**Escola Politécnica da Universidade de São Paulo**

**MAP3121 - Métodos Numéricos e Aplicações**

**Exercício Programa 2 para Engenharia Elétrica**

**Recuperação**

**Gabriel da Cunha Rodrigues - 8992930**

**Prof. Antoine Laurain**

**Turma 2**

**São Paulo, 27 de Julho de 2018**

Sumário

[**1. Introdução**](#_kf79kj72vbwp)**…………………………………………………………………………………….. 3**

[**2. Objetivos**](#_iz60jw2nyt6t)**………………………………………………………………………………………. 4**

[**3. Metodologia**](#_s7shlf32udx3)**…………………………………………………………………………………... 5**

[**4. Desenvolvimento**](#_tfokq5t6nonr)**…………………………………………………………………………….. 7**

[**5. Resultados**](#_2e1mh4o2lbih)**…………………………………………………………………………………….. 8**

[5.1. Teste A](#_gtmx473ny56u)……………………………………………………………………………………. 8

[5.2. Teste B](#_zfuvmygogwhe)……………………………………………………………………………………. 8

[5.3. Teste C](#_igh890pea4vl)……………………………………………………………………………………. 9

[5.4. Decomposição de sons](#_g7r979w10ac9)…………………………………………………………………. 10

[5.4.1. Compressão](#_1368brmgxf8w)……………………………………………………………………….. 10

[5.4.2. Filtros](#_jch6ex6giytm)……………………………………………………………………………….. 10

# 1. Introdução

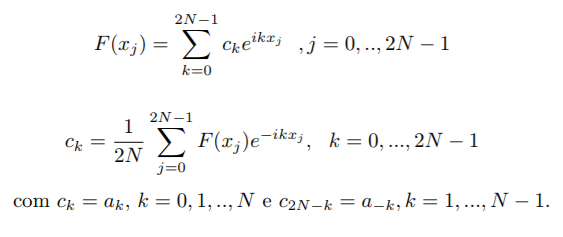
Este relatório consiste na documentação do Exercício Programa 2 para Engenharia Elétrica da disciplina de Métodos Numéricos e Aplicações.

# 2. Objetivos

O objetivo principal do exercício programa é realizar um estudo sobre sons, ou uma música simples, através da análise de Fourier, que permite a decomposição de um som em harmônicos, filtrando-se algumas frequências, de forma a remover graves ou agudos, ou testando-se a compressão do som, em que seu armazenamento é feito utilizando-se menos espaço. Ao final do exercício programa, pretende-se reconstruir o som utilizando-se a transformada inversa de Fourier, de modo a verificar se o resultado do som filtrado, comprimido ou não, foi satisfatório.

# 3. Metodologia

Para realizar os objetivos propostos, implentaram-se as transformadas diretas e inversas de Fourier, tanto através da forma recursiva da Transformada Rápida de Fourier (FFT), quanto na forma menos eficiente dada pelas equações das figuras abaixo:



*Figura 1 - Fórmula direta para a transformada de Fourier*

Através da transformada, é possível verificar as frequências dominantes do som analisado e a frequência de Nyquist, máxima frequência que pode ser amostrada com o número de pontos escolhidos, além de aplicar filtros que selecionam faixas específicas dessas frequências, ou seja, eliminando-se os sons mais graves, mais agudos, ou ambos. Esses filtros podem ser dos tipos passa-altas, passa-baixas ou passa-bandas, respectivamente.

Por meio da transformada inversa de Fourier, é possível reconstituir o som filtrado, com as frequências eliminadas, observando-se a diferença de eficiência entre os dois métodos utilizados e comparando-se também com a eficiência de rotinas de um pacote computacional fornecido (FFTPACK4), implementando-se as rotinas *ezffti*, *ezfftf* e *ezfftb,* específicas para dados reais.

Para realizar as análises de Fourier de sinais sonoros, foi necessário utilizar o software SOX para obter os dados necessários dos arquivos .wav fornecidos como dados para o exercício programa e para ouvir o som reconstituído pela transformada inversa em dados numéricos.

Por fim, fizeram-se necessários alguns tratamentos de dados, como para o caso de dados incompletos, em que completaram-se as lacunas através de interpolação cúbica, quando havia dois valores anteriores e posteriores próximos ao ponto desejado, ou interpolação linear quando havia apenas um valor em um dos lados. Também completou-se a tabela de dados com o valor médio dos dados existentes, de modo a totalizar uma potência de 2 quando não havia essa quantidade.

# 4. Desenvolvimento

Inicialmente, foram desenvolvidas funções básicas para execução do programa, sendo elas *tamanho\_arquivo*, que lê o arquivo .dat para verificar o número de amostras realizadas, que é utilizado para a alocação dos vetores; *ler\_arquivo\_dat*, que de fato recolhe informações do arquivo .dat e coloca os valores nos vetores respectivos; *escrever\_arquivo\_dat*, que escreve os mesmos vetores depois da análise e modificação em um arquivo .dat na mesma formatação dos arquivos fornecidos; *teste\_complexos*, que foi confeccionada para testar o pacote de variáveis complexas no C.

Em seguida, fez-se funções mais específicas com as necessidades dos problemas. As funções referentes à transformada de Fourier foram *fftrec*, que aplica o algoritmo fornecido no enunciado para a FFT de implementação recursiva; *filtro*, que aplica um filtro em um vetor de coeficientes, zerando os que não estão na especificação da frequência de corte fornecida. A FFTPACK4 foi implementada diretamente nas funções específicas dos testes iniciais e também na *main*.

Também foram feitas 3 funções para cada teste inicial: *teste\_inicial\_a*, *teste\_inicial\_b* e *teste\_inicial\_c*. Nelas foram utilizadas tanto as funções fftrec quanto as do FFTPACK4, para obter duas formas distintas de análise e permitir a comparação entre elas.

Ao longo do desenvolvimento do exercício programa, realizaram-se diversas alterações visando a melhoria da eficiência do programa e tomando cuidados como não deixar variáveis não utilizadas e desalocar os espaços de memória das matrizes e vetores dinâmicos. O código foi amplamente comentado, para esclarecer o significado de cada variável adotada, facilitar o entendimento de cada função e a importância de cada bloco.

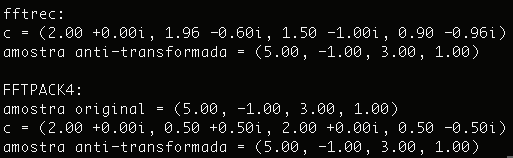
A função *main* integra todas as funções, executando o teste ou abrindo o arquivo a ser analisada através de um seletor determinado pelo usuário. O resultado de cada teste e rede é mostrado no item 5.

# 5. Resultados

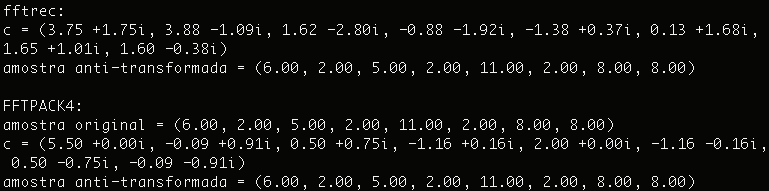
Apesar de inúmeros esforços para depurar o código, não foi possível encontrar o erro que o levou a produzir resultados incorretos da função *fftrec*. O código foi inteiramente baseado no pseudo-código mostrado no EP, porém não foi possível obter os mesmos resultados da FFTPACK4. O vetor de amostras, no entanto, é regenerado como deveria, indicando uma transformação bijetora.

O resultado de cada caso de teste pode ser observado abaixo.

## 5.1. Teste A

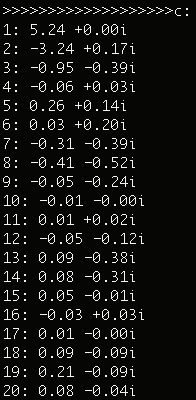


## 5.2. Teste B



## 5.3. Teste C





## 5.4. Decomposição de sons

### 5.4.1. Compressão

Existem duas formas eficiente de se comprimir um arquivo de som.

A primeira é, ao invés de armazenar todos os valores amostrados no tempo, armazenar os coeficientes de Fourier, e quando for necessário recompor o som para ouvir a música, aplica-se a transformada inversa de Fourier. Desta maneira o arquivo ocupa um espaço menor no computador. É possível reduzir as harmônicas também, o que implica em um erro associado, porém muitas vezes imperceptível ao ouvido humano.

A segunda maneira, e mais simples, é filtrar as faixas de frequência não audíveis para o ouvido humano. Sabendo-se que a audição humana escuta frequências entre 20 Hz em 20.000 Hz, é possível usar a transformada de Fourier para encontrar e eliminar frequências fora dessa faixa.

Apesar de pleno conhecimento de como deveria ser aplicado este filtro, não foi possível colocá-lo em prática neste EP por falta de tempo hábil para tal.

### 5.4.2. Filtros

Um filtro de frequências pode se constituir em 3 diferentes tipos: filtros passa-baixas, filtros passa-altas e filtros passa-bandas.

O primeiro permite passar frequências menores que uma determinada frequência de corte K. O segundo, frequências maiores que uma determinada frequência de corte K. O terceiro, frequências entre uma frequência de corte inferior dita KL e outra superior, KH.

Aplicar um filtro utilizando uma transformada de Fourier é simples: basta deixar os coeficientes ck que podem passar pelo filtro intactos e zerar os que estão fora da frequência de corte estabelecida (ou das frequências, no caso do passa-bandas).

O protótipo da função foi feita, porém não houve tempo de testá-la. Abaixo se encontra a função conforme descrita.

