UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
http://www.icmc.usp.br/

Professor: Rodrigo Fernandes de Mello (mello@icmc.usp.br)
Alunos PAE: Lucas Pagliosa (lucas.pagliosa@usp.br)
Felipe Duarte (fgduarte@icmc.usp.br)

Trabalho 02: Transformada de Fourier Aplicada a Áudio

1 Prazos e Especificações

O trabalho descrito a seguir é individual e não será tolerado qualquer tipo de plágio ou cópia em partes ou totalidade do código. Caso seja detectado alguma irregularidade, os envolvidos serão chamados para conversar com o professor responsável pela disciplina.

A entrega deverá ser feita única e exclusivamente por meio do sistema run.codes no endereço eletrônico `https://run.codes` até o dia **09 de setembro de 2016 às 23 horas e 59 minutos**. Sejam responsáveis com o prazo final para entrega, o run.codes está programado para não aceitar submissões após este horário e não será aceito entrega fora do sistema.

Leia a descrição do trabalho com atenção e várias vezes, anotando os pontos principais e as possíveis formas de resolver o problema. Comece a trabalhar o quanto antes para não ficar dúvidas e você consiga entregar o trabalho a tempo.

2 Descrição do Problema

Uma das ferramentas mais úteis na análise e processamento de imagens e sinais é a transformada de Fourier. Portanto, é muito provável que você reveja e replique os conceitos envolvidos nesta transformada no decorrer do curso. Logo, mais do que apenas uma descrição de trabalho, este documento tem como propósito explicar passo-a-passo como funciona a transformada discreta e contínua de Fourier.

Uma vez entendido como o algoritmo funciona, você deve implementar a transformada discreta de Fourier (Equação 10) e sua inversa (Equação 11) e aplicá-las sobre sinais de áudio a fim de manter os componentes mais relevantes de um sinal. Mais detalhes sobre a aplicação na Seção 4.

3 Conceitos

Para entender a transformada discreta de Fourier, vamos recapitular e/ou introduzir, resumidamente e sem muito formalismo, os seguintes conceitos:

- Produto interno;
- Base e combinação linear;
- Propriedades de senos e cossenos;
- Série de Fourier;
- Transformada de Fourier.

Além do resultado prático, um dos fortes aspectos deste trabalho é ressaltar a relevância da matemática para a computação, dado o número de conceitos matemáticos necessários para entender (e portanto programar) a transformada discreta de Fourier. Apesar de, na prática, muitas pessoas utilizarem pacotes com código pronto e otimizado, é importante você programá-la ao menos uma vez para adquirir maturidade no que se refere a: i) ver um código/problema parecido e associar com Fourier; e ii) estender o algoritmo caso seja necessária uma variação no código.

Obs: Não assuste caso você nunca tenha visto integral antes! A integral $\int_a^b f(x)\mathrm{d}x$ significa **simplesmente** o somatório de *todos* os valores produzidos pela função $f(x)$ no intervalo $[a, b]$.

3.1 Produto Interno

O produto interno existe em um espaço métrico (onde existe a noção geométrica de objetos) se para todos os vetores $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathbb{C}^n$, as seguintes propriedades são satisfeitas:

- Simetria hermitiana: $\langle \mathbf{u}, \mathbf{v} \rangle = \overline{\langle \mathbf{v}, \mathbf{u} \rangle}$;
- Distributividade: $\langle \mathbf{u} + \mathbf{v}, \mathbf{w} \rangle = \langle \mathbf{u}, \mathbf{w} \rangle + \langle \mathbf{v}, \mathbf{w} \rangle$;
- Homogeneidade: $\langle \lambda \mathbf{u}, \mathbf{v} \rangle = \lambda \langle \mathbf{u}, \mathbf{v} \rangle$;
- Positividade: $\langle \mathbf{v}, \mathbf{v} \rangle \geq 0$,

em que λ é um subcorpo (número inteiro, racional, real ou complexo) e $\overline{z}$ é o conjugado de z.

Logo, o produto interno mais comumente utilizado entre dois vetores $\mathbf{u} = [u_1, u_2, \cdots =, u_n]$ e $\mathbf{v} = [v_1, v_2, \cdots, v_n]$ em um espaço métrico é dado por:

$$\langle \mathbf{u}, \mathbf{v} \rangle = u_1 v_1 + u_2 v_2 + \cdots + u_n v_n.$$

A partir do produto interno, pode-se incluir conceitos de ortogonalidade, norma, distância e ângulo entre dois vetores (o que caracteriza a geometria). No nosso caso estamos, mais interessados em definir a similaridade entre dois vetores, o que está diretamente relacionado com o ângulo entre os mesmos, definido por:

$$\cos(\theta)(\mathbf{u}, \mathbf{v}) = \left(\frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\|\mathbf{u}\| \, \|\mathbf{v}\|} \right),$$

em que $\|\mathbf{v}\| = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle}$ é a norma de $\mathbf{v}$ e

$$\cos(\theta)(\mathbf{x}, \mathbf{y}) = \begin{cases} 1, & \text{se os vetores } \mathbf{u}, \mathbf{v} \text{ têm a mesma direção e sentido,} \\ 0, & \text{se os vetores } \mathbf{u}, \mathbf{v} \text{ são ortogonais entre si,} \\ -1, & \text{se os vetores } \mathbf{u}, \mathbf{v} \text{ têm a mesma direção mas sentidos opostos.} \end{cases}$$

Quando estamos em um espaço de funções, o conceito de produto interno e similaridade é idêntico ao produto interno de vetores (uma vez que funções podem ser interpretadas como vetores contínuos – se houver maior interesse, procure por auto-espaços vetoriais). Logo, têm-se que o produto interno de duas funções f, g no intervalo $[a, b]$ é dado por:

$$\langle f, g \rangle = \int_a^b f(t) g(t) \mathrm{d}t, \tag{1}$$

tal que a similaridade entre duas funções é dada por $\cos(\theta)(f, g)$ sendo $\|f\| = \int_a^b \sqrt{f^2(t)} \mathrm{d}t$. No entanto, a similaridade entre duas funções não precisa estar normalizada no intervalo $[-1, 1]$ para indicar como uma função está relacionada à outra. A grosso modo, se não dividirmos pelas normas, apenas o produto interno já nos diz o quanto duas funções são similares: se este produto resultar em um número positivo, então as duas funções "apontam" ou "crescem" juntas; se resultar em zero, então não existe relação entre as duas funções; se produzir valor negativo, ambas são inversamente similares.

3.2 Base e Combinação Linear

Imagine que $\mathbf{f} = \sum_k c_k \mathbf{i}_k$ seja um vetor para uma receita de bolo em um espaço $\mathbb{C}^n$, em que c_k um escalar que define a quantidade do ingrediente $\mathbf{i}_k \in \mathbb{C}^n$. Suponha, por um instante, que somos estudantes da USP de ICC2 e não temos dinheiro para nada, nem mesmo a chance de transferir para outro oferecimento da mesma disciplina :-(, e queremos fazer o melhor bolo possível com apenas um ingrediente $\mathbf{i}_1$. Este bolo vai gerar um vetor de erro $\mathbf{e}_1 \in \mathbb{C}^n$:

$$\mathbf{f} = c_1\mathbf{i}_1 + \mathbf{e}_1,$$

que terá valor mínimo quando $\mathbf{i}_1$ for ortogonal ($\perp$) a $\mathbf{e}_1$ (i.e., ortogonal a $\mathbf{f} - c_1\mathbf{i}_1$), ou seja, o erro não tem relação com $\mathbf{i}_1$. Logo:

$$\begin{gathered}(\mathbf{f} - c_1\mathbf{i}_1) \perp \mathbf{i}_1, \\ \langle (\mathbf{f} - c_1\mathbf{i}_1), \mathbf{i}_1 \rangle = 0, \text{(Distributividade)} \\ \langle \mathbf{f}, \mathbf{i}_1 \rangle = \langle c_1\mathbf{i}_1, \mathbf{i}_1 \rangle, \text{(Homogeneidade)} \\ \langle \mathbf{f}, \mathbf{i}_1 \rangle = c_1 \langle \mathbf{i}_1, \mathbf{i}_1 \rangle.\end{gathered}$$

Quando ganhamos nossa primeira bolsa PET ou de monitoria, já conseguimos comprar n ingredientes para o bolo, de forma que agora temos:

$$\mathbf{f} = \left[\sum_{k=0}^{n} c_k \mathbf{i}_k \right] + \mathbf{e},$$

tal que as quantidades de ingredientes c_k ótimas são obtidas resolvendo o sistema linear (não é difícil mostrar, fica a seu critério):

$$\begin{bmatrix} \langle \mathbf{i}_1, \mathbf{i}_1 \rangle & \langle \mathbf{i}_1, \mathbf{i}_2 \rangle & \cdots & \langle \mathbf{i}_1, \mathbf{i}_n \rangle \\ \langle \mathbf{i}_2, \mathbf{i}_1 \rangle & \langle \mathbf{i}_2, \mathbf{i}_2 \rangle & \ddots & \langle \mathbf{i}_2, \mathbf{i}_n \rangle \\ \vdots & \cdots & \cdots & \vdots \\ \langle \mathbf{i}_n, \mathbf{i}_1 \rangle & \langle \mathbf{i}_n, \mathbf{i}_2 \rangle & \cdots & \langle \mathbf{i}_n, \mathbf{i}_n \rangle \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} \langle \mathbf{f}, \mathbf{i}_1 \rangle \\ \langle \mathbf{f}, \mathbf{i}_2 \rangle \\ \vdots \\ \langle \mathbf{f}, \mathbf{i}_n \rangle \end{bmatrix}.$$

Caso os ingredientes formem um sistema linearmente independente tal que $\langle \mathbf{i}_j, \mathbf{i}_k \rangle = 0, \forall j \neq k$, isto é, formam uma *base* (para mais informações sobre base algébrica sugere-se `http://ocw.mit.edu/courses/mathematics/18-06-linear-algebra-spring-2010/`), tem-se que:

$$c_k = \frac{\langle \mathbf{f}, \mathbf{i}_k \rangle}{\langle \mathbf{i}_k, \mathbf{i}_k \rangle}. \tag{2}$$

Note que todo conceito acima pode ser aplicado para funções na forma $f(t) = \sum_k c_k i_k(t)$.

3.3 Propriedades de senos e cossenos

Brevemente, das seguintes propriedades:

$$\int_T \cos(nt)\cos(mt)\mathrm{d}t = \begin{cases} 0, & n = m, \\ \frac{T}{2}, & n \neq m = 0; \end{cases} \tag{3}$$

$$\int_T \sin(nt)\sin(mt)\mathrm{d}t = \begin{cases} 0, & n = m, \\ \frac{T}{2}, & n \neq m = 0; \end{cases} \tag{4}$$

$$\int_T \cos(nt)\sin(mt)\mathrm{d}t = 0. \tag{5}$$

concluímos que o conjunto de funções $\{\sin(nt), \cos(mt)\}$ forma uma base algébrica no intervalo $[0, T]$. Outra propriedade importante que será utilizada futuramente na transformada de Fourier é a de arco duplo:

$$\cos(n+m) = \cos(n)\cos(m) - \sin(n)\sin(m). \tag{6}$$

Por fim, também utilizaremos a fórmula de Euler:

$$e^{ix} = \cos(x) + i\sin(x). \tag{7}$$

3.4 Séries de Fourier

O que Joseph Fourier percebeu é que se existe uma função periódica $f(t)$ com frequência angular $\omega = 2\pi/T$ definida no intervalo $[0, T]$, isto é, $f(t) = f(t+T)$. Podemos representar esta função pela soma de cossenos na forma $f(t) = \sum_{k=0}^{\infty} c_k i_k(t)$, tal que:

$$\begin{aligned} i_0(t) &= \cos(0\omega t) = 1, \\ i_1(t) &= \cos(1\omega t + \theta_1), \\ i_2(t) &= \cos(2\omega t + \theta_2), \\ &\vdots \end{aligned}$$

de forma que:

$$f(t) = c_0 + \sum_{k=1}^{\infty} c_k \cos(k\omega t + \theta_k), \text{(Equação 6)}$$

$$f(t) = c_0 + \sum_{k=1}^{\infty} c_k \cos(k\omega t)\cos(\theta_k) - c_k \sin(k\omega t)\sin(\theta_k),$$

fazendo $a_k = c_k \cos\theta_k$ e $b_k = -c_k \sin\theta_k$, temos a equação da série de Fourier completa:

$$f(t) = c_0 + \sum_{k=1}^{\infty} a_k \cos(k\omega t) + b_k \sin(k\omega t).$$

Baseado no conceito de produto interno (Equação 1), coeficientes de sistemas lineares (Equação 2) e ortogonalidade de pares de senos cossenos (Equações 3, 4, 5) temos que:

$$a_k = \frac{\langle f(t), \cos(k\omega t)\rangle}{\langle \cos(k\omega t), \cos(k\omega t)\rangle} = \frac{2}{T}\int_T f(t)\cos(k\omega t)\mathrm{d}t,$$

$$b_k = \frac{\langle f(t), \sin(k\omega t)\rangle}{\langle \sin(k\omega t), \cos(k\omega t)\rangle} = \frac{2}{T}\int_T f(t)\cos(k\omega t)\mathrm{d}t,$$

$$c_0 = \frac{\langle f(t), 1\rangle}{\langle 1, 1\rangle} = \frac{1}{T}\int_T f(t)\mathrm{d}x.$$

Para facilitar o problema e apresentá-lo de forma mais elegante, usamos a fórmula de Euler (Equação 7) para obter a série compacta de Fourier no intervalo $[0, T]$ da seguinte forma:

$$f(t) = \frac{\epsilon}{T}\sum_{k=0}^{\infty} c_k e^{ik\omega t}, \quad \epsilon = \begin{cases} 1, & k = 0, \\ 2, & k > 0. \end{cases} \tag{8}$$

Note que desta forma, cada coeficiente c_k corresponde a um número complexo tal que suas partes real e imaginária representam a_k e b_k, respectivamente.

Resumindo, uma função (um sinal, por exemplo) pode ser representada por um sistema linear de infinitas senoides. tal que cada senoide tem:

- frequência representada pelo índice k;
- amplitude do sinal igual a $c_k = \sqrt{a_k^2 + b_k^2}$;
- deslocamento de fase igual a $\theta_k = \tan^{-1}\left(\frac{-b_k}{a_k}\right)$.

3.5 Transformada Discreta de Fourier

Note que a série de Fourier é definida apenas para funções periódicas. Para funções não periódicas, temos *a mesma coisa* mas com a pequena restrição de não ser possível definir o sinal em um intervalo fechado de período T, mas sim em um intervalo infinito de tempo, surgindo então a transformada de Fourier. Dessa forma, a única diferença entre série de Fourier (Equação 8) e transformada de Fourier, definida como:

$$c_k = \int_{-\infty}^{\infty} f(t) e^{-ik\omega t} \mathrm{d}t, \tag{9}$$

é que a primeira é aplicada para funções periódicas e a outra (Equação 9) para funções não periódicas, além da transformada ser geralmente dada em função dos coeficientes, e a série em função de $f(t)$.

Na prática, como só podemos processar problemas finitos, isto é, $f(t)$ é representado como um vetor $\mathbf{x} = \{x_0, \cdots, x_N\}$ com N de observações, o que fazemos é restringir a função para o intervalo de interesse N e assumimos que fora deste intervalo $f(t) = 0$. Neste cenário temos a transformada discreta de Fourier:

$$c_k = \frac{\epsilon}{N} \sum_{k=1}^{N} x_k e^{-ik\omega t}, \quad \epsilon = \begin{cases} 1, & k = 0, \\ 2, & k > 0, \end{cases} \tag{10}$$

de tal forma que cada componente c_k é utilizado para recompor o sinal original *sem perda de energia* na forma:

$$x_k = \sum_{k=1}^{N} c_k e^{ik\omega t}. \tag{11}$$

A fim de exemplificar, considere o sinal dado por $f(t) = 2 + \frac{1}{2}\sin(3\omega t + \frac{\pi}{2}) + \frac{1}{4}\sin(10\omega t + 3\pi)$, em que $T = 6.28$, $\omega = 2\pi/T$ e t é uma sequência de $N = 1000$ elementos igualmente distribuídos no intervalo $[0, T]$, conforme ilustrado na Figura 1.

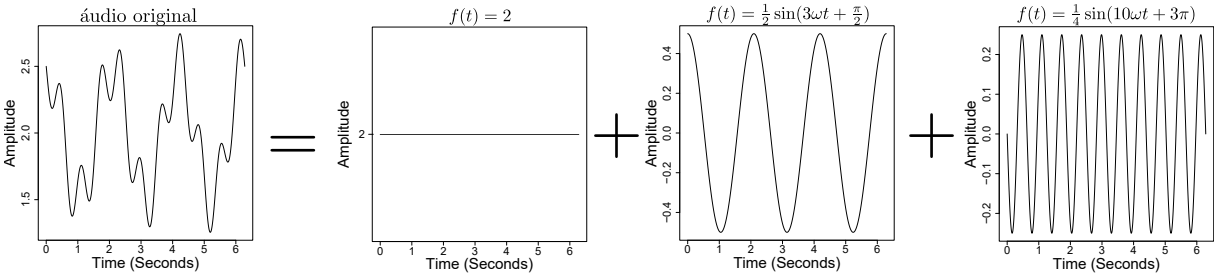

Figura 1: Sinal representado por somas de senoides.

Após aplicarmos a transformada de Fourier, podemos reconstruir os principais componentes do sinal (em ordem de amplitude) da seguinte forma: Note que apesar de haver predominan-

Frequência k	Amplitude c_k	Fase θ_k
0	2.000500	0.000000
3	0.500243	90.558849
10	0.249834	181.822723
11	0.002648	0.237650
$\vdots$	$\vdots$	$\vdots$

Tabela 1: Principais coeficientes do sinal $f(t)$ por ordem descendente de amplitude.

temente três senoides neste sinal (contando o termo constante), ainda existem outras senoides que possuem uma certa influência, ainda que pequena, na composição do sinal. No processo de filtragem do sinal, estes demais componentes são geralmente descartados.

Obs: Os coeficientes da transformada de Fourier são espelhados no sentido que se $c_k = a + ib$ então $c_{N-k} = a - ib$. Logo, geralmente utilizasse somente metade dos coeficientes para recompor os componentes mais relevantes do sinal, uma vez que as magnitudes da primeira metade do vetor de coeficientes são iguais as magnitudes da segunda metade.

4 Aplicação: Remoção de Ruídos de Sinais de Áudio

Considerando um arquivo de áudio gravado em `unsigned char`, ou seja, cada valor de variação do diafragma do microfone é dado por um inteiro de 8 bits sem sinal:

1. O arquivo de entrada o conterá os seguintes dados:

   ```
   nome
   C
   ```

2. Abra o arquivo com nome de arquivo `nome` em modo leitura de arquivo binário. Leia todo seu conteúdo e armazene em um vetor dinamicamente alocado (memória *heap*). Este arquivo contém um sinal formado por N observações. Entenda por observação cada inteiro de 8 bits sem sinal contido no arquivo de áudio.

3. Em seguida, considere esse vetor como entrada para a transformada discreta de Fourier, ou seja, esse vetor definirá o conjunto $\mathbf{x} = \{x_0, x_1, \cdots, x_k\}$.

4. O número inteiro $N/2 \leq C \leq N$ representa o número de componentes mais relevantes (maiores amplitudes) que devem permanecer no sinal resultante. Os demais $N - C$ componentes serão descartados, já que não são tão relevantes e podem conter possíveis ruídos.

5. Aplique a transformada gerando todos os N coeficientes. Dessa maneira, serão produzidos os seguintes coeficientes após aplicar a transformada: $c_0, c_1, \cdots, c_N$. Armazene todos esses coeficientes em um vetor dinamicamente alocado.

6. Calcule a magnitude (sinônimo de amplitude) para esses coeficientes. Em seguida ordene o vetor de coeficientes de maneira decrescente segundo suas magnitudes, ou seja, o coeficiente de maior magnitude será o primeiro do vetor, em seguida o segundo de maior magnitude e assim sucessivamente. Será preciso, no entanto, guardar qual é a posição original de cada coeficiente no vetor $c_0, c_1, \cdots, c_N$.

7. Remova os componentes menos importantes, atribuindo zero aos coeficientes das posições maiores do que C. Devem ser mantidos no vetor de coeficientes somente os valores dos C primeiros coeficientes (eles são os mais relevantes).

8. Em seguida, volte os coeficientes para suas posições originais, ou seja, para as posições que eles tinham antes da ordenação.

9. Aplique a transformada inversa de Fourier utilizando os coeficientes modificados. Note que a inversa poderá gerar valores de ponto flutuante, que deverão, na hora da escrita, serem convertidos para inteiros de 8 bits sem sinal (`unsigned char`).

10. Salve o resultado dessa transformada inversa em um arquivo binário de áudio em que as amplitudes do diafragma são formadas por 8 bits sem sinal. Este sinal será melhor que o sinal original no sentido que possíveis ruídos (barulhos de fundo, ecos, etc) que não são preponderantes no áudio serão descartados. Cabe ressaltar que vários codecs de áudio utilizados em celulares e Voip usam praticamente este mesmo método (ex: G.721).

11. O programa deverá mostrar no arquivo de saída:

    - o número de observações lidas do arquivo de áudio;
    - o número de magnitudes dos coeficientes maiores que 0.1, isto é, o número de coeficientes relevantes no áudio original;
    - a sequência ordenada das magnitudes dos coeficientes das posições de 1 a C, convertidas para um inteiro usando arredondamento com `cast` (truncando casas depois da vírgula).

12. Como é inviável exemplificar casos binários, suponha que a série descrita na Figura 1 seja o sinal de áudio fornecido como arquivo binário e que o usuário especifique $C = 5$. Sabendo que o símbolo ␣ significa um espaço em branco, o seu arquivo de saída deverá respeitar o seguinte formato:

    ```
    1000\n
    997\n
    2_0_0_0_0\n
    ```

    Note que as amplitudes 0.5 e 0.25 foram convertidas para 0 na conversão para inteiro com truncamento por *casting*.

5 Observações importantes

- Programe as impressões na tela EXATAMENTE como exemplificado no decorrer deste documento. Tome cuidado com pulos de linha, tabs, espaços, etc.
- Note que a transformada inversa de Fourier não é necessária para gerar a saída do trabalho. No entanto, será conferido se a inversa foi implementada, assim se o arquivo filtrado foi gerado corretamente.
- Coloque dentro do zip todos os arquivos de código (`*.h *.c`), o makefile e um arquivo texto com o nome e número USP.