dias associados a um evento, com um número inimaginável de clientes em uma escala global. Da mesma forma, a caracterização e os modelos gerados também contêm um grande número de dados e variáveis.

5.2 Conjunto de Dados

Os dados analisados foram retirados de registros de acesso gerados pelos servidores de mídia ao vivo durante a transmissão dos 64 jogos da copa do mundo de futebol da FIFA, realizado entre junho e julho de 2014. Todos os jogos do evento foram transmitidos por redes de televisão, por rádio e pela Internet. Foram coletados registros de acesso da transmissão pela Internet em todos os dias em que ocorreram jogos. Em geral, os servidores atenderam entre 300 mil e 1,1 milhões de IPs únicos durante a transmissão de cada jogo, com picos de até 320 mil endereços IP únicos simultâneos. Foi observado que, para um único jogo, um volume total de tráfego de rede de até 300 TB e picos de até 60 GB/s. Um estudo sobre os jogos utilizados pode ser encontrado em (Santos, 2016). A base de dados se constitui dos acessos de IPs únicos a cada jogo. Foi usado um período de tempo de 30 minutos antes do início do jogo até 150 minutos após seu início.

5.3 Método Proposto

Agora será apresentado uma proposta para um método de geração de carga, para isso, é apresentada uma proposta de algoritmo simples que encontra na base de jogos duas curvas que sejam mais parecidas, isso é, que gerem o menor somatório do módulo da diferença entre cada ponto de duas funções, após isso, modifica-se a amplitude de uma das curvas para que ela se aproxime mais da outra. Mais precisamente é feito a seguinte ordem de procedimentos.

1. É encontrado o jogo B mais parecido com o jogo objetivo A
2. É usado a Transformada de Fourier nas duas funções

3. É retirado o somatório SA e SB das funções A e B respectivamente

4. B é multiplicado por SA/SB

5. A Transformada Inversa de Fourier é aplicada em B

A partir do procedimento especificado acima, foi criado um algoritmo que o simula e que poderá ser diretamente reconstruído com a Transformada de Fourier. O Algoritmo 1 utiliza 3 constantes: NTotal que é a quantidade total de pontos que definem a funções; NPrevisão que representa o tempo utilizado para previsão, neste trabalho 30 minutos ou 3600 segundos, e F que é o N da Transformada de Fourier ou o número de coeficientes que foi utilizado para previsão. Note que, há um pré-processamento de eventos conhecidos e que a busca por similaridade ocorre com dados do pré-evento (e.g. 30 minutos anteriores ao jogo). As cinco funções utilizadas no algoritmo serão especificadas abaixo:

Algoritmo 1: Algoritmo de predição

```
1 Pré-Processamento Declara Variaveis NTotal ← NúmeroDePontosDaFuncao
  NPrevisao ← Primeiros30Minutos F ← NúmeroDeCoeficientes
  Tempo[NPrevisao] ← FuncaoInicial Prepara funções
  FreqTempo[F] ← TransfFourier(Tempo, F, NPrevisao)
  Solucao[NTotal] ← GeraFuncProxima(Tempo, NPrevisao)
2 FreqSol[F] ← TransfFourier(Solucao, F, NTotal)
3 SolPrevisao[NPrevisao] ← Solucao[NPrevisao]
4 FreqSolucaoPrevisao[F] ← TransfFourier(SolucaoPrevisao, F, NPrevisao) Previsão
  SomatorioTempo ← SomatorioDF(FreqTempo, NPrevisao)
5 SomatorioSol ← SomatorioDF(FreqSolucaoPrevisao, NPrevisao)
6 Heuristica ← SomatorioTempo/SomatorioSol
7 FreqSolucao ← Multiplicação(FreqSolucao, Heuristica)
8 Solucao ← TransfInversaFourier(FreqSolucao, F, NTotal)
```

(I) $TransfFourier(Tempo, F, NPrevisao)$ e $TransfInversaFourier(FreqSolucao, F, NTotal)$: são derivadas das fórmulas originais da transformada e sua inversa. O $Tempo$ representa a função a ser transformada, F o número de frequências utilizadas na transformada e, $NPrevisao$ o número de pontos da função Tempo. $FreqSolucao$ é o conjunto de coeficientes da transformada e $NTotal$ a quantidade de pontos da função de saída.

(II) $GeraFuncProxima(Tempo, NPrevisao)$: retorna como saída a função de um jogo dentre os jogos conhecidos que tenham os primeiros 30 minutos mais parecidos com nossa função de entrada $Tempo$. Para isso, ela calcula o somatório da diferença entre o módulo de duas funções nos seus NPrevisao segundos.

(III) $SomatorioDF(FreqTempo, NPrevisao)$: Para essa função implementamos a segunda operação apresentada na seção 3.6. A escolha se deve, logicamente, pela melhor

convergência gerada por esse método.

(IV) $Multiplicação(FreqSolucao, Heuristica)$: A multiplicação da função foi dada pela implementação da operação dada na seção 3.1.

Perceba que a função $GeraFuncProxima(Tempo, NPrevisao)$ é definida por NPrevisao que, apesar de, no problema, ser tratada como constante, pode variar dependendo da função de entrada. Essa operação poderia ser feita no domínio comprimido, mas isso poderia implicar em erros no resultado. Portanto, preferiu-se considerá-la como pré-processamento e gerar um resultado mais próximo do ótimo do algoritmo, já que só ocorrerá uma vez. Trabalhos que realizem essa operação no domínio compactado e analisem seus resultados podem ser feitos no futuro.

Das 5 funções utilizadas no algoritmo, duas são caracterizadas pelas próprias transformadas. Duas outras funções possuem ordem de complexidade definidas por F, que pode ser definido como constante e irrelevante ao problema. Em resumo, a ordem de complexidade do algoritmo proposto é O(N logN) para as transformadas, O(NPrevisao) no pré-processamento, onde é representado o tamanho de interesse dos dados usados para a previsão e O(1) para a previsão, pois o algoritmo é definido por F que é constante.

Em comparação a um possível algoritmo feito sem a Transformada de Fourier, desconsiderando o pré-processamento, tem-se o custo do algoritmo sendo pelo menos O(N) onde esse N representaria um custo de percorrer todos os pontos da função para realizar a operação de somatório e multiplicação. Ou seja, a proposta conseguiu, além de reduzir o número de dados usados para a representação dos dados (será melhor analisado na próxima seção), diminuir a complexidade temporal de parte do algoritmo, ou seja, desconsiderando o pré-processamento onde é realizado a Transformada de Fourier, o custo final que seria de O(N) pode ser reconstruído em O(1) e O(NPrevisao).

5.4 Experimentos e Resultados

O Algoritmo 1 foi realizado em todos os dados relativos aos 64 jogos. Foi utilizado um número de frequências variando de 3 a 100 e um total de minutos para a previsão igual a 30 minutos. Todos os jogos da base foram avaliados e serão analisados. Para cada avaliação, os dados do próprio jogo são retirados do conjunto total de dados pré-

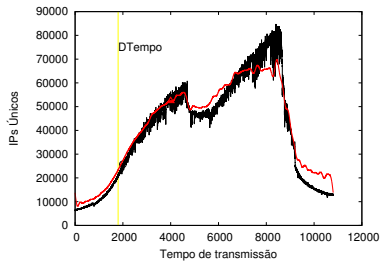

Figura 5.1: (a) Argentina x Irã

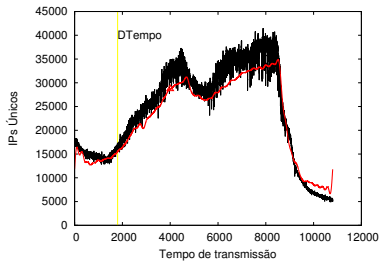

Figura 5.2: (b) Colombia x Uruguai

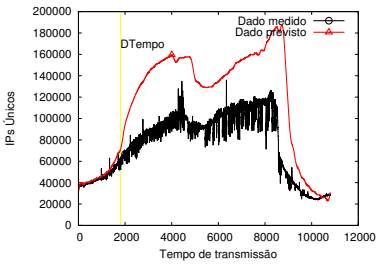

Figura 5.3: (c) Alemanha x Brasil

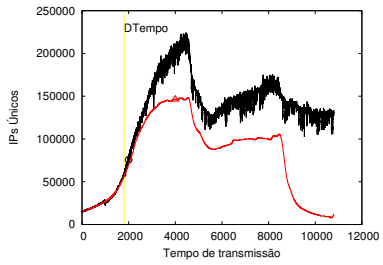

Figura 5.4: (d) Alemanha x Argelia

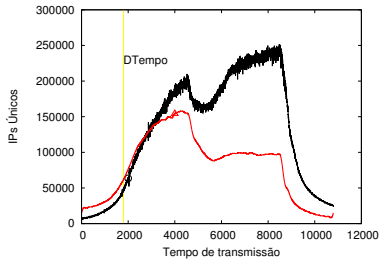

Figura 5.5: (e) Alemanha x EUA

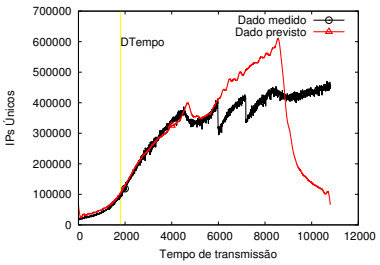

Figura 5.6: (f) Argentina x Suiça

Figura 5.7: Exemplos de uso do método de geração de carga.

processados. Finalmente, os erros relativos são verificados a cada instante da partida, com granularidade de 1 s.

Os erros apresentados são gerados com suas curvas deslocadas no eixo Y, objetivando aproximar mais as curvas sem alterar suas estruturas. A função de deslocamento pode ser facilmente realizada com a operação especificada na seção 3.2. Considerando que essa operação pode ser realizada em tempo O(1), ela não gera maiores custos para o algoritmo usado.

Alguns dos jogos são representados nas Figuras 5.7. A maioria dos jogos, como se vê na figura, se inicia com um crescimento rápido no seu número de expectadores, há um pico no 1º tempo da partida (1800s a 4500s), uma queda visível que acompanha o intervalo (4500s a 5400s), e um outro momento de grande demanda no 2º tempo da partida (5400s a 8100s). Assim que a partida termina (além de 8100s), há uma queda acentuada no número de expectadores.

As partidas 5.7-a e 5.7-b têm variações no número máximo de expectadores e detalhes ao longo de suas transmissões (como crescimentos ou períodos estáveis). Entretanto, o algoritmo proposto foi capaz de predizer o comportamento de toda a partida,

mesmo contando apenas com dados de 30 minutos anterior a seu acontecimento.

As Figuras 5.7-c a 5.7-f mostram casos que o método proposto não se encaixou bem. O método foi capaz de recriar as tendências, mas o erro relativo se apresentou alto. Intencionalmente, o método não conta com informações de contexto. A partida da figura 5.7-c aconteceu em feriado nacional, onde os usuários tradicionais de transmissões pela Internet preferiram assistir os jogos pela TV aberta. A partida ilustrada na figura 5.7-d e 5.7-f foram jogos atípicos, onde as partidas tiveram prorrogação. Finalmente, o jogo da figura 5.7-e foi um jogo com demanda diferencial de todas as demais partidas e é considerado o evento com maior transmissão ao vivo pela Internet até o momento.

Visualmente, as predições realizadas para as partidas das figuras 5.7-a e 5.7-b apresentam baixo erro em relação as previsões geradas. As partidas das figuras 5.7-e e 5.7-f tem boa aproximação, mas o algoritmo não se ajusta ao tempo de prorrogação.

Para todos os jogos, os erros relativos são maiores, quanto mais distante o momento avaliado. De acordo com a figura 5.8, para todos os jogos, até 80% dos pontos de comparação tem erros relativos inferiores a 10%, quando o tempo avaliado está distante até 120 minutos a partir do primeiro ajuste do algoritmo. Para ajustes maiores que 120 minutos (a 3a. hora avaliada), os erros relativos são de pelo menos 20%, para a grande maioria dos pontos avaliados. Tal comportamento é esperado, pois o algoritmo utiliza apenas um período anterior ao jogo para ajustar a predição que equivale aos 30 minutos antes do jogo começar.

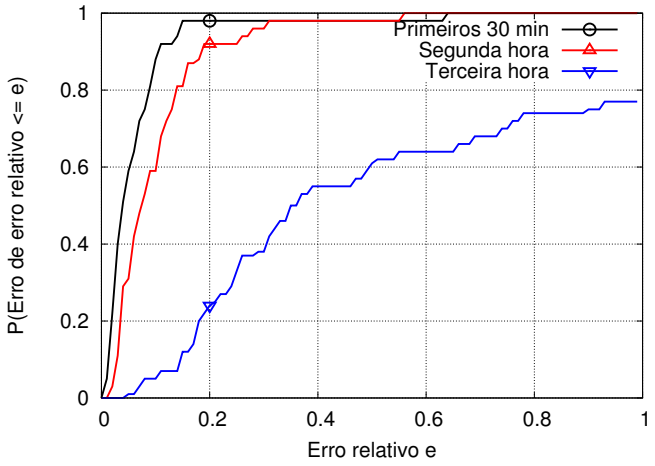

Figura 5.8: Erro gerado por jogo da base de jogos

Analisando a Figura 5.9, é visto novamente que, para todos os jogos, os erros

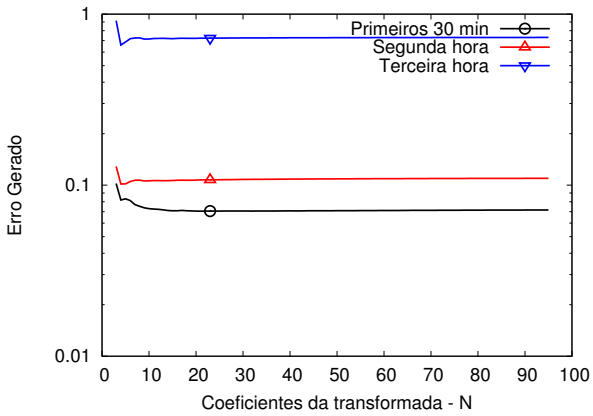

Figura 5.9: Erro médio gerado na nossa base de jogos variando o valor de N.

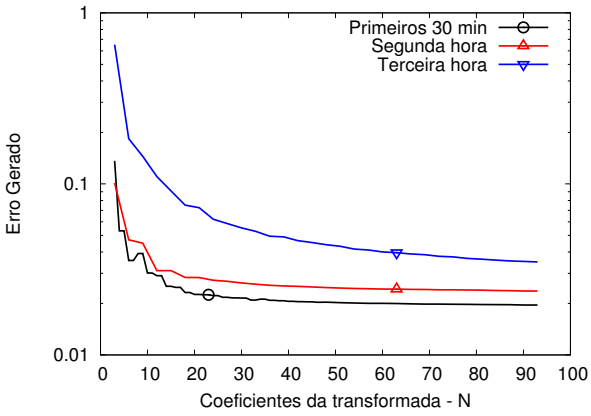

Figura 5.10: Composição do erro gerado pela Transformada de Fourier mais a operação de Multiplicação e Somatório.

relativos são maiores quanto mais distante o momento avaliado. De acordo com a figura, percebe-se que o erro médio para os primeiros 30 minutos de jogo se mantém abaixo de 10%, os próximos 60 minutos do jogo o erro aumenta, mas ainda se mantém próximo de 10% e apenas na próxima hora o erro se mostra alto em relação aos outros períodos.

A figura 5.10 dá um comparativo que permitirá analisar melhor os resultados, ela apresenta o erro médio gerado em toda a base de jogos pela composição das operações apresentadas nas seções 2.1(Transformada de Fourier), 3.1(Multiplicação por escalar), 3.6(Somatório). É feita uma multiplicação sobre os coeficientes gerados pela Transformada de Fourier e depois é calculado o somatório da função gerada, é retirada a variação desse resultado com a função gerada por todas essas expressões feitas no domínio original. Como era esperado, pelas análises anteriores, os erros se aproximam muito rapidamente de 0.

É interessante perceber que todas as operações usadas para calcular o erro apresentado na figura 5.9 estão presentes no algoritmo, com isso, o erro gerado por essas operações compõe o erro gerado pelo algoritmo mostrado na figura. 5.9. Considerando que o erro do algoritmo é próximo de constante e não varia com N e que o erro gerado pelas operações varia com N, pode-se concluir que o erro das operações é suficientemente pequeno ao ponto de não alterar a precisão do algoritmo de previsão, ou seja, o erro gerado pelo algoritmo e mostrado na figura 5.9 representa em sua maioria o próprio erro do algoritmo de previsão utilizado.

Pode-se concluir dos resultados, que o método proposto não gera maiores erros ao

algoritmo original para um número pequeno de frequências (em torno de 5), conseguindo reduzir drasticamente o valor de N e, com isso, uma diminuição dos dados utilizados para representar a função, anteriormente 10800 segundos agora em apenas 5 pontos.

6 Conclusão

Ao longo do trabalho foram mostrados um conjunto de propriedades e aplicações de uma transformada matemática no contexto da computação. Especificamente foi utilizado a Transformada de Fourier para sintetizar procedimentos e análises de interesse ao trabalho.

Uma rápida introdução a Transformada de Fourier foi dada e seguiu-se para sua aplicação em funções. Destacando que, o principal objetivo de usar a transformada era conseguir uma compactação dos dados, foi estudada a aplicação da transformada em funções com uma forma de diminuir a quantidade total de dados que representava aquela função, acrescentando um erro a sua representação. Esse erro foi estudado e analisado para diferentes funções e para diferentes características de funções.

Após um estudo sobre a aplicabilidade da transformada em funções foi visto sua aplicabilidade em operar funções. Um conjunto extenso de operações foi explicitado e analisado, conjunto este que conseguiu sintetizar alguns algoritmos posteriormente.

Por fim, dois problemas foram introduzidos ao trabalho, o problema de previsão e de geração de carga. Para ambos os problemas, um conjunto de dados foi apresentado e um algoritmo foi explicitado ou criado utilizando a Transformada de Fourier para realizar as operações que compunham esse algoritmo. Testes foram realizados e os resultados foram analisados.

Foi percebido que, em geral a Transformada de Fourier consegue sintetizar com muita eficiência e poucos dados, séries temporais reais que possuem pouca variação durante o tempo. Notamos que a transformada possui um grau de convergência que pode depender da primeira e segunda derivada da função, além que, a derivada pontual pode ser importante para definir a convergência de um conjunto restrito de pontos vizinhos. Apesar disso, foi visto também, que a convergência de qualquer função pode variar muito já que, em geral, cada ponto de uma função é definido por todos os pontos dessa mesma função na Transformada de Fourier.

Foi visto também que a Transformada de Fourier consegue sintetizar uma grande quantidade de operações, na maioria das vezes, sem prejudicar a complexidade temporal

da operação, porém, em alguns casos, ela obteve resultados melhores ou piores. Viu-se que um grande desafio da transformada é sintetizar operações que atuam em um único ponto da função como o GET e SET, mas que em geral, ela possui grande eficiência em operações que atuam em todo conjunto de pontos da função.

Como primeira forma de testar a aplicabilidade das operações, foram apresentados dois algoritmos famosos de previsão e a partir das operações conseguiu-se remodelar tais algoritmos de forma a manter seus resultados. Foi visto que a transformada gera um ganho no algoritmo ao utilizar menos coeficientes e que tal dimuição altera muito pouco a qualidade do resultado do algoritmo. Porém, foi visto que, em um dos dois algoritmos, apesar de se conseguir uma total remodelagem do mesmo com a transformada, gerou-se um novo algoritmo com um aumento na complexidade temporal do método, o que nos mostra que deve haver um cuidado ao utilizar a transformada.

Por fim, a transformada de Fourier foi utilizada para a geração de carga em dados gerados por grandes eventos na Internet. A partir de um conjunto de operações apresentadas no trabalho, simulou-se um método de geração de carga diminuindo sua possível ordem de complexidade temporal prevista para O(N) e simulada em O(1). Ainda é perceptível que essa modificação não prejudicou substancialmente o erro gerado pelo algoritmo, que se manteve próximo de 10%, mesmo utilizando apenas um tamanho equivalente a 0,04% do tamanho da função original.

Conclui-se que, em geral a Transformada de Fourier pode ser utilizada para sintetizar vários algoritmos, que ela pode gerar melhorias significativas em relação à complexidade do tempo de processamento e ao espaço utilizado para o processo e com isso um ganho paralelo à velocidade total do algoritmo. Porém, a função utilizada como base de dados, as operações que compõem o algoritmo, assim como, suas eficiências temporais devem ser analisadas com cuidado para garantir a eficiência da aplicação da Transformada de Fourier.

Como trabalhos futuros pode ser analisado características de convergência da Transformada de Fourier para outras funções assim como para outras características de funções. Pensar num conjunto complementar de operações e analisar as já criadas a fim de melhorar sua eficiência temporal ou espacial, melhorando e analisando a convergência

da operação para diversas funções. Outros algoritmos de previsão mais bem elaborados podem ser sintetizados usando a Transformada de Fourier ou outras transformadas. Novos problemas ou algoritmos podem ser apresentados a fim de reproduzir a ideia mostrada no trabalho com a Transformada de Fourier ou outras transformadas.

Bibliografia

Ahmed, N.; Natarajan, T. ; Rao, K. R. Discrete cosine transfom. **IEEE transactions on Computers**, , n.1, p. 90–93, 1974.

Almeida, W.; Santos, B.; Vieira, A. B.; Cunha, Í. ; Almeida, J. Caracterizaç ao da transmissao de um grande evento esportivo. 2015.

Bracewell, R. The fourier transform and is applications. **New York**, v.5, 1965.

Brigham, E. O.; Brigham, E. O. **The fast Fourier transform**, volume 7. Prentice-Hall Englewood Cliffs, NJ, 1974.

Cooley, J. W.; Tukey, J. W. An algorithm for the machine calculation of complex fourier series. **Mathematics of computation**, v.19, n.90, p. 297–301, 1965.

Dugad, R.; Ahuja, N. A fast scheme for image size change in the compressed domain. **Circuits and Systems for Video Technology, IEEE Transactions on**, v.11, n.4, p. 461–474, 2001.

Erman, J.; Ramakrishnan, K. K. **Understanding the super-sized traffic of the super bowl**. In: Proceedings of the 2013 conference on Internet measurement conference, p. 353–360. ACM, 2013.

Gonçalves, L. A. **Um estudo sobre a Transformada Rápida de Fourier e seu uso em processamento de imagens**. 2004. Tese de Doutorado - UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL.

Gottlieb, D.; Shu, C.-W. On the gibbs phenomenon and its resolution. **SIAM review**, v.39, n.4, p. 644–668, 1997.

Hu, O. R.; Raunheitte, L. T. Padrão jpeg de compactação de imagens. **Revista Mackenzie de Engenharia e Computação**, v.1, n.1, 2010.

IBGE. **Disponível em: <http://seriesestatisticas.ibge.gov.br>**. Acesso em, 2016.

Lelewer, D. A.; Hirschberg, D. S. Data compression. **ACM Computing Surveys (CSUR)**, v.19, n.3, p. 261–296, 1987.

Maçada, A. C. G.; Brinkhues, R. A. ; Júnior, J. C. F. Big data e as capacidades de gestão da informação. **ComCiência. Disponível:¡ http://www. comciencia. br/comciencia**.

Marcellin, M. W.; Gormish, M. J.; Bilgin, A. ; Boliek, M. P. **An overview of jpeg-2000**. In: Data Compression Conference, 2000. Proceedings. DCC 2000, p. 523–541. IEEE, 2000.

Meng, J.; Chang, S.-F. **Embedding visible video watermarks in the compressed domain**. In: Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on, volume 1, p. 474–477. IEEE, 1998.

Mitchell, J. L. **MPEG video compression standard**. Springer Science & Business Media, 1997.

Morettin, P. A.; de Castro Toloi, C. M. **Modelos para previsão de séries temporais**, volume 1. Instituto de matematica pura e aplicada, 1981.

Oppenheim, A. V.; Schafer, R. W. **Discrete-time signal processing**. Pearson Higher Education, 2010.

Pedroso, C. M.; Fonseca, K. **Um modelo para geração de carga de servidores web utilizando classificação de conteúdo**. In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES, 2006.

Rao, K. R.; Yip, P. **Discrete cosine transform: algorithms, advantages, applications**. Academic press, 2014.

Santos, F. J. Introdução às séries de fourier. **Belo Horizonte: PUC Minas**, 2004.

Shao, X.; Xu, C.; Wang, Y. ; Kankanhalli, M. S. **Automatic music summarization in compressed domain**. In: Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on, volume 4, p. iv–261. IEEE, 2004.

Stone, M. L. Big data for media. **Report from the Reuters Institute for the Study of Journalism**, 2014.

Wang, Y.; Vilermo, M. **A compressed domain beat detector using mp3 audio bitstreams**. In: Proceedings of the ninth ACM international conference on Multimedia, p. 194–202. ACM, 2001.