TRABALHO DE CONCLUSÃO DE CURSO

TÉCNICA DE SEPARAÇÃO CEGA DE BAIXA COMPLEXIDADE PARA SOLUÇÃO DO PROBLEMA DE PERMUTAÇÃO EM MISTURAS CONVOLUTIVAS DE SINAIS DE FALA

Pedro Fernando Cavalcante Lima

Brasília, julho de 2016

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

TRABALHO DE CONCLUSÃO DE CURSO

TÉCNICA DE SEPARAÇÃO CEGA DE BAIXA COMPLEXIDADE PARA SOLUÇÃO DO PROBLEMA DE PERMUTAÇÃO EM MISTURAS CONVOLUTIVAS DE SINAIS DE FALA

Pedro Fernando Cavalcante Lima

Trabalho de Conclusão de Curso submetido ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Engenheiro Eletricista

Banca Examinadora

Prof. João Paulo C. L. da Costa, Dr. ENE/UnB, EMS/Fraunhofer IIS e EMS/TU Ilmenau _____
Orientador

Prof. Ricardo Zelenovsky, Dr. ENE / UnB _____
Coorientador

Prof. Leonardo R. A. X. Menezes, Dr. ENE / UnB _____
Examinador interno

Ricardo Kehrle Miranda, MSc. ENE / UnB _____
Examinador interno

FICHA CATALOGRÁFICA

LIMA, PEDRO FERNANDO CAVALCANTE
TÉCNICA DE SEPARAÇÃO CEGA DE BAIXA COMPLEXIDADE PARA SOLUÇÃO DO PROBLEMA DE PERMUTAÇÃO EM MISTURAS CONVOLUTIVAS DE SINAIS DE FALA [Distrito Federal] 2016.
xvi, 74 p., 210 x 297 mm (ENE/FT/UnB, Engenheiro, Engenharia Elétrica, 2016).
Trabalho de Conclusão de Curso - Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Elétrica

1. Separação cega de fontes sonoras	2. Ambiguidade de permutação
3. Misturas convolutivas	4. Separação de sinais de fala
I. ENE/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

LIMA, P. F. C. (2016). *TÉCNICA DE SEPARAÇÃO CEGA DE BAIXA COMPLEXIDADE PARA SOLUÇÃO DO PROBLEMA DE PERMUTAÇÃO EM MISTURAS CONVOLUTIVAS DE SINAIS DE FALA*. Trabalho de Conclusão de Curso, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 74 p.

CESSÃO DE DIREITOS

Pedro Fernando Cavalcante Lima
QRSW 7, bloco A-9, apartamento 304
Setor Sudoeste
CEP 70675-709 - Brasília - DF - Brasil
pfcl@msn.com

Dedicatória

Dedico este trabalho com carinho a meus pais Vera e Pedro e a minha amada esposa Monyze.

Pedro Fernando Cavalcante Lima

Agradecimentos

À minha maravilhosa e amada esposa Monyze, que sempre esteve ao meu lado durante esta longa caminhada, pelo seu amor e carinho imensuráveis. Ao meu orientador professor João Paulo pela dedicação e disponibilidade, ultrapassando a barreira da distância. Ao meu co-orientador Ricardo Kehrle pelas incessantes prontidão, paciência e cuidado minucioso. Ao meu co-orientador professor Ricardo Zelenovsky pela disponibilidade e atenção. À minha família pelo amor incondicional e carinho. Em especial à minha mãe Vera, ao meu Pai Pedro e à minha avó de coração Agar. Aos meus grandes amigos de caminhada, Fernando e Ronald, que em muitas madrugadas me mostraram a força da sinergia. À Stefan Feistel, pela motivação e todo apoio durante esta jornada. À todos que fizeram parte da minha formação, muito obrigado.

Pedro Fernando Cavalcante Lima

RESUMO

Arranjos de microfones podem ser incorporados em diversos dispositivos, desde aparelhos auditivos e instrumentos de gravação bioacústicos até equipamentos para teleconferência e gravadores de áudio forenses, para atenuar a interferência de sons indesejados. A separação de misturas de sinais de fala pode ser realizada no domínio da frequência independentemente para cada componente de frequência. Entretanto, para combinar os sinais separados de cada componente de frequência, a ambiguidade de permutação deve ser resolvida. A técnica no estado da arte depende da computação iterativa de dois conjuntos de dispersões das diferenças entre os perfis das fontes. Neste trabalho, é proposta uma técnica de baixa complexidade para solução da ambiguidade de permutação, com acurácia similar ao estado da arte e baseada em apenas um conjunto de dispersões.

ABSTRACT

Microphone arrays can be incorporated in several devices ranging from hearing aids and bioacoustic recording equipment to teleconference phones and forensic sound recorders in order to attenuate the interference of unwanted sounds. The separation of speech mixtures can be performed on the frequency domain independently for each frequency component. However, in order to combine the separated signals of each frequency component, the permutation ambiguity should be solved. The state-of-the-art technique relies on the iterative computation of two sets of dispersions of the differences between the source profiles. In this work, it is proposed a low complexity solution for the permutation ambiguity with similar accuracy to the state-of-the-art, based on only one dispersions set.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS, NOMENCLATURA E ABREVIAÇÕES

J	Número de microfones
I	Número de fontes sonoras
e	Número de Euler
n	Amostra no tempo
N_x	Número total de amostras em um trecho de áudio gravado
$x_j(n)$	Sinal gravado pelo j-ésimo microfone
$\mathbf{x}(n)$	Vetor contendo os sinais $x_j(n)$
$s_i(n)$	Sinal emitido pela i-ésima fonte sonora
$\mathbf{s}(n)$	Vetor com os sinais da fontes
$h_{ij}(l)$	Resposta ao impulso de uma sala entre fonte i e microfone j
$\mathbf{H}$	Matriz dos canais de mistura contendo os vetores h_{ij}
l	Deslocamento no tempo para um filtro
$*$	Operador para cálculo da convolução
L	Comprimento do filtro (número de *taps*)
$\mathbf{W}$	Matriz inversa dos canais de mistura contendo os vetores w_{ij} com os coeficientes dos filtros
K	Número de *taps* da matriz inversa dos canais de mistura
$y_i(n)$	Sinal na i-ésima saída do sistema de separação
$\mathbf{y}(n)$	Vetor com os sinais ideais de saída so sistema de separação
B	Número de blocos para segmentação de $x_j(n)$ no tempo
N	Comprimento (número de amostras) de cada bloco para segmentação de $x_j(n)$
$\lfloor \cdot \rfloor$	Operador para cálculo da função piso
b	Índice para o número do bloco
$\mathbf{T}(f)$	Matriz quadrada cujas colunas são iguais entre si e são obtidas pela f-ésima coluna da matriz DFT Q
$\mathbf{Q}$	Matriz DFT
i	Número imaginário
F	Número de componentes de frequência
$\|\cdot\|$	Operador para cálculo da norma Euclidiana
m	Número de blocos no tempo para estimação da matriz espectral variante no tempo $\hat{\mathbf{S}}_{\mathbf{x}}(b, f)$

$\mathbf{D}(f)$	Matriz diagonal
$\mathbf{\Pi}(f)$	Matriz de permutação com um único 1 por linha e zeros em todo resto
π_{ij}	Elemento da i-ésima linha e j-ésima coluna de $\mathbf{\Pi}(f)$
I_I	Matriz identidade de dimensões $I \times I$
!	Operador para cálculo do fatorial
$\max(\cdot)$	Valor máximo em um conjunto de valores possíveis de seu argumento
$\min(\cdot)$	Valor mínimo em um conjunto de valores possíveis de seu argumento
$\lvert \cdot \rvert$	Operador para cálculo do valor absoluto
arg	Operador que retorna o índice de um elemento fornecido
log	Logaritmo de base 10
ln	Logaritmo natural
$\mathrm{diag}(\cdot)$	Operador que extrai os elementos da diagonal da matriz fornecida em seu argumento em um vetor
$\triangleq$	Igual por definição
M	Parâmetro de largura de banda de suavização, onde a largura de banda total é dada por $2M + 1$
σ^2	Variância de uma variável aleatória
T_h	Valor de limiar
Z	Número de zonas de busca, ou zonas de frequências. Também, o número de permutações encontradas em uma iteração
n_f	Número de componentes de frequência

Símbolos Gregos

ε	Fração muito pequena de uma certa grandeza
α	Sobreposição entre blocos no tempo
π	Representa a permutação entre duas fontes

Sobrescritos

$\bar{\cdot}$	Valor médio
$\tilde{\cdot}$	Mediana
$\cdot^T$	Transposto
$\hat{\cdot}$	Estimado
$\cdot^{-1}$	Operador para cálculo da matriz inversa
$\cdot^H$	Transposto conjugado
$\cdot^f$	Índice do componente de frequência
$\cdot \times \cdot$	Dimensões do vetor ou matriz
$\cdot \times \cdot \times \cdot$	Dimensões do tensor

Subescritos

i_t	Índice que indica o número da iteração

Siglas e Abreviações

SOS	Estatísticas de segunda ordem - do inglês, *Second Order Statistics*
ETC	Curva energia-tempo - do inglês, *Energy-Time Curve*
DFT	Transformada discreta de Fourier
IDFT	Transformada discreta de Fourier inversa
AJD	Diagonalização conjunta aproxima - do inglês *Approximate Joint Diagonalization*
STFT	Transformada de Fourier de tempo curto
BSS	Separação cega de fontes - do inglês, *Blind Source Separation*

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

O termo separação cega de fontes - do inglês, *Blind Source Separation* (BSS) - se refere, em geral, à estimação de um conjunto de sinais, após eles terem sido misturados por canais desconhecidos e capturados por um arranjo de sensores. Este problema multisensorial tem sido explorado em diferentes áreas da ciência utilizando vários tipos de sensores, abrangendo desde aplicações de engenharia biomédica, como arranjos de eletrodos na encefalografia, até telecomunicações com o uso de arranjos de antenas. Neste trabalho, foco é dado a sinais de fala gravados em uma ambiente acústico como, por exemplo, salas de reunião e cafeterias, por um arranjo de microfones. Neste cenário, cada microfone recebe tanto os sinais das fontes quanto sinais provenientes de reflexões no ambiente, que são cópias atrasadas e filtradas dos sinais originais, caracterizando um problema de separação de misturas convolutivas.

Técnicas de separação cega de fontes sonoras são utilizadas para resolver diferentes problemas, como na assistência a deficiências auditivas através aparelhos auditivos (PEDERSEN, 2006). A Fig. 1.1 mostra um cenário onde dispositivos auditivos baseados em BSS podem ser empregados para realçar o discurso desejado e atenuar sons provenientes de outras pessoas conversando ao mesmo tempo.

Figura 1.1: Exemplo de cenário onde dispositivos auditivos baseados em BSS podem ser empregados.[1]

Ténicas de BSS também podem ser usadas no reconhecimento de locutores em aplicações forenses ou de segurança da informação - onde a separação do discurso principal de fontes secundárias é o primeiro passo para permitir a identificação do locutor por técnicas de reconhecimento, como as propostas por (SILVEIRA et al., 2013) e (DENK; COSTA; SILVEIRA, 2014). O dia-

[1]Fonte: modificado de <http://www.steeldart.org/dartforum/dartsport/oesterreich/meet-new-people-online>.

grama mostrado na Fig. 1.2 mostra o fluxo dos processos em aplicações como essa. Note que, após captura dos sinais das fontes emissoras por um arranjo de microfones, sejam elas vários locutores ou também sinais de máquinas e ruídos externos, o algoritmo de separação é implementado para remoção de sinais interferentes, antes da identificação do locutor. Outras aplicações incluem maquinários acionados por comandos de voz e realce de discursos e melhoria da inteligibilidade em teleconferências (SCHOBBEN, 2001).

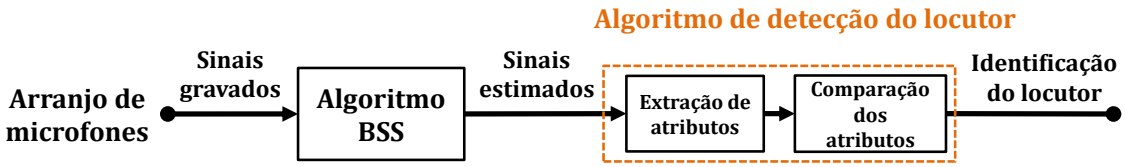

Figura 1.2: Diagrama de blocos exemplificando a aplicação de técnicas de BSS para melhoria de algoritmos de identificação de locutores

1.2 INTRODUÇÃO AO PROBLEMA E OBJETIVO DO TRABALHO

Dentre os métodos para resolução do problema de separação, encontram-se técnicas que exploram a não estacionariadade típica de sinais de fala supondo que os sinais das fontes são não correlacionados, como (NION et al., 2010), (PHAM; SERVIÈRE; BOUMARAF, 2003c) e (PARRA; SPENCE, 2000). Dependendo, assim, de estatísticas de segunda ordem - do inglês, Second Order Statistics (SOS). Para diminuição do custo computacional, o problema de separação de misturas convolutivas é mapeado para o domínio da frequência, onde é tratado como problemas disjuntos de separação de misturas instantâneas para cada componente de frequência. Entretanto, este método tem um custo: quando combinadas, as soluções independentes para cada componente de frequência formam uma solução global apenas até ambiguidades de permutação e escala. Isto é, não há como garantir que as fontes em um componente de frequência dos sinais estimados estejam alinhadas e tenham a mesma escala entre componentes de frequência vizinhos. Essas permutações entre fontes e mudanças de escala acontecem de forma aleatória em função da frequência.

A ambiguidade de permutação dependente da frequência ainda é uma problema considerável em aplicações de áudio do mundo real. Em geral, as soluções podem ser agrupadas em duas categorias principais (PEDERSEN et al., 2007): técnicas baseadas na consistência dos coeficientes dos filtros de mistura, que são estimados como parte da solução; e métodos baseados na consistência de representações em tempo-frequência dos sinais das fontes estimados. Em (SERVIÈRE; PHAM, 2006) ambas perspectivas são exploradas em uma abordagem em dois estágios. O primeiro estágio realiza uma correção preliminar supondo que as respostas de frequência dos filtros de mistura são contínuas, como originalmente proposto por (PHAM; SERVIÈRE; BOUMARAF, 2003b). Para resolver as permutações remanescentes, o segundo estágio proposto por (SERVIÈRE; PHAM, 2006) é empregado supondo que os perfis das fontes estimadas, que são representações em tempo-frequência, variam com a frequência de forma suave o suficiente. Este

estágio é iterativo, requerendo uma iteração por permutação a ser corrigida, e depende da computação de dois conjuntos de dados de dispersão.

Neste trabalho, é proposto um método parcialmente iterativo como alternativa ao segundo estágio proposto por (SERVIÈRE; PHAM, 2006), que utiliza apenas um conjunto de dispersões para corrigir as permutações remanescentes do primeiro estágio da solução para o problema de ambiguidade de permutação. Espera-se, dessa forma, diminuir a complexidade computacional obtendo resultados similares à técnica original. O método proposto é comparado em termos de tempo de cálculo e percentual de sucesso com as seguintes soluções no estado da arte: (SERVIÈRE; PHAM, 2006); e (PHAM; SERVIÈRE; BOUMARAF, 2003c) com modificações propostas por (NION et al., 2010) para computação dos "centroides" dos perfis das fontes baseada em um algoritmo de agrupamento do tipo *k-means*.

1.3 ORGANIZAÇÃO DO MANUSCRITO

Este trabalho é organizado da seguinte forma. No capítulo 2, o problema da separação cega de misturas convolutivas é introduzido, explicando o significado do termo misturas convolutivas e definindo os processos do sistema de separação de forma detalhada. Além disso, os problemas de ambiguidade de escala e permutação, que precisam ser resolvidos antes que os sinais das fontes sejam efetivamente separados, são elencados. No capítulo 3, soluções para a correção da ambiguidade da permutação de acordo com o proposto por (PHAM; SERVIÈRE; BOUMARAF, 2003b) e (SERVIÈRE; PHAM, 2006) são descritas e um novo método baseado em (SERVIÈRE; PHAM, 2006), é proposto. Simulações e resultados experimentais considerando ainda um terceiro método no estado da arte, (PHAM; SERVIÈRE; BOUMARAF, 2003c) com modificações por (NION et al., 2010), são mostrados no capítulo 4. Finalmente, as conclusões do trabalho são encontradas no capítulo 5.

2 SEPARAÇÃO CEGA DE MISTURAS CONVOLUTIVAS DE SINAIS DE ÁUDIO

Neste capítulo os processos que envolvem o problema da separação cega de misturas convolutivas são descritos. Primeiramente, na seção 2.1 o problema é apresentado e o significado do termo misturas convolutivas é esclarecido considerando sinais de áudio emitidos em um ambiente genérico. Em seguida, na seção 2.2, os subsistemas que compõem o sistema de separação são especificados em detalhes. Finalmente, na seção 2.3, os problemas de ambiguidade de escala e permutação, que são parte do sistema de separação, são explicados.

2.1 MISTURAS CONVOLUTIVAS

O primeiro passo na resolução do problema de separação cega é a gravação dos sinais das fontes misturados em um ambiente acústico. Essas fontes são em geral interlocutores, pessoas conversando em um recinto genérico onde os sons emitidos por elas são frequentemente mascarados pela interferência de sons secundários. Estes sons interferentes podem ser emitidos tanto por outros locutores quanto por máquinas ou fontes externas. Entretanto, tendo em vista as técnicas utilizadas aqui, especialmente no que tange à correção do assim chamado problema da ambiguidade de permutação, consideramos que os sons interferentes são causados por outros locutores, como ilustrado na Fig. 2.1. A aquisição é realizada utilizando-se um arranjo de microfones idênticos, dispostos em uma linha no espaço e com espaçamento uniforme. Note que cada microfone recebe os sinais superpostos das fontes e também as respectivas reflexões no ambiente e em objetos nele inseridos. Devido ao tempo que a onda sonora leva para se propagar no ar e à absorção de sua energia acústica (em função da frequência) ao incidir sobre uma superfície, uma reflexão é uma cópia atrasada e filtrada do sinal de uma fonte. Cada microfone capta inúmeras dessas cópias com o tempo até que a energia delas seja mínima.

A superposição dos sinais e suas reflexões pode ser matematicamente descrita como uma convolução entre as respostas ao impulso do ambiente acústico e os sinais das fontes. O objetivo do sistema de separação, descrito no capítulo 2.2, é a obtenção dos sinais originalmente transmitidos por meio de deconvolução e separação dos sinais misturados de forma convolutiva.

Considerando que os sinais misturados $x_j(n),\ j = 1,\ \ldots\ , J$, são capturados por J microfones, onde $n = 1,\ \ldots\ , N_x$ é um índice de amostra no tempo e N_x é o número de amostras em um trecho de áudio gravado, o vetor dos sinais observados é então denotado por:

$$\mathbf{x}(n) = [x_1(n)\ \ x_2(n)\ \ \ldots\ \ x_J(n)]^{\mathrm{T}} \in \mathbb{R}^{J\times 1}, \tag{2.1}$$

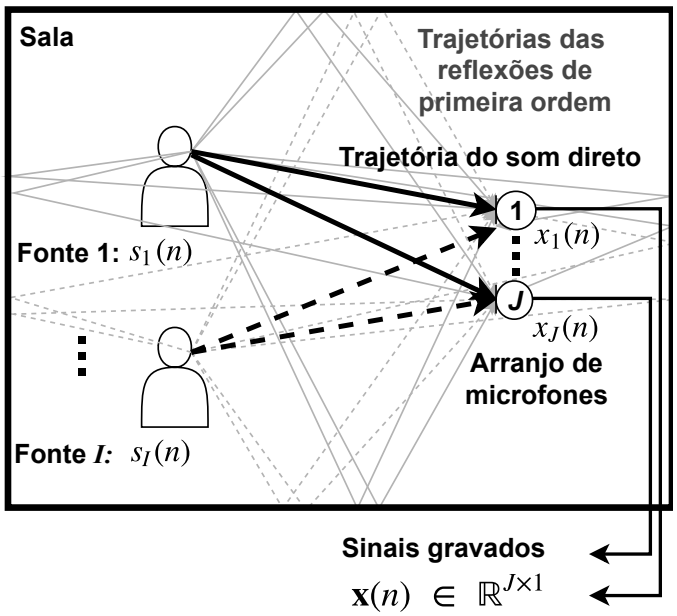

Figura 2.1: Diagrama simplificado da gravação de fontes sonoras em uma sala, ilustrando a geração de misturas convolutivas. Os sinais gravados são entradas para o sistema de separação.

onde o operador $(\cdot)^{\mathrm{T}}$ indica o transposto. De forma análoga, sendo o sinal da i-ésima fonte designado por $s_i(n),\ i = 1,\ \ldots\ , I$, o vetor dos sinais das fontes é dado por:

$$\mathbf{s}(n) = [s_1(n)\ \ s_2(n)\ \ \ldots\ \ s_I(n)]^{\mathrm{T}} \in \mathbb{R}^{I\times 1}. \tag{2.2}$$

Neste trabalho, consideramos que o número de fontes I é conhecido. Entretanto, na prática, técnicas de seleção da ordem do modelo podem ser aplicadas em cada componente de frequência de acordo com (WAX; KAILATH, 1985) e (QUINLAN et al., 2007) para estimar o número de fontes. Os sinais gravados $x_j(n)$ estão relacionados com os sinais da fontes $s_I(n)$ da seguinte maneira:

$$x_j(n) = \sum_{i=1}^{I}\sum_{l=1}^{L} h_{ji}(l)s_i(n-l) = \sum_{i=1}^{I} h_{ji}(l) * s_i(n), \tag{2.3}$$

onde $h_{ji}(l)$ é a resposta ao impulso da sala entre a fonte i e o microfone j, modelando a geometria do ambiente acústico e sua influência sobre o sinal, o operador $*$ denota a convolução e $l = 1,\ \ldots\ , L$ representa o deslocamento no tempo discreto para operação da convolução (MIRANDA, 2013).

Note que supomos um comprimento L comum entre as respostas ao impulso $h_{ji}(l)$. Se os microfones encontram-se próximos o suficiente, de modo que estão incluídos no mesmo ambiente acústico, e os filtros são longos o suficiente, de forma que a respostas ao impulso compreendam tanto as reflexões iniciais quanto as reflexões tardias e com baixa energia, podemos inferir que as respostas ao impulso h_{ji} podem ser modeladas tendo o mesmo comprimento L.

A Fig. 2.2 mostra exemplo da resposta ao impulso de uma sala, a partir da posição de uma fonte até a posição de um microfone. Quando uma fonte sonora está ligada tempo o suficiente em um ambiente, de modo que a densidade de energia acústica encontra-se em regime permanente, o tempo que leva para o nível de pressão sonora ser atenuado em 60 dB, após a fonte ter sido desligada, é chamado tempo de reverberação (KINSLER et al., 2010). Dependendo do volume da sala, sua geometria e características de absorção sonora (que variam de acordo com os materiais das quais são constituídas as superfícies), o tempo de reverberação pode variar de alguns milissegundos a poucos segundos, o que pode corresponder a largos filtros mesmo sob reduzidas taxas de amostragem. Supondo que a sala não possua reflexões discretas e tardias com alta intensidade (ecos), o tempo de reverberação nos fornece um referencial seguro para o comprimento dos canais. Se o L for muito curto, podemos inferir que reflexões com energia considerável não são modeladas e os sinais provindos dessas reflexões não são removidos pelo sistema de separação, o que diminui a relação sinal-ruído obtida no final do processo. Assim, L depende não apenas de quão viva é sala onde se encontra o arranjo de microfones, mas também do critério de qualidade para o sistema de separação.

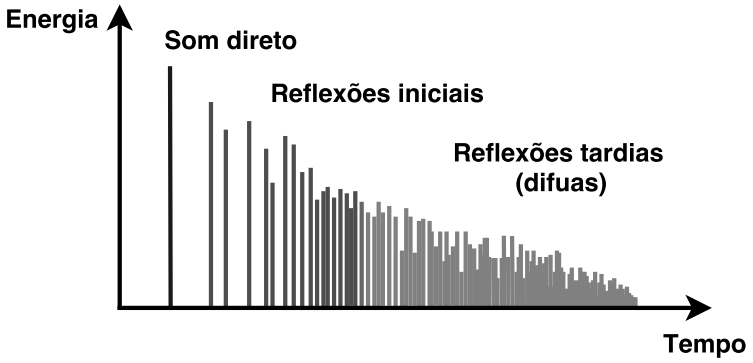

Figura 2.2: Exemplo de curva energia-tempo - do inglês, *Energy-Time Curve* (ETC): logaritmo de base 10 da response ao impulso da sala ao quadrado.

Considerando um comprimento comum para os canais, podemos reescrever a Eq. (2.3) na forma matricial, obtendo o modelo convolutivo livre de ruído para os sinais superpostos:

$$\mathbf{x}(n) = \mathbf{H}(l) * \mathbf{s}(n), \tag{2.4}$$

onde $\mathbf{H}(l) \in \mathbb{R}^{J \times I}$ é a matriz dos canais de mistura, com elementos $h_{ji}(l)$, e $l = 1, \ \ldots \ , L$.

Na prática, os canais de mistura são definidos não apenas pelo ambiente acústico, mas também incluem os efeitos de outros fenômenos. Entre eles, a filtragem dos sinais captados pelos microfones, devidos às suas respostas não ideais. Além disso, ocorre difração das ondas sonoras em torno da cabeça e dos ombros de uma fonte humana, o que dita uma atenuação do sinal de fala (para cada componente de frequência) em função da direção de propagação. Entretanto, o ambiente acústico exerce uma influência mais crítica, já que os filtros que o modelam podem ter comprimentos consideravelmente longos.

2.2 DESCRIÇÃO DO SISTEMA DE SEPARAÇÃO

Para recuperar os sinais das fontes, o objetivo do sistema de separação é encontrar a matriz inversa dos canais de mistura $\mathbf{W}(k) \in \mathbb{R}^{I \times J}$, de forma que:

$$\mathbf{y}(n) = \sum_{k=1}^{K} \mathbf{W}(k)\mathbf{x}(n-k) = \mathbf{W}(k) * \mathbf{x}(n), \tag{2.5}$$

onde $\mathbf{y}(n) = [y_1(n) \;\; y_2(n) \;\; \ldots \;\; y_I(n)]^{\mathrm{T}} \in \mathbb{R}^{I \times 1}$ denota os sinais de saída (ou recuperados) e $k = 1, \ldots K$, onde K é o comprimento do filtro $\mathbf{W}(k)$. Neste trabalho, é considerado que K é conhecido. Entretanto, técnicas de seleção da ordem do modelo também podem ser utilizadas para estimar o número de *taps* de cada filtro $w_{ij}(k)$. Na prática, primeiro estimamos a matriz dos canais de mistura no domínio da frequência $\hat{\mathbf{H}}(f) \in \mathbb{C}^{J \times I}$. Se o número de fontes I é igual ao número de microfones J, a matriz inversa dos canais de mistura estimada $\hat{\mathbf{W}}(f) \in \mathbb{C}^{I \times J}$ pode ser calculada por:

$$\hat{\mathbf{W}}(f) = \hat{\mathbf{H}}(f)^{-1}. \tag{2.6}$$

Com intuito de estimar $\hat{\mathbf{H}}(f) \in \mathbb{C}^{J \times I}$, para fazer uso das estatísticas de segunda do ordem dos sinais capturados $\mathbf{x}(n)$, é supomos que os sinais das fontes $s_i(n)$ são mutualmente não correlacionados e aproximadamente não estacionários. Ainda, é suposto que as respostas ao impulso dos canais de mistura são invariantes no tempo, i.e. que não há mudanças no ambiente ou nas posições das fontes e dos microfones durante as observações.

A Figura 2.3 mostra um diagrama de blocos simplificado do sistema de separação. De acordo com os blocos 1 e 2, após captura dos sinais $\mathbf{x}(n) \in \mathbb{R}^{J \times 1}$ o problema é mapeado para o domínio da frequência através da aplicação da transformada de Fourier de tempo curto - do inglês, *short time Fourier transform (STFT)*. Em seguida, os canais de mistura são estimados via diagonalização conjunta aproxima, obtendo $\hat{\mathbf{H}}_{\mathrm{ajd}}(f)$. No bloco 4, as correções das ambiguidades de permutação e escala são resolvidas e os sinais são estimados no domínio da frequência. Finalmente, os sinais estimados são mapeados de volta para o domínio do tempo, onde são reconstruídos. Estes processos são detalhados nas seções seguintes.

2.2.1 Transformada de Fourier de tempo curto (STFT)

Após gravação dos sinais $\mathbf{x}(n) \in \mathbb{R}^{J \times 1}$ pelo arranjo linear com J microfones, a STFT é aplicada da seguinte forma. Primeiramente, o vetor dos sinais observados $\mathbf{x}(n)$ é divido em B blocos consecutivos e sobrepostos de tamanho N. O número de blocos B pode ser calculado por:

$$B = \left\lfloor \frac{N_x - N\alpha}{N - N\alpha} \right\rfloor, \tag{2.7}$$

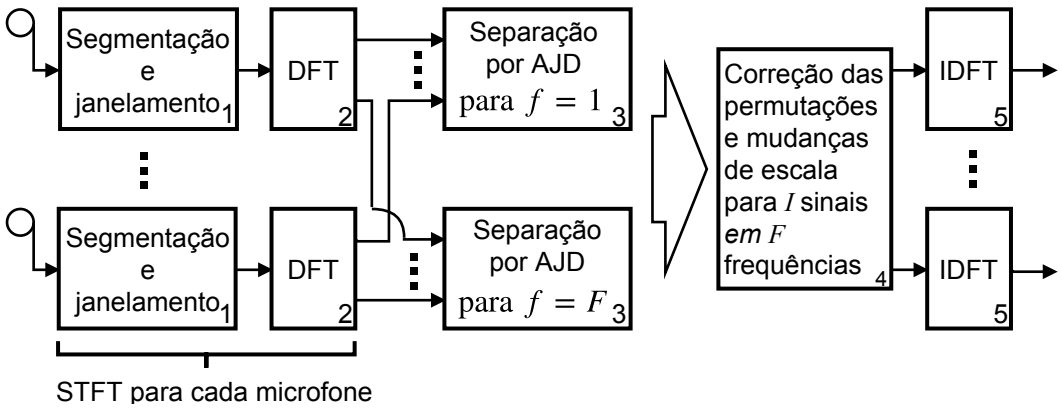

Figura 2.3: Diagrama de blocos simplificado do sistema de separação BSS.

onde $0 \leq \alpha \geq 1$ é a sobreposição escolhida entre os blocos, e.g. $\alpha = 0.75$, de forma que $N\alpha$ fornece o número de amostras sobrepostas entre os dois segmentos. O operador $\lfloor \cdot \rfloor$ representa a função piso, que fornece o primeiro número inteiro menor ou igual a seu argumento. Note que N deve ser escolhido de forma que seja consideravelmente mais largo do que o comprimento L dos canais de mistura $h_{ji}(l)$ (NION et al., 2010). A cada bloco b, é a aplicada a janela de Hann (SERVIÈRE; PHAM, 2006) de comprimento N dada por:

$$\mathrm{Hann}(n) = 1 - \cos\left(\frac{2\pi n + \pi}{N}\right),\ 0 \leq n < N, \tag{2.8}$$

através de multiplicação ponto-a-ponto. O janelamento tem a função de reduzir o vazamento espectral devido à aplicação da transformada discreta de Fourier - do inglês, discrete Fourier transform (DFT) - a sinais não periódicos, como no bloco 2 da Fig. 2.3. Além disso, no bloco 3 da Fig. 2.3, a densidade espectral de potência dos sinais observados é estimada como parte do processo. O janelamento reduz o viés deste estimador (SERVIÈRE; PHAM, 2006). Finalmente, é computada a DFT de cada bloco b resultante do processo de janelamento:

$$\tilde{\mathbf{x}}(b, f) = \tilde{\mathbf{X}}(b)\mathbf{T}(f) \in \mathbb{C}^{J \times 1}, \tag{2.9}$$

onde $f = 1, \ \ldots \ , F = N$, $\mathbf{X} \in \mathbb{R}^{J \times N}$ é a matriz na qual a janela de Hann dada pela Eq. (2.8) é aplicada em cada linha e $\mathbf{T}(f) \in \mathbb{C}^{N \times N}$ é uma matriz quadrada cujas colunas são iguais entre si e são obtidas pela f-ésima coluna da matriz DFT $\mathbf{Q}$. Os elementos de $\mathbf{Q}$ são obtidos por:

$$q_{j,k} = e^{-2\pi ijk/N}, \ \ j, k = 0, 1, \ \ldots \ N - 1, \tag{2.10}$$

sendo $i = \sqrt{-1}$ o número imaginário.

2.2.2 Estimação dos canais de mistura via AJD

Considerando B blocos no tempo e F componentes de frequência, a saída da transformada discreta de Fourier após segmentação e janelamento $\tilde{\mathbf{x}}(b, f)$ é denotada pelo tensor de dados $\tilde{\mathcal{X}} \in \mathbb{C}^{J \times B \times F}$, ilustrado na Fig. 2.4a. Note que o tensor de dados $\tilde{\mathcal{X}}$ tem 3 dimensões: espaço (microfone), tempo e frequência. De acordo com o bloco 3 da Fig. 2.3, o problema da estimação da matriz dos canais de mistura $\hat{\mathbf{H}}(f)$ é resolvido para cada fatia de $\tilde{\mathcal{X}}$ com dimensões $J \times B$, ilustradas na Fig. 2.4b.

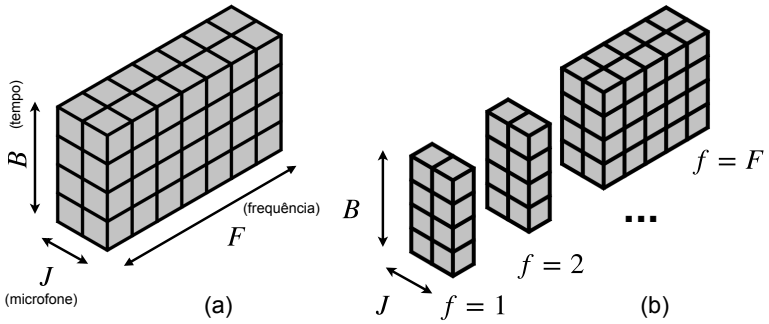

Figura 2.4: Representação do tensor de dados $\tilde{\mathcal{X}} \in \mathbb{C}^{J \times B \times F}$, saída da STFT.

Após aplicação da STFT, procedemos para o bloco 3 da Fig. 2.3, através da estimação da matriz espectral dos sinais gravados $\hat{\mathbf{S}}_x(b, f) \in \mathbb{C}^{J \times J}$. Para isso, $\tilde{\mathbf{x}}(b, f)$ é utilizado para calcular o seguinte periodograma (SERVIÈRE; PHAM, 2006):

$$\mathbf{P}_{\mathbf{x}}(b, f) = \frac{\tilde{\mathbf{x}}(b, f)\tilde{\mathbf{x}}(b, f)^{\mathrm{H}}}{\|\mathrm{Hann}\|^2} \in \mathbb{C}^{J \times J}, \tag{2.11}$$

onde o operador $\|(\cdot)\|$ denota a norma Euclidiana e $(\cdot)^{\mathrm{H}}$ indica o transposto conjugado. O denominador na Eq. (2.11) é um fator de normalização devido à aplicação da janela de Hann antes da DFT. A matriz espectral (variante no tempo) é então estimada calculando-se a média de $\mathbf{P}_{\mathbf{x}}(b, f)$ entre m blocos consecutivos no tempo, de acordo com:

$$\hat{\mathbf{S}}_{\mathbf{x}}(b, f) = \frac{1}{m} \sum_{b_k = b-(m-1)/2}^{b+(m-1)/2} \mathbf{P}_{\mathbf{x}}(b_k, f) \in \mathbb{C}^{J \times J}. \tag{2.12}$$

O j-ésimo elemento da diagonal dessa matriz é uma estimativa da densidade espectral de potência em função do tempo para o j-ésimo sinal gravado, sendo uma representação em tempo-frequência deste. Com o problema mapeado para o domínio da frequência, a convolução se torna uma série de multiplicações e a Eq. (2.4) se torna:

$$\tilde{\mathbf{x}}(b, f) \approx \mathbf{H}(f)\tilde{\mathbf{s}}(b, f), \tag{2.13}$$

onde $\mathbf{H}(f) \in \mathbb{C}^{J \times I}$ é a matriz de respostas de frequência dos canais de mistura. Além disso:

$$\tilde{\mathbf{s}}(b, f) \approx \mathbf{W}(f)\tilde{\mathbf{x}}(b, f), \tag{2.14}$$

onde $\mathbf{W}(f) \in \mathbb{C}^{I \times J}$ é a matriz de respostas de frequência da matriz inversa do canais. Com isso, a equação (2.12) pode ser reescrita como:

$$\hat{\mathbf{S}}_{\mathbf{x}}(b, f) = \mathbf{H}(f)\hat{\mathbf{S}}_{\mathbf{s}}(b, f)\mathbf{H}(f)^{\mathrm{H}}, \tag{2.15}$$

onde $\hat{\mathbf{S}}_{\mathbf{s}}(b, f) \in \mathbb{C}^{I \times I}$ é uma estimativa da matriz espectral variante no tempo dos sinais das fontes (SERVIÈRE; PHAM, 2006). De forma análoga a $\hat{\mathbf{S}}_{\mathbf{x}}(b, f)$, o i-ésimo elemento da diagonal de $\hat{\mathbf{S}}_{\mathbf{s}}(b, f)$ é uma estimativa da densidade espectral de potência em função do tempo para o i-ésimo sinal de fonte.

Supondo que os sinais das fontes são não correlacionados, $\hat{\mathbf{S}}_{\mathbf{s}}(b, f)$ é uma matriz diagonal $\in \mathbb{R}^{I \times I}$. Para cada componente de frequência f, se os sinais das fontes $s_i(n)$ forem aproximadamente não-estacionários entre os B blocos no tempo, podemos explorar essas propriedades em um problema de diagonalização conjunta aproximada - do inglês, *Approximate Joint Diagonalization* (AJD). A partir da Eq. (2.15), para cada componente de frequência f, o problema busca computar uma matriz $\hat{\mathbf{H}}_{\mathrm{ajd}}(f)$ comum entre os B blocos no tempo, tal que $\hat{\mathbf{S}}_{\mathbf{s}}(b, f)$ seja o mais próximo de uma matriz diagonal possível, de acordo com algum critério. A matriz $\hat{\mathbf{H}}_{\mathrm{ajd}}(f) \in \mathbb{C}^{J \times I}$ é uma estimativa da matriz de respostas de frequência dos canais de mistura, após os processos de AJD para os F componentes de frequência. Note que, em geral, sinais de fala são considerados não estacionários para períodos superiores a 40 ms (NION et al., 2010). Logo, a resolução temporal para a STFT deve ser ajustada de forma que os sinais das fontes sejam não estacionários entre blocos consecutivos.

Soluções para o problema da diagonalização conjunta aproximada, considerando aplicações de BSS, foram propostas em e.g. (PHAM; CARDOSO, 2001) e (YEREDOR, 2002). Assim, de acordo com o bloco 3 da Fig. 2.3, após estimação da matriz espectral variante no tempo $\hat{\mathbf{S}}_{\mathbf{x}}(b, f)$, uma técnica de AJD pode ser empregada para se estimar $\hat{\mathbf{H}}_{\mathrm{ajd}}(f)$ e $\hat{\mathbf{S}}_{\mathbf{s}}(b, f)$ a partir do tensor de dados $\hat{\mathcal{S}}_{\mathbf{x}}(f) \in \mathbb{C}^{J \times J \times B}$, para cada componente de frequência f separadamente.

Não obstante, como a partir da técnica de AJD os canais de mistura são estimados para cada componente de frequência f de forma independente, $\hat{\mathbf{H}}_{\mathrm{ajd}}(f)$ é um estimador dos canais de mistura $\mathbf{H}(\mathbf{f})$ apenas até ambiguidades de permutação e mudança de escala, que são descritas no Capítulo 2.3. Após os problemas de ambiguidades de escala e permutação serem solucionadas de acordo com o bloco 4 da Fig. 2.3, os sinais das fontes estimados $\hat{\mathbf{s}}(b, f)$ podem ser obtidos no domínio da frequência através da Eq. (2.14) usando a matriz estimada e com ambiguidades corrigidas $\hat{\mathbf{W}}(f)$. Então, de acordo com o bloco 5 da Fig. 2.3, é aplicado o procedimento inverso da DFT para mapear os sinais $\hat{\mathbf{s}}(b, f)$ de volta para o domínio do tempo. Por fim, os sinais estimados da fontes $\hat{\mathbf{y}}(n)$ são reconstruídos aplicando-se uma operação inversa à segmentação e ao janelamento empregados no bloco 1 na Fig. 2.3, superpondo as B saídas da transformada de acordo com a taxa de sobreposição α escolhida anteriormente (Eq. 2.7).

2.3 AMBIGUIDADES DE ESCALA E PERMUTAÇÃO

Devido às soluções separadas entre os f componentes de frequência de $\tilde{\mathcal{X}} \in \mathbb{C}^{J\times B\times F}$, o método baseado em AJD fornece a matriz dos canais de mistura $\hat{\mathbf{H}}_{\mathrm{ajd}}(f) \in \mathbb{C}^{J\times I}$ apenas até permutações e mudanças de escala de suas colunas. Essas ambiguidades ocorrem de forma arbitrária entre consecutivos componentes de frequência de $\hat{\mathbf{H}}_{\mathrm{ajd}}(f)$. Se, por exemplo, considerarmos que $\hat{\mathbf{H}}_{\mathrm{ajd}}(f) \in \mathbb{C}^{J\times I}$ é um estimador consistente de $\mathbf{H}(f)$, i.e. que a separação pelo o processo de AJD foi perfeita, então:

$$\hat{\mathbf{H}}_{\mathrm{ajd}}(f) = \mathbf{H}(f)\mathbf{D}^{-1}(f)\mathbf{\Pi}^{-1}(f), \tag{2.16}$$

onde $\mathbf{D}^{-1}(f) \in \mathbb{R}^{I\times I}$ é uma matriz diagonal desconhecida, que muda de forma aleatória a escala das colunas de $\mathbf{H}(f)$ ao multiplica-la pela direita. $\mathbf{\Pi}^{-1}(f)$ é uma matriz de permutação de tamanho $I \times I$ desconhecida, composta de um único 1 por linha e zero em todos outros elementos, que permuta de forma aleatória as colunas de $\mathbf{H}(f)$, também ao multiplica-la pela direita. A matriz inversa dos canais também pode ser escrita em termos dessas matrizes. Se, por exemplo, considerarmos estimação perfeita e já que a partir da Eq. (2.6), $\hat{\mathbf{W}}_{\mathrm{ajd}}(f) = \hat{\mathbf{H}}_{\mathrm{ajd}}(f)^{-1}$, então:

$$\hat{\mathbf{W}}_{\mathrm{ajd}}(f) = \mathbf{\Pi}(f)\mathbf{D}(f)\mathbf{W}(f). \tag{2.17}$$

Note que, em termos da matriz inversa dos canais de mistura, $\mathbf{D}(f)$ muda a escala das linhas de $\mathbf{W}(f)$, quando a multiplica pela esquerda, e $\mathbf{\Pi}(f)$ permuta as linhas de $\mathbf{W}(f)$, também ao multiplica-la pela esquerda.

Desta forma, e de acordo com o bloco 4 da Fig. 2.3, após estimação do canal de mistura via AJD, ainda é necessário encontrar as matrizes $\mathbf{D}(f)$ e $\mathbf{\Pi}(f)$ antes de obter os sinais estimados $\tilde{\mathbf{s}}(b, f) \in \mathbb{C}^{I\times 1}$ no domínio da frequência a partir da Eq. (2.14). Essas matrizes são conhecidas como ambiguidades de escala e permutação, respectivamente.

A ambiguidade de escala é considerada um problema menos crítico. A matriz $\mathbf{D}(f)$ filtra os canais de mixagem, e consequentemente os sinais recuperados com estes canais, alterando a magnitude da resposta de frequência de cada um desses de forma aleatória. A questão pode ser solucionada, por exemplo, através da implementação do princípio da distorção mínima - do inglês, Minimum Distorion Principle (MDP), como originalmente proposto em (MATSUOKA, 2002) e revisado (MATSUOKA, 2008). O princípio se baseia no fato de que diferentes canais separadores podem ser escolhidos para o sistema, ocasionando diferentes combinações lineares dos sinais estimados na saída. Isso pode ser visto na Fig. 2.5, onde o diagrama de blocos dos processos de mistura e separação com duas fontes e dois microfones é mostrado. Suponde que a influência de cada canal de mistura não seja relevante na modificação dos sinais das fontes, podemos escolher como saída desejada do sistema de separação, por exemplo o sinal da fonte de número 1 captado pelo microfone 1. Ou, podemos escolher o sinal da fonte 1 captado pelo microfone 2. Em ambas as situações, o sinal da fonte 1 é estimado e separado do sinal interferente. Entretanto, dois sinais

diferentes são obtidos considerando a influência dos respectivos canais. Dessa forma, podemos escolher um separador que forneça a menor distorção dos sinais na saída do sistema devido aos canais de mistura (MATSUOKA, 2008). Com base nesse princípio, a ambiguidade de escala pode ser resolvida de tal que forma que os elementos da diagonal de $\mathbf{D}(f)$ são obtidos por:

$$\text{diag}[\mathbf{D}(f)] = \text{diag}[\mathbf{G}\hat{\mathbf{H}}_{\text{ajd}}(f)], \tag{2.18}$$

sendo os elementos fora da diagonal de $\mathbf{D}(f)$ todos iguais a zero. $\mathbf{G}$ é uma matriz de dimensões $J \times J$, cujos elementos são iguais a $1/J$. Com essa escolha de $\mathbf{G}$ e desconsiderando a ambiguidade de permutação, podemos interpretar o separador escolhido da tal forma que o i-ésimo sinal na saída do sistema é dado pela média dos I sinais das fontes como capturados na posições dos microfones (NION et al., 2010).

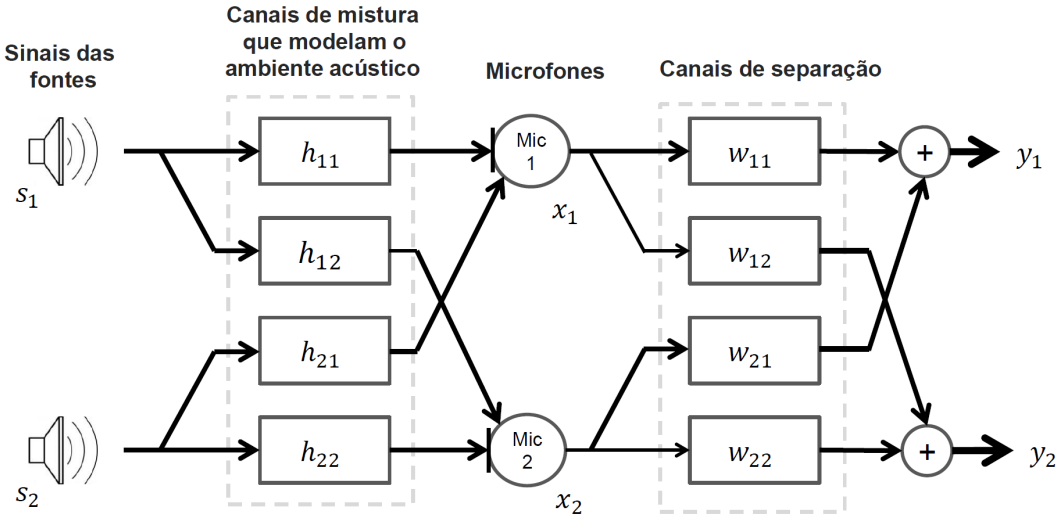

Figura 2.5: Diagrama de blocos mostrando os processos de mistura e separação considerando um sistema com duas fontes e dois microfones.

Já a ambiguidade de permutação $\mathbf{\Pi}(f)$, tem se mostrado um problema mais considerável. Se computarmos os sinais estimados das fontes no domínio da frequência $\hat{\mathbf{s}}(b, f)$ a partir de $\hat{\mathbf{W}}_{\text{ajd}}(f)$ e da Eq. (2.14), devido à ambiguidade de permutação, não há garantias que as fontes em uma fatia com dimensões $I \times B$ do tensor $\hat{\mathcal{S}} \in \mathbb{C}^{I \times B \times F}$ sejam as mesmas em fatias ou frequências vizinhas. Ou seja, em uma determinada saída do sistema $\hat{s}_i(b, f)$, as fontes podem ser alternar entre os diferentes componentes de frequência. Na prática, isso significa que os sinais das fontes ainda estão misturados. No capítulo 3 soluções para o problema da permutação são revisadas e um novo método é proposto.

3 SOLUÇÕES PARA A AMBIGUIDADE DE PERMUTAÇÃO

Neste capítulo, diferentes soluções para o problema da ambiguidade de permutação são apresentados, incluindo o método proposto neste trabalho. Na Seção 3.1, o método proposto em (PHAM; SERVIÈRE; BOUMARAF, 2003b), baseado na resposta de frequência dos canais, é descrito. Por conseguinte, na Seção 3.2, a técnica no estado da arte (SERVIÈRE; PHAM, 2006) é revisada. Trata-se de um sistema de dois estágios que faz uso dos resultados provindos de (PHAM; SERVIÈRE; BOUMARAF, 2003b) para solucionar o problema. Finalmente, na Seção 3.3 descrevemos o método proposto neste trabalho, tendo por base (SERVIÈRE; PHAM, 2006), oferecendo modificações de forma que a performance em termos de tempo de cálculo do método original possa ser melhorada.

3.1 MÉTODO USANDO A RESPOSTA DE FREQUÊNCIA DO CANAL ESTIMADO VIA AJD

Após estimação preliminar da matriz dos canais de mistura $\hat{\mathbf{H}}_{\mathrm{ajd}}(f) \in \mathbb{C}^{J\times I}$, como descrito na Seção 2.2, o problema da ambiguidade de permutação pode ser resolvido de acordo com (PHAM; SERVIÈRE; BOUMARAF, 2003b) como descrito nesta Seção. Como demonstrado pelos autores, o método funciona bem, mas se os canais de mistura contém reflexões fortes, i.e. com alta energia quando comparadas ao som direto da fonte, uma rápida variação ocorre na resposta de frequência e o algoritmo é quebrado, causando um erro na correção da permutação que se espalha por um largo bloco de frequências.

Além disso, em algumas frequências, a matriz $\mathbf{H}(f)$ pode gerar um sistema de equações mal condicionado, i.e. o sinais estimados (em uma frequência f) após inversão de $\mathbf{H}(f)$ não são soluções confiáveis do sistema de equações. Sob essa condição, também ocorre um erro na correção na permutação que pode se estender por um longo bloco de componentes de frequência. Note que um sistema de equações lineares genérico $\mathbf{Ax} = \mathbf{b}$ é considerado bem-condicionado quando uma pequena variação na matriz de coeficientes $\mathbf{A}$, ou uma pequena mudança no vetor de termos constantes $\mathbf{b}$, causa uma pequena variação no vetor de soluções $\mathbf{x}$. Caso contrário, se uma mudança grande ocorrer no vetor de soluções, dizemos que essa matriz é mal-condicionada. Podemos quantificar quão bem condicionada é uma matriz quadrada e inversível $\mathbf{A}$, ou quão confiáveis são as soluções $\mathbf{x}$, através do número de condição da matriz (GOLUB; Van Loan, 1996):

$$\kappa(\mathbf{A}) = \|\mathbf{A}\|\|\mathbf{A}^{-1}\|, \tag{3.1}$$

onde o $\|\cdot\|$ é uma norma matricial consistente e considerando a convenção $\kappa(\mathbf{A}) = \infty$ quando a matriz $\mathbf{A}$ é singular. Utilizando a norma 2 da matriz, adicionamos o subíndice correspondente e obtemos o número de condição de norma 2 como:

$$\kappa_2(\mathbf{A}) = \|\mathbf{A}\|_2 \|\mathbf{A}^{-1}\|_2. \tag{3.2}$$

Se $\kappa_2(\mathbf{A})$ é grande, dizemos que a matriz é mal condicionada. Uma matriz com número de condição pequeno é chamada bem condicionada. Por exemplo, utilizando uma matriz $\mathbf{O}$ ortogonal, o número de condição de norma 2 é igual a: $\kappa_2(\mathbf{O}) = 1$ (GOLUB; Van Loan, 1996).

Dessa forma, esse método de correção da permutação pode ser tanto utilizado por si só para solução do problema de permutação, considerandos o disposto acima, quanto servir como primeiro estágio das técnicas mais robustas que utilizam dois estágios, descritas nas Seções 3.2 e 3.3.

3.1.1 Descrição do algoritmo

Supondo que a resposta de frequência dos canais de mistura $\mathbf{H}(f) \in \mathbb{C}^{J\times I}$ seja contínua, para um valor suficientemente pequeno de ε e em uma frequência f_l, $\mathbf{H}(f_l)$ tende a $\mathbf{H}(f_l - \varepsilon)$. Além disso, $\mathbf{H}(f)$ ser contínua implica a continuidade da resposta de frequência da matriz inversa dos canais $\mathbf{W}(f) = \mathbf{H}(f)^{-1}$. Assim, de forma análoga, $\mathbf{H}(f_l)^{-1}$ tende a $\mathbf{H}(f_l - \varepsilon)^{-1}$. Então, a matriz de razão $\mathbf{R}$ é dada por:

$$\mathbf{R}(f_l, f_l - \varepsilon) = \mathbf{H}(f_l)^{-1}\mathbf{H}(f_l - \varepsilon) = I_I, \tag{3.3}$$

onde I_I é a matriz identidade de dimensão $I \times I$. Na prática, a matriz dos canais de mistura estimada por AJD $\hat{\mathbf{H}}_{\text{ajd}}(f)$ é uma função discreta. Dessa forma, podemos procurar uma matriz de permutação $\mathbf{\Pi}_i(f_l)$ de tal forma que a relação $\mathbf{\Pi}_i(f_l)\hat{\mathbf{R}}(f_l, f_l - 1)$ seja o mais próximo possível da matriz identidade de dimensão $I \times I$, de acordo com o seguinte critério. Para cada componente de frequência f, uma matriz de permutação $\mathbf{\Pi}_i(f),\ i = 1, \ldots, I!$ é procurada entre, em princípio, todas as $I!$ matrizes de permutação possíveis, de tal forma que:

$$i = \underset{i=1,\ldots,I!}{\arg\max} \prod \left(\text{diag}\left[|\mathbf{\Pi}_i(f)\hat{\mathbf{R}}(f, f-1)| \right] \right) \tag{3.4}$$

seja resolvido, onde $\hat{\mathbf{R}}$, estimativa da matriz de razão $\mathbf{R}$, é obtida por:

$$\hat{\mathbf{R}}(f, f-1) = \hat{\mathbf{H}}_{\text{ajd}}(f)^{-1}\hat{\mathbf{H}}_{\text{ajd}}(f-1), \tag{3.5}$$

$(\cdot)!$ indica o fatorial, $\prod(\cdot)$ denota o produtório dos elementos em seu argumento, $\arg\max(\cdot)$ retorna o argumento cujo emprego resulta no máximo valor, $|(\cdot)|$ é o valor absoluto e $\arg(\cdot)$

retorna os índices do (nesse caso, vetor) em seu argumento.

Aplicando a Eq. (3.4), podemos encontrar a matriz $\mathbf{\Pi}(f)$ desejada para cada componente de frequência. Entretanto, como sugerido por (PHAM; SERVIÈRE; BOUMARAF, 2003b), a busca entre as $I!$ matrizes de permutação possíveis na Eq. (3.4) pode ser evitada. Isso acontece quando em um componente frequência f_l o índice $j_{\max}(i)$, da coluna do elemento de maior valor absoluto da i-ésima linha de $\hat{\mathbf{R}}(f_l, f_l - 1)$, é distinto dos índices das colunas que possuem maior valor absoluto nas outras $I - 1$ linhas:

$$j_{\max}(i) = \underset{j=1,\cdots,I}{\arg\max} |\hat{r}_{ij}(f_l, f_l - 1)|,\ i = 1, \cdots, I, \tag{3.6}$$

onde $\hat{r}_{ij}(f_l, f_l - 1)$ é um elemento de $\hat{\mathbf{R}}(f_l, f_l - 1)$ e o operador $\arg(\cdot)$ retorna o índice do elemento fornecido. Neste caso, $\mathbf{\Pi}(f_l)$ é a matriz de permutação que permuta as linhas de $\hat{\mathbf{R}}(f, f_l - 1)$ de tal forma que os elementos de maior valor absoluto encontram-se na diagonal principal da matriz $\mathbf{\Pi}(f_l)\hat{\mathbf{R}}(f, f_l - 1)$. Assim:

$$\pi_{ij}(f_l) = \begin{cases} 1,\ j = j_{\max}(i) \\ 0,\ j \neq j_{\max}(i) \end{cases},\ i, j = 1,\ \ldots\ , I, \tag{3.7}$$

onde $\pi_{ij}(f_l)$ denota um elemento de $\Pi(f_l)$. Note que, como o índice $j_{\max}(i)$ indica a coluna do elemento de máximo valor absoluto da i-ésima linha de $\hat{\mathbf{R}}(f_l, f_l - 1)$, o índice também nos indica qual coluna da i-ésima linha de $\mathbf{\Pi}(f_l)$ será igual a 1. Em todas outras colunas desta i-ésima linha os elementos são iguais a 0. Claro, considerando os casos onde todos os I índices $j_{\max}(i)$ são diferentes entre si, como mencionado no parágrafo anterior.

Após determinação das matrizes de permutação $\mathbf{\Pi}(f)$, $f = 1,\ \ldots\ ,\ F$, a correção do problema da ambiguidade da permutação pode ser finalmente efetuada. Para cada componente de frequência f_l onde uma permutação realmente ocorre, i.e. quando $\mathbf{\Pi}(f) \neq I_I$, permutamos as linhas da matriz inversa dos canais preliminarmente estimada por AJD $\hat{\mathbf{W}}_{\text{ajd}}(f)$, para todas frequências iguais ou superiores a f_l, de acordo com $\mathbf{\Pi}(f_l)$:

$$\hat{\mathbf{W}}(f) = \mathbf{\Pi}(f_l)\hat{\mathbf{W}}_{\text{ajd}}(f), f = f_l,\ \ldots\ ,\ F. \tag{3.8}$$

Como resultado, obtemos a estimativa da resposta de frequência da matriz inversa dos canais $\hat{\mathbf{W}}(f) \in \mathbb{C}$. Esta matriz pode então ser utilizada para obter os sinais de fonte estimados, no domínio da frequência:

$$\hat{\mathbf{s}}(b, f) = \hat{\mathbf{W}}(f)\tilde{\mathbf{x}}(b, f), \tag{3.9}$$

de forma que possam ser mapeados novamente para o domínio do tempo e reconstruídos, como descrito no último parágrafo da Seção 2.2.

Alternativamente, como mencionado no primeiro parágrafo desta Seção, a saída deste processo pode ser utilizada como primeiro estágio de um método mais robusto e de dois estágios para correção do problema de permutação. Neste caso, a partir da Eq. (3.8) podemos nos referir à saída deste processo como $\hat{\mathbf{W}}_{\mathrm{stg1}}(f)$:

$$\hat{\mathbf{W}}_{\mathrm{stg1}}(f) = \mathbf{\Pi}(f_l)\hat{\mathbf{W}}_{\mathrm{ajd}}(f), f = f_l, \ \ldots \ , \ F, \tag{3.10}$$

considerando que permutações remanescentes deste subsistema serão corrigidas em um segundo estágio. O segundo estágio proposto por (SERVIÈRE; PHAM, 2006), é resumido na Seção 3.2. Em seguida, na Seção 3.3, o método proposto neste trabalho para o segundo estágio é descrito.

3.2 TÉCNICA NO ESTADO DA ARTE

Nesta Seção, o método proposto por (SERVIÈRE; PHAM, 2006) para resolver o problema da permutação é descrito. Trata-se de uma técnica que utiliza dois estágios para realizar a correção.

Após estimação preliminar da matriz dos canais de mistura via AJD, como descrito na Seção 2.2, $\hat{\mathbf{H}}_{\mathrm{ajd}}(f) \in \mathbb{C}^{J\times I}$ alimenta o primeiro estágio da solução, que é resolvido como originalmente proposto em (PHAM; SERVIÈRE; BOUMARAF, 2003b) e descrito na Seção 3.1. Neste ponto de partida da solução, as permutações das colunas de $\hat{\mathbf{H}}_{\mathrm{ajd}}(f)$ - e, consequentemente, das linhas de $\hat{\mathbf{W}}_{\mathrm{ajd}}(f) = \hat{\mathbf{H}}^{-1}_{\mathrm{ajd}}(f)$ - ocorrem de forma aleatória entre consecutivos componentes de frequência. Este comportamento é mostrado em um exemplo na Fig. 3.1. As curvas laranja e azul mostram a magnitude da resposta de frequência de $w_{11}(f)$ e $w_{21}(f)$, respectivamente, sendo elementos de uma matriz inversa dos canais de mistura conhecida $\mathbf{W}(f) \in \mathbb{C}^{2\times 2}$. Note que os dois elementos estão na mesma coluna de $\mathbf{W}(f)$, mas em linhas diferentes. Supondo estimação perfeita após os processos de AJD e que a ambiguidade de escala foi solucionada, a curva tracejada mostra uma simulação da magnitude da resposta de frequência de $\hat{w}_{11,\mathrm{ajd}}(f)$ estimado. Note que, após estimação preliminar de $\hat{\mathbf{W}}_{\mathrm{ajd}}(f) = \hat{\mathbf{H}}^{-1}_{\mathrm{ajd}}(f)$, os valores de $20\log|\hat{w}_{11,\mathrm{ajd}}(f)|$ se alternam de forma frequente entre $w_{11}(f)$ e $w_{21}(f)$, indicando as permutações entre as duas linhas de $\hat{\mathbf{W}}_{\mathrm{ajd}}(f)$.

Após a correção realizada pelo primeiro estágio, a matriz estimada $\hat{\mathbf{W}}_{\mathrm{stg1}}(f) \in \mathbb{C}^{I\times J}$ alimenta o segundo estágio da solução. Diferentemente da saída dos processos de AJD, as permutações em $\hat{\mathbf{W}}_{\mathrm{stg1}}(f)$ tendem a ocorrer em componentes isolados de frequência (PHAM; SERVIÈRE; BOUMARAF, 2003b), como pode ser visto na Fig. 3.1. Note que um erro na correção de uma permutação é propagado por um número mais largo de frequências. A curva pontilhada na Fig. 3.1 mostra esse comportamento. $20\log|\hat{w}_{11,\mathrm{stg1}}(f)|$ é elemento de $\hat{\mathbf{W}}_{\mathrm{stg1}}(f)$, um exemplo de matriz inversa estimada e preliminarmente corrigida pelo primeiro estágio da solução. Note que os valores de $20\log|\hat{w}_{11,\mathrm{stg1}}(f)|$ se alternam com menos frequência que $20\log|\hat{w}_{11,\mathrm{ajd}}(f)|$. Neste exemplo, é possível visualizar uma permutação no componente de frequência de número 269, onde $20\log|\hat{w}_{11,\mathrm{stg1}}(f)|$ deixa de assumir valores de $w_{11}(f)$ e passa a assumir valores de $w_{21}(f)$. O mesmo acontece para o elemento $20\log|\hat{w}_{12,\mathrm{stg1}}(f)|$, que passa a assumir valores de $w_{22}(f)$

(não mostrado aqui). Ou seja, no contexto desta figura, as linhas de $\hat{\mathbf{W}}_{\text{stg1}}(f)$ são permutadas no componente de frequência 269 e se mantém desta forma por um número maior de frequências. O segundo estágio da solução para ambiguidade da permutação explora este comportamento e supõe que o espectro variante no tempo dos sinais estimados variam de forma suave o suficiente com a frequência.

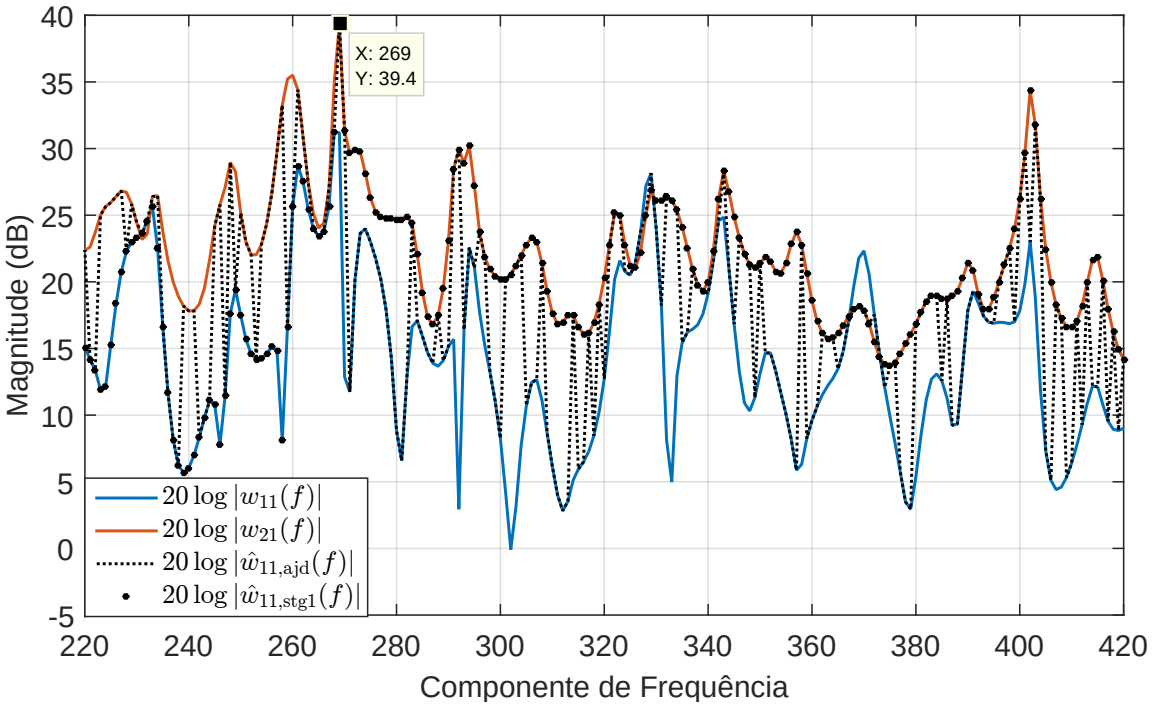

Figura 3.1: Exemplo do comportamento das permutações antes e após a correção pelo primeiro estágio da solução para o problema de ambiguidade de permutação.

3.2.1 Descrição do Algoritmo

A Figura 3.2 mostra uma visão geral da solução para o problema da permutação de (SERVIÈRE; PHAM, 2006) em um diagrama de blocos, realçando seu segundo estágio. O primeiro estágio é representado pela caixa preta no passo 1. Os processos relacionados ao segundo estágio são representados nos passos de 2 a 13, os quais são descritos a seguir.

De acordo com o segundo processo no diagrama da Fig. 3.2, o primeiro passo do segundo estágio é computar o espectro variante no tempo $\hat{S}_y(f, b, i)$, $i = 1, \ldots, I$ do i-ésimo sinal de fonte estimado $\hat{y}_i(n)$, utilizando a saída do primeiro estágio $\hat{\mathbf{W}}_{\text{stg1}}(f)$ e a matriz espectral $\hat{\mathbf{S}}_x(b, f) \in \mathbb{C}^{J\times J}$ estimada de acordo com a Eq. 2.12, da seguinte forma:

$$\hat{\mathbf{S}}_y(f, b) = \text{diag}\left[\hat{\mathbf{W}}_{\text{stg1}}(f)\hat{\mathbf{S}}_{\mathbf{x}}(f, b)\hat{\mathbf{W}}_{\text{stg1}}(f)^{\text{H}}\right] \in \mathbb{R}^{I\times 1}, \tag{3.11}$$

onde o operador $\text{diag}[\cdot]$ extrai os elementos da diagonal de uma matriz de dimensões $I \times I$ em um

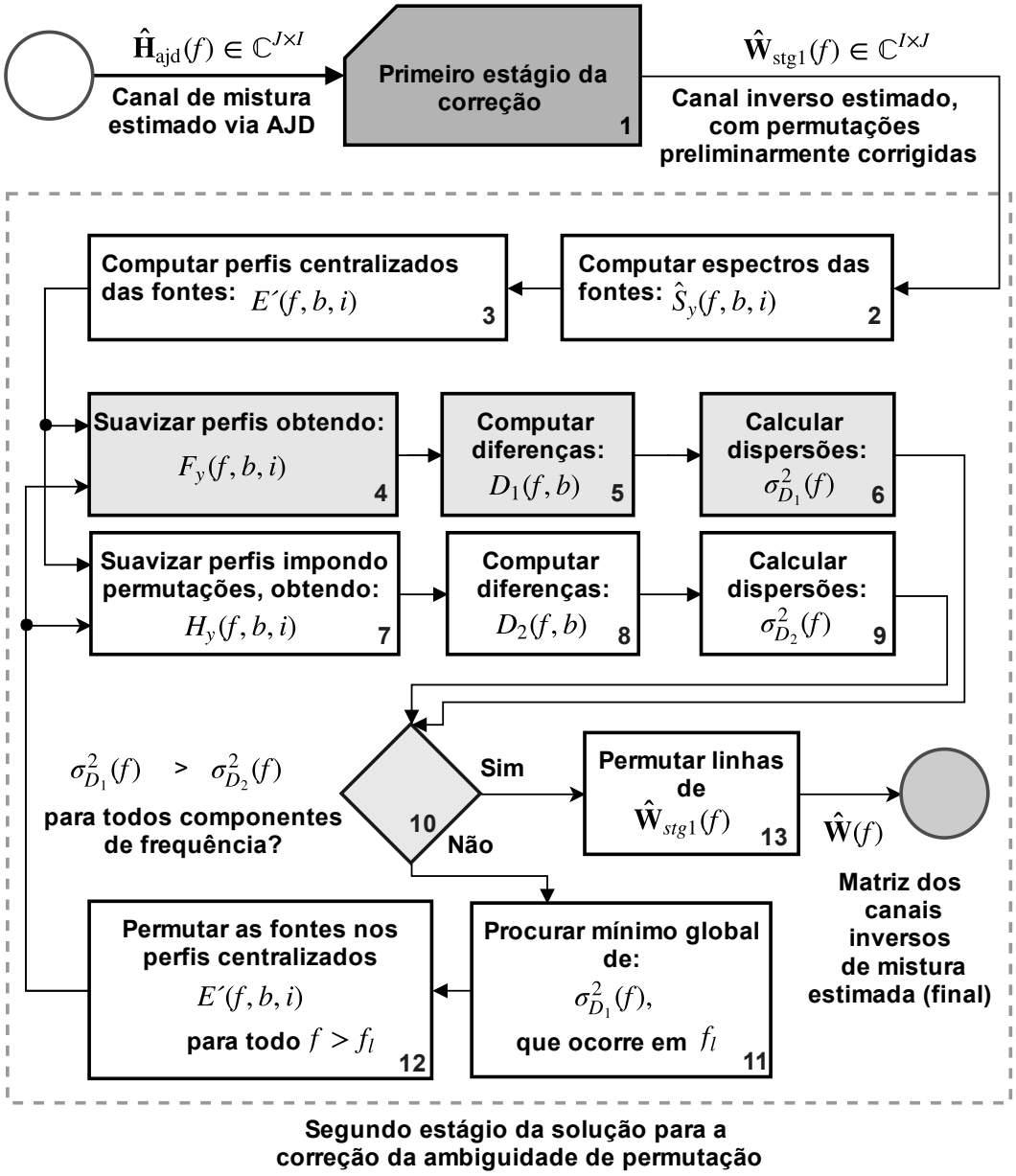

Figura 3.2: Visão geral da solução de dois estágios proposta por (SERVIÈRE; PHAM, 2006), com foco no segundo estágio do algoritmo.

vetor de tamanho $I \times 1$. Note que, para fins da correção da ambiguidade da permutação, podemos fazer uso de um lado do espectro, contendo apenas frequências positivas, e espelhar o conjugado dos dados para o outro lado do espectro após as correções terem sido realizadas. Desta forma, $f = 1, \ldots, F/(2{+}1)$. A Fig. 3.3(a) mostra um exemplo de espectro de duas fontes. Comparando com a Fig. 3.3(b), onde é possível ver os espectros reconstruídos $\hat{\mathbf{S}}_y(f,b)$ considerando uma única permutação remanescente, podemos notar que ela adiciona uma certa descontinuidade na frequência onde ocorre, indicada na figura. O objetivo deste estágio de correção é detectar essas descontinuidades para encontrar as frequências onde as permutações acontecem.

Devido à ambiguidade de escala, cada espectro é recuperado a menos de um fator de ganho

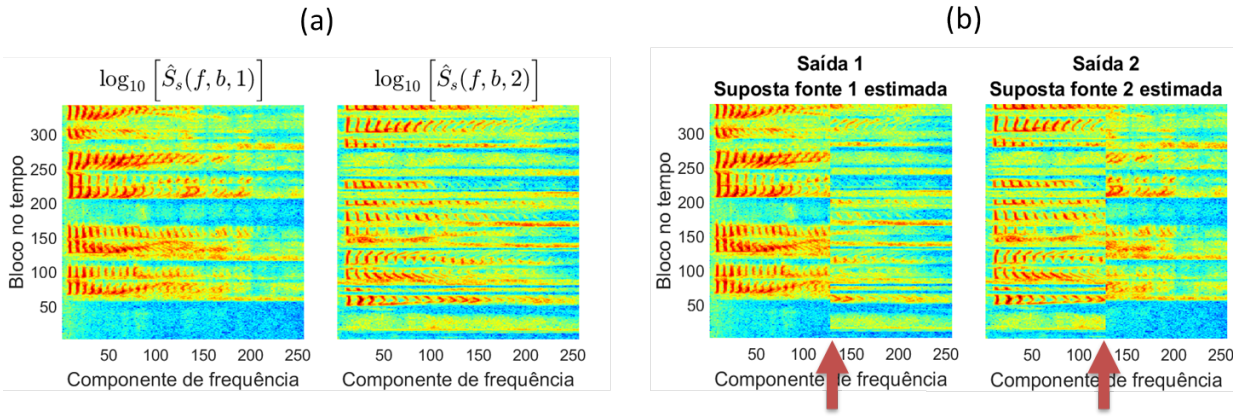

Figura 3.3: Estimativa da densidade espectral de potência dos sinais das fontes (a). Espectros dos sinais estimados considerando uma permutação remanescente do primeiro estágio (b).

(SERVIÈRE; PHAM, 2006). Tomando seu logaritmo natural, os chamados perfis das fontes $E(f,b,i)$ podem ser obtidos por obtidos por:

$$E(f,b,i) \triangleq \ln[\hat{S}_y(f,b,i)]. \tag{3.12}$$

Se então subtrairmos a média de $E(f,b,i)$ sobre os B blocos no tempo, removemos este fator de ganho e os assim chamados perfis centralizados das fontes $E'(f,k,i)$, mostrados no bloco 3 da Fig. 3.2, são obtidos:

$$E'(f,b,i) = E(f,b,i) - \frac{1}{B}\sum_{b_k=1}^{B} E(f,b_k,i). \tag{3.13}$$

Os perfis centralizados das fontes são representações em tempo-frequência dos sinais das fontes. Como, para detecção das permutações, queremos que esses perfis variem de forma suave com a frequência, computamos os perfis suavizados $F_y(f,b,i)$ através do cálculo da média dos perfis centralizados da fontes $E'(f,k,i)$ em uma determinada largura de banda $[f_l - M, f_l + M]$, para cada bloco b, de tal forma que:

$$F_y(f_l,b,i) = \frac{1}{2M+1}\left(\sum_{k=l-M}^{n=l+M} E'(f_k,b,i)\right). \tag{3.14}$$

Os primeiros e últimos M pontos de $E'(f,b,i)$ são negligenciados, já que não há pontos suficientes para o cálculo da média para eles. Estas representações em tempo-frequência das fontes são base para todos os cálculos subsequentes. Um exemplo de perfil suavizado $F_y(f,b,1)$ para uma sinal reconstruído e sem permutações é mostrado na Fig. 3.4. Note o efeito da suavização com relação às frequências para o bloco $b = 20$ em foco.

O método em (SERVIÈRE; PHAM, 2006), descrito aqui, pode lidar com mais de duas fontes

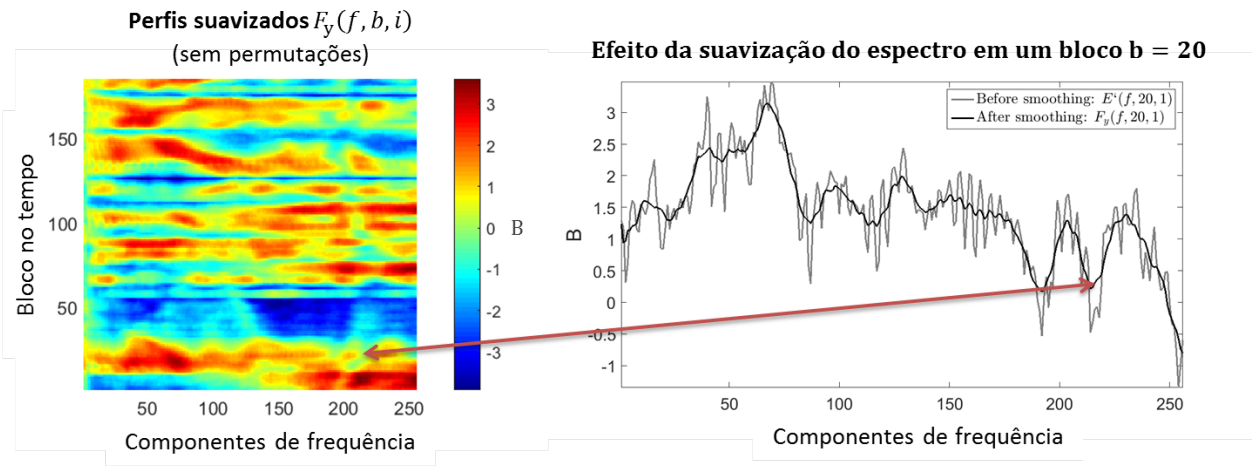

Figura 3.4: Exemplo de perfil suavizado $F_y(f, b, 1)$ para uma sinal reconstruído sem permutações. O efeito da suavização com relação às frequências pode ser visto para o bloco $b = 20$ em foco. Parâmetros da simulação: $N = 512$; $m = 3$; $\alpha = 0,75$; sinal da fonte MALEVOICE WoDC.wav; $N_x = 24000$ amostras; taxa de amostragem $= 11000$ Hz; e $M = 3$.

analisando-as de duas em duas. Portanto, sem perda de generalidade e supondo que duas fontes são selecionadas para a análise, consideramos que $I = J = 2$ de agora em diante.

Após cálculo de $F_y(f, b, i)$, para detectar as permutações, primeiro computamos as diferenças $D_1(f, b)$ entre os perfis suavizados $F_y(f, b, i)$, dadas por:

$$D_1(f, b) = F_y(f, b, 1) - F_y(f, b, 2). \tag{3.15}$$

Em um componente de frequência $f_l - M$ e bloco b no tempo, se os perfis suavizados variam devagar o suficiente com a frequência e, por exemplo, $F_y(f_l - M, b, 1) > F_y(f_l - M, b, 2)$, $D_1(f_l - M, b)$ é um escalar positivo. Se há uma permutação entre as fontes em f_l, $D_1(f_l + M, b)$ tem sinal oposto, sendo um escalar negativo. Essa mudança de sinal na frequência f_l é o que podemos detectar para identificar as frequências onde as permutações ocorrem. A Fig. 3.5 mostra um exemplo das diferenças $D_1(f, b)$ em função da frequência, para vários blocos no tempo. Note a indicação dos pontos onde ocorrem mudanças de sinais.

Já que a Eq. (3.15) é realizada para B blocos no tempo, podemos explorar as informações contidas em cada um desses blocos através do cálculo das dispersões $\sigma^2_{D_1}(f)$ das diferenças $D_1(f, b)$ entre os B blocos, de tal forma que:

$$\sigma^2_{D_1}(f) = \frac{1}{B} \sum_{b=1}^{B} D_1^2(f,b). \tag{3.16}$$

Como mostrado na Fig. 3.6, uma mudança de sinal em $D_1(f_l + M, b)$ é refletida em um ponto de mínimo de $\sigma^2_{D_1}(f)$, o que pode nos fornecer as frequências f_l onde as permutações ocorrem. Este pontos são indicados por setas vermelhas na Fig. 3.6. Entretanto, $\sigma^2_{D_1}(f)$ tem vários pontos de mínima dispersão. Se, por exemplo, os dois pontos de mínimo $\sigma^2_{D_1}(f)$ indicados na Fig. 3.6

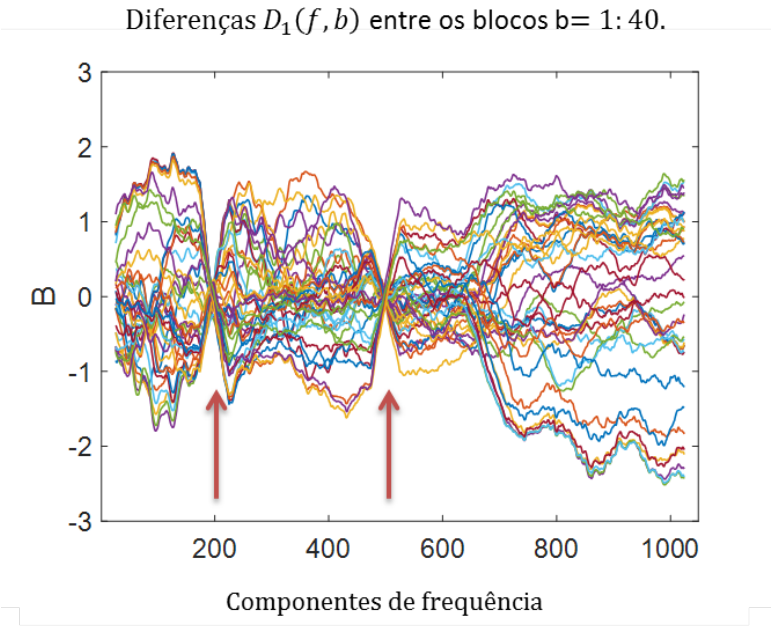

Figura 3.5: Exemplo de diferenças $D_1(f,b)$ em funçaõ da frequência, para blocos no tempo $1 \leq b \leq 41$. Parâmetros da simulação: $N = 2048$, $m = 5$; $\alpha = 0,75$; sinais: speech1.wav; speech2.wav; $N_x = 24000$ amostras; taxa de amostragem $= 11.025$ Hz; $\hat{\mathbf{W}}(f)$ conhecido e com duas permutações simuladas; $M = 12$.

fossem removidos (se as permutações correspondentes fossem corrigidas e as dispersões recalculadas), note que o novo mínimo global de $\sigma^2_{D_1}(f)$ seria o indicado pelo círculo verde. Este não corresponde a uma permutação.

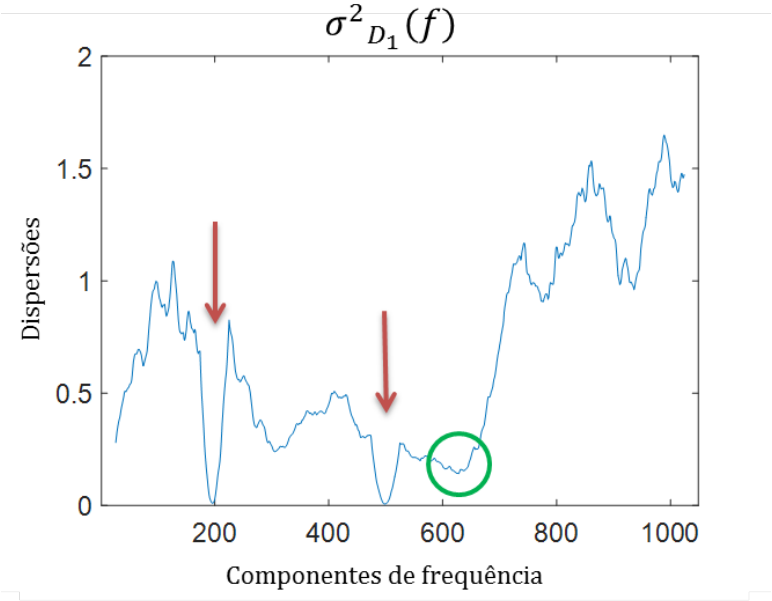

Figura 3.6: Exemplo de dispersões $\sigma^2_{D_1}(f)$. Mesmos parâmetros de simulação que a Fig. 3.5. As setas vermelhas indicam pontos de mínima dispersões, onde permutações ocorrem. O círculo verde indica o próximo ponto de mínimo global de $\sigma^2_{D_1}(f)$ que seria encontrado caso os outros dois pontos fossem removidos. Este não corresponde a uma permutação.

Assim, (SERVIÈRE; PHAM, 2006) computa também um segundo conjunto de dispersões

$\sigma^2_{D_2}(f)$. de acordo com a Fig. 3.2, blocos de 7 a 9, para calcular $\sigma^2_{D_2}(f)$, precisamos primeiro computar os perfis suavizados impondo permutações $H_y(f, b, i)$:

$$H_y(f_l, b, i) = \frac{1}{2M+1}\left(\sum_{k=l-M}^{n=l} E'(f_k, b, i) + \sum_{k=l+1}^{n=l+M} E'(f_k, b, \pi)\right), \tag{3.17}$$

com π representando a permutação entre as duas fontes. Os perfis $H_y(f, b, i)$ são obtidos através do cálculo da média dos perfis centralizados das fontes E'(f, b, i) em uma largura de banda $[f_l - M, fl + M]$, para cada bloco b, porém impondo a permutação das fontes no lado direito da banda $[f_l+1, fl+M]$. Cada componente de frequência f_l de $H_y(f, b, 1)$ é computado como a média dos $f_l - M, \ \ldots \ , f_l$ pontos de E'$(f, b, 1)$ e dos $f_l + 1, \ \ldots \ , fl + M$ pontos de E'$(f, b, 2)$. De forma análoga ao cálculo de $F_y(f, b, i)$, os primeiros e últimos M pontos de E'(f, b, i) são descartados. As diferenças $D_2(f, b)$ entre os perfis suavizados com permutações impostas $H_y(f, b, i)$ são então obtidas de:

$$D_2(f, b) = H_y(f, b, 1) - H_y(f, b, 2). \tag{3.18}$$

Finalmente, as dispersões $\sigma^2_{D_2}(f)$ das diferenças $D_2(f, b)$, entre os B blocos no tempo, é dada para cada componente de frequência por:

$$\sigma^2_{D_2}(f) = \frac{1}{B}\sum_{b=1}^{B} D_2^2(f,b). \tag{3.19}$$

Um exemplo de dispersões $\sigma^2_{D_2}(f)$ é mostrado na Fig. 3.7. Uma vez que para calcular $H_y(f, b, i)$ permutações foram impostas, quando uma permutação ocorre em uma frequência f_l, as fontes em $H_y(f, b, i)$ estão alinhadas entre $f = f_l - M, \cdots , f_l + M$ e, portanto, não há mudanças de sinal em $D_2(f_l, b)$. Assim, em f_l, não há pontos de mínima dispersão $\sigma^2_{D_2}(f)$, mas pontos de máxima dispersão. Consequentemente, $\sigma^2_{D_2}(f_l) > \sigma^2_{D_1}(f_l)$ e f_l é um ponto de máximo com um padrão distinto.

Após cálculo das dispersões, os passos de 10 a 12 da Fig. 3.2 são realizados. Primeiramente, é verificado se de fato existem permutações a serem corrigidas, i.e. se $\sigma^2_{D_2}(f) > \sigma^2_{D_1}(f)$ para qualquer componente de frequência. Então, se os sinais de fala são aproximadamente não-estacionários e se eles possuem variações de energia distintas o suficiente entre os B blocos para os F componentes de frequência, cada pico de $\sigma^2_{D_1}(f)$ que faz com que $\sigma^2_{D_2}(f) > \sigma^2_{D_1}(f)$ seja reconhecido como um pico ocasionado por uma permutação. Assim, após verificar se existem permutações a serem corrigidas, procuramos a frequência de permutação f_l onde o mínimo global de $\sigma^2_{D_1}(f)$ ocorre. Tendo encontrado f_l, atualizamos os perfis centralizados E'(f, k, i) permutando suas fontes para todas as frequências acima de f_l, inclusive. Considerando a correção efetuada, E'(f, k, i) alimenta os processos na próxima iteração, que começa no passo 4 do diagrama da Fig. 3.2. Devido à permutação de E'(f, k, i) com base na frequência encontrada, as novas dispersões

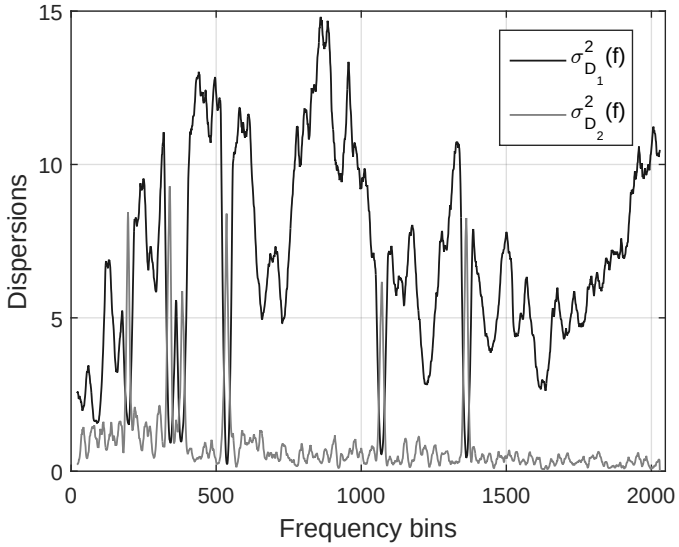

Figura 3.7: Exemplo de dispersões $\sigma^2_{D_1}(f)$ e $\sigma^2_{D_2}(f)$. Parâmetros da simulação: $F = 4096$, $\alpha = 0,75$, $m = 1$, duração dos sinais das fontes: 2 s, $L = 256$, $M = 20$, nome dos arquivos das fontes: 'poem male 30s.wav' e 'sentences female 28s.wav'.

a serem computadas não apresentarão o pico anteriormente encontrado. As iterações continuam até que $\sigma^2_{D_2}(f) < \sigma^2_{D_1}(f)$ para todo f.

Após finalização do processo iterativo, a resposta de frequência da matriz inversa dos canais estimada $\hat{\mathbf{W}}_{\text{stg1}}(f) \in \mathbb{C}^{I \times J}$ pode ser efetivamente corrigida: para cada frequência f_l encontrada, permuta-se suas linhas para todo $f = f_l, \ldots, F$. Esse processo corresponde ao último passo na Fig. 3.2. Dessa forma, a matriz inversa dos canais estimada $\hat{\mathbf{W}}(f)$ é finalmente obtida e pode ser utilizada para recuperar os sinais das fontes a partir da aplicação da Eq. 3.9, seguida pela transformada inversa de Fourier e final reconstrução dos sinais por superposição no domínio do tempo, como mencionado no último parágrafo da Seção 2.2.

3.3 MÉTODO PROPOSTO

Nesta Seção, é proposto um método parcialmente iterativo como alternativa ao segundo estágio da solução para o problema de permutação proposta por (SERVIÈRE; PHAM, 2006). Diferentemente da técnica original, este método não requer o cálculo das dispersões $\sigma^2_{D_1}(f)$. Para isso, é utilizado um valor limiar para para detecção dos pontos de máximo de $\sigma^2_{D_2}(f)$ relacionados a permutações. Além disso, para detecção de permutações existentes a serem corrigidas, a variação da média de $\sigma^2_{D_2}(f)$ entre iterações é verificada. Com essas técnicas são esperados ganhos computacionais mantendo resultados similares ao método original. Esta Seção é dividida em quatro Subseções: a Subseção 3.3.1 descreve o fluxo dos processos do segundo estágio proposto; 3.3.2 mostra como a existência de permutações remanescentes do primeiro estágio da solução é detectada; a computação de um valor limiar para as dispersões $\sigma^2_{D_2}(f)$, de forma que várias frequências

onde permutações ocorrem possam ser detectadas em uma única iteração, é apresentada na Subseção 3.3.3; e, por fim, 3.3.4 resume o algoritmo para o segundo estágio proposto.

O método proposto neste trabalho se baseia nas seguintes observações. A técnica em (SERVIÈRE; PHAM, 2006) é iterativa e requer uma iteração por permutação a ser corrigida. Uma pergunta natural que surge é se o problema pode ser resolvido em menos iterações sem sacrificar os resultados. Esse aspecto é importante se os canais de mistura tiverem longos comprimentos L, o que acarreta em um número maior de permutações remanescente do primeiro estágio. Além disso, em cada iteração, ambos conjuntos de dispersões $\sigma^2_{D_1}(f)$ e $\sigma^2_{D_2}(f)$ são calculados com o objetivo principal de detectar a existência de permutações a serem corrigidas. Se a mesma tarefa pode ser realizada utilizando apenas $\sigma^2_{D_2}(f)$, podemos esperar ganhos em termos de esforço computacional. Finalmente, se em um trecho do áudio das fontes, selecionado para a análise, a energia em um componente de frequência não varia suficientemente com o tempo, as dispersões $\sigma^2_{D_1}(f)$ e $\sigma^2_{D_2}(f)$ podem aproximar-se de forma prematura e, neste caso, o processo iterativo só chega ao fim se um número máximo e arbitrário de iterações for atingido. Como os resultados do método original são aceitáveis se o problema ocorrer no extremo inferior do espectro de frequência, uma alternativa que tenha o potencial de evitar o número extra de iterações, diminuindo o tempo de cálculo, e que tenha uma taxa de acerto na correção das permutações comparável ao método original é bem vinda.

3.3.1 Fluxo dos processos do método proposto

A Fig. 3.8 mostra uma visão geral dos processos que envolvem a solução para o problema de permutação, realçando o segundo estágio proposto neste trabalho, que começa no bloco 2. Note que, após estimação preliminar da matriz dos canais de mistura $\hat{\mathbf{H}}_{\mathrm{ajd}}(f) \in \mathbb{C}^{J\times I}$ via AJD, como descrito na Seção 2.2.2, o primeiro estágio da correção é realizado de acordo com o proposto em (PHAM; SERVIÈRE; BOUMARAF, 2003b) e detalhado na Seção 3.1. Em seguida, a saída desse subsistema $\hat{\mathbf{W}}_{\mathrm{stg1}}(f) \in \mathbb{C}^{I\times J}$ alimenta o segundo estágio da solução.

Como o método utiliza representações em tempo-frequência dos sinais de fonte estimados para a análise, o primeiro passo no segundo estágio proposto é a computação do espectro variante no tempo $\hat{S}_y(f, b, i)$, $i = 1, \cdots, I$ do i-ésimo sinal de fonte estimado. Isso pode ser feito a partir da Eq. (3.11), utilizando a saída do primeiro estágio $\hat{\mathbf{W}}_{\mathrm{stg1}}(f)$ e a matriz espectral dos sinais gravados $\hat{\mathbf{S}}_x(b, f)$, já obtida como parte do processo de estimação de $\hat{\mathbf{H}}_{\mathrm{ajd}}(f)$, a partir da Eq. (2.12).

Em seguida, obtemos os perfis centralizados da fontes $E'(f, b, i)$ a partir da equação (3.12). De forma análoga ao método original, no qual este é baseado, removemos assim o fator de ganho desconhecido devido à ambiguidade de escala.

Diferentemente de (SERVIÈRE; PHAM, 2006), ao invés de procurarmos por pontos de mínimo de $\sigma^2_{D_1}(f)$ para detectar as frequências onde as permutações ocorrem, procuramos aqui por pontos de máximo de $\sigma^2_{D_2}(f)$, já que a curva resultante desses dados possui um formato caracterís-

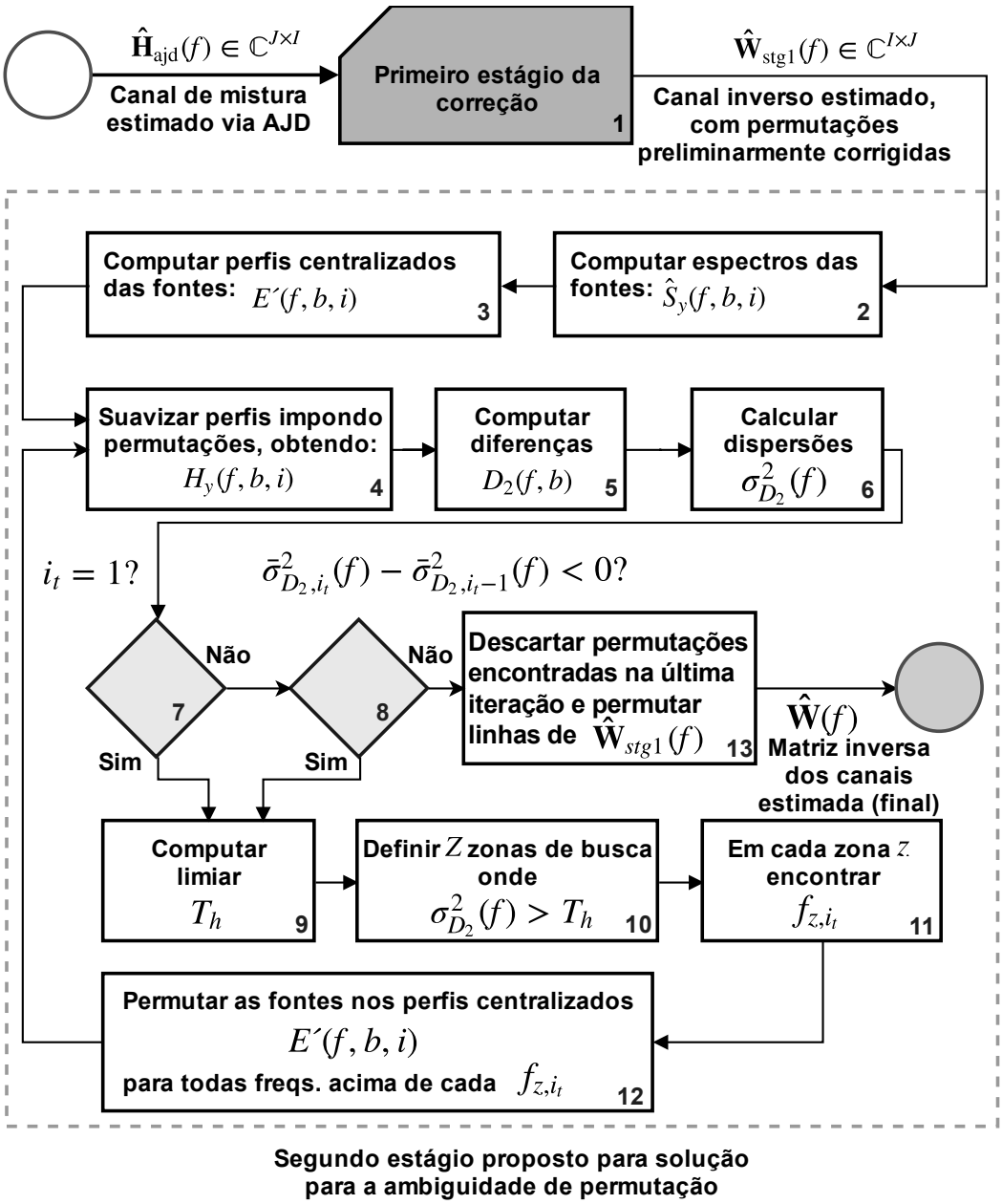

Figura 3.8: Visão geral dos processos que envolvem a solução para o problema de permutação, realçando o segundo estágio proposto.

tico mais apropriado para o nosso método baseado em um valor limiar. Isto é, pelas Figuras 3.6 e 3.7, podemos notar que os "valores de teto" de $\sigma^2_{D_1}(f)$ possuem grande variabilidade, o que torna mais difícil a distinção de vales causados por permutações de outros vales aleatórios. De forma contrária, pelas Figuras 3.7 e 3.9 podemos ver que $\sigma^2_{D_2}(f)$ possui menor variabilidade em seus "valores de base", o que diminui o mascaramento dos picos relacionados a permutações. Esses "valores de teto e de base" são variações naturais nas dispersões que ocorrem quando não há permutações remanescentes a serem corrigidas. Se as premissas para o método forem atendidas, os pontos de máximo de $\sigma^2_{D_2}(f)$, destacáveis do demais pontos de máximo local (ou "valores base"), indicam as frequências onde as permutações ocorrem. Neste caso, é possível traçar uma linha

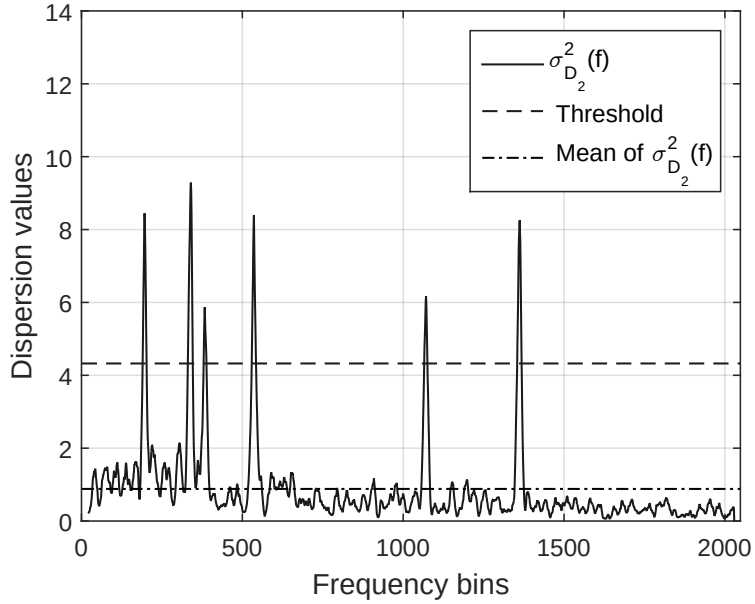

Figura 3.9: Exemplo de dispersões $\sigma^2_{D_2}(f)$ sendo divididas em zonas de busca por um valor limiar. A média de $\sigma^2_{D_2}(f)$ também é mostrada. Parâmetros da simulação: $F = 4096$, $\alpha = 0.75$, $m = 1$, duração dos sinais $= 2$ s, $L = 256$, $M = 20$, nome dos arquivos: 'poem male 30s', 'sentence female 28s'.

limiar de tal forma que possamos distinguir os picos relacionados a permutações de outros picos de ruído, como mostrado na Fig. 3.9. Portanto, procedemos computando as dispersões $\sigma^2_{D_2}(f)$ pela Eq. (3.19), primeiro obtendo os perfis suavizados com permutações impostas $H_y(f, b, i)$ a partir da Eq. (3.17) e então suas diferenças $D_2(f, b)$ pela Eq. (3.18). Aqui, um processo iterativo se inicia. A cada iteração, novas dispersões $\sigma^2_{D_2}(f)$ são calculadas.

Após cálculo das dispersões $\sigma^2_{D_2}(f)$, passo 7 da Fig. 3.8, na primeira iteração supomos que existem permutações a serem corrigidas e seguimos para os passos de 9 a 11 da Fig. 3.8 para encontrar os Z componentes de frequência onde elas ocorrem:

$$\mathbf{f}_{l,i_t} = [f_{1,i_t} \;\; f_{2,i_t} \;\; \ldots \;\; f_{z,i_t}]^{\mathrm{T}} \in \mathbb{N}^{*Z\times 1}, \tag{3.20}$$

onde $z = 1, \ldots, Z$, $i_t = 1, \ldots, I_t$ é um índice que indica o número da iteração atual e $\mathbb{N}^* = \mathbb{N} - \{0\}$ (conjunto dos naturais sem o zero). Note que Z pode variar entre iterações.

De acordo com o passo 9 da Fig. 3.8, para detectar as frequências $\mathbf{f}_{l,it}$ onde permutações em potencial ocorrem, primeiramente o valor de limiar T_h para $\sigma^2_{D_2}(f)$ precisa ser calculado. Isso é realizado como descrito na Subseção 3.3.3.

Após computação de T_h, prosseguimos para os passos 10 e 11 da Fig. 3.8, procurando pelos pontos de máximo de $\sigma^2_{D_2}(f)$ que se encontram dentro de zonas de frequência onde $\sigma^2_{D_2}(f) > T_h$. Uma zona de frequência z é delimitada por $f = f_{z,start}, \ldots, f_{z,end}$, onde $f_{z,start}$ é o z-ésimo primeiro ponto onde $\sigma^2_{D_2}(f) > T_h$ e $(f_{z,end} + 1)$ é o primeiro ponto, de frequência maior que $f_{z,start}$, onde $\sigma^2_{D_2}(f) < T_h$. Para cada zona de frequência z, o componente de frequência onde uma permutação em potencial ocorre é calculado por:

$$f_{z,i_t} = \arg\max_{f} \left[\sigma^2_{D_2}(f)\right] + M + 1,\ f_{z,start} \leq f \leq f_{z,end}, \tag{3.21}$$

onde o fator aditivo $(M+1)$ decorre de que, quando os perfis suavizados das fontes $H_y(f,b,i)$ são computados de acordo com a Eq. (3.17), os primeiros M componentes de frequência de $E'(f,b,i)$ são descartados, já que não há pontos suficientes para cálculo do valor médio para estes. Assim, $H_y(f,b,i)$ contém M pontos a menos nas baixas frequências e este fato é compensado aqui. Além disso, um componente de frequência f_k de $H_y(f,b,i)$ é calculado pela Eq. (3.17) impondo uma permutação a partir de $f_k + 1$ em diante. Assim, o valor de máximo em $\sigma^2_{D_2}(f)$ na verdade não ocorre exatamente na frequência onde uma permutação ocorre, mas em um componente de frequência acima. Isto é compensando pelo adição do fator unitário na Eq. (3.21).

Tendo encontrado os candidatos $\mathbf{f}_{l,i_t} \in \mathbb{N}^{*Z\times 1}$, prosseguimos para o passo 12 da Fig. 3.8. Neste processo, atualizamos os perfis centralizados $E'(f,b,i)$ permutando suas fontes para todas as frequências acima de cada $f_{l,i_t}(z)$ encontrado, inclusive. Na próxima iteração, que começa no passo 4 da Fig. 3.8, $\sigma^2_{D_2}(f)$ é computado novamente e checamos se de fato existiam permutações a serem corrigidas na iteração anterior. Isso é feito no passo 8 da Fig. 3.8, através da verificação da variação da média de $\sigma^2_{D_2}(f)$, como descrito na Subseção 3.3.2. Se a variação é negativa, é considerado que permutações realmente existiam e foram corrigidas em $E'(f,b,i)$. Portanto, mantemos o vetor de frequências encontradas $\mathbf{f}_{l,i_t-1}$ e procuramos por mais candidatos, continuando o processo iterativo. Se a variação da média é positiva, é consideramos que não existiam permutações remanescentes a serem corrigidas. Portanto, o processo iterativo é finalizado e o vetor das frequências encontradas na última iteração $\mathbf{f}_{l,I_t-1}$ é descartado. Dessa forma, o vetor $\mathbf{f}_l$, contendo todas frequências encontradas onde permutações ocorrem, é dado por:

$$\mathbf{f}_l = \begin{cases} [\mathbf{f}_{l,1}^{\mathrm{T}}\ \ \mathbf{f}_{l,2}^{\mathrm{T}}\ \ \ldots\ \ \mathbf{f}_{l,I_t-2}^{\mathrm{T}}]^{\mathrm{T}},\ I_t > 2 \\ 0,\ I_t = 2 \end{cases}. \tag{3.22}$$

Note que, em cada iteração, mais de uma frequência onde ocorre uma permutação pode ser encontrada. Além disso, nesse processo iterativo, pelo menos duas iterações são necessárias.

Após o fim das iterações e a obtenção do vetor de frequências $\mathbf{f}_l$ onde as permutações ocorrem, a matriz inversa de resposta de frequência dos canais $\hat{\mathbf{W}}_{\mathrm{stg1}}(f)$ é efetivamente corrigida no passo 13 da Fig. 3.8. Para cada frequência f_l encontrada, permutamos as linhas de $\hat{\mathbf{W}}_{\mathrm{stg1}}(f)$ para todo $f = f_l, \ldots, F$. Assim, a matriz inversa dos canais $\hat{\mathbf{W}}(f)$ é finalmente estimada utilizando o segundo estágio proposto, podendo ser utilizada para recuperar os sinais das fontes como descrito no último parágrafo da Seção 2.2 (página 10).

3.3.2 Determinação da existência de permutações

Em (SERVIÈRE; PHAM, 2006), a existência de permutações remanescentes do primeiro estágio, ou entre iterações consecutivas, é detectada verificando se $\sigma^2_{D_1}(f) < \sigma^2_{D_2}(f)$ para qualquer

componente de frequência (passo 10 na Fig. 3.2). O método proposto neste trabalho explora o comportamento dos picos em $\sigma^2_{D_2}(f)$ de uma perspectiva diferente, sem utilizar $\sigma^2_{D_1}(f)$.

Para cada permutação corrigida, i.e. após atualizar os perfis centralizados $E'(f, b, i)$ no passo 12 da Fig. 3.8, é observado que de uma iteração para a próxima o pico ou valor de máximo correspondente em $\sigma^2_{D_2}(f)$ é removido, se tornando então um valor de mínima dispersão. De forma análoga, se um falso pico ou valor de máximo local for corrigido, pela permutação das fontes de $E'(f, b, i)$ e alimentação dos processos na próxima iteração, um pico é criado como resultado da falsa correção. Como os valores de dispersão ao redor do pico criado aumentam, podemos inferir que a média $\bar{\sigma}^2_{D_2}(f)$ também aumenta:

$$\bar{\sigma}^2_{D_2}(f) = \frac{1}{F - 2M} \sum_{f=1}^{F-2M} \sigma^2_{D_2}(f), \tag{3.23}$$

onde o termo $(F - 2M)$ expressa a quantidade total de componentes de frequência em $\sigma^2_{D_2}(f)$, já que $2M$ elementos são descartados para cálculo de $H_y(f, b, i)$ a partir da Eq. (3.17). Note que não importa quanto a média $\bar{\sigma}^2_{D_2}(f)$ aumenta, apenas que a variação seja positiva. Portanto, a variação de $\bar{\sigma}^2_{D_2}(f)$ é analisada entre iterações consecutivas da seguinte forma. Se:

$$\bar{\sigma}^2_{D_2,i_t}(f) - \bar{\sigma}^2_{D_2,i_t-1}(f) > 0, \tag{3.24}$$

as potenciais frequências de permutação encontradas em uma iteração anterior, se existirem, são descartadas e é considerado que não existiam permutações a serem corrigidas. Caso contrário, se $\bar{\sigma}^2_{D_2,i_t}(f) - \bar{\sigma}^2_{D_2,i_t-1}(f) < 0$, as frequências encontradas são mantidas e o processo iterativo continua. Assim, a existência de permutações é sempre verificada após permutações em potencial serem preliminarmente corrigidas, atualizando-se os perfis centralizados $E'(f, b, i)$ no passo 12 da Fig. 3.8 e os fornecendo aos processos na próxima iteração.

3.3.3 Determinação do valor limiar

Um valor ideal de limiar T_h para $\sigma^2_{D_2}(f)$ deve ser capaz de interceptar o máximo número de picos relacionados a permutações, enquanto negligencia outros picos não relacionados. O cálculo de T_h a seguir é baseado nos mesmos pressupostos para o segundo estágio da correção: variação suficiente da energia dos sinais das fontes para os diferentes componentes de frequência (não estacionaridade); além de variação lenta o suficiente dos perfis suavizados das fontes $F_y(f, b, i)$ com relação a frequência. Para computar um valor ótimo para T_h, primeiro ordenamos os dados de dispersão $\sigma^2_{D_2}(f)$ em ordem ascendente, de tal forma que:

$$\sigma^2_{D_2,\text{sort}}(n_f) = \big[\min(\sigma^2_{D_2}(f)),\ \ldots\ , \max(\sigma^2_{D_2}(f))\big]. \tag{3.25}$$

De acordo com a Fig. 3.10, $\sigma^2_{D_2,\text{sort}}(n_f)$ pode ser interpretado como o número de pontos

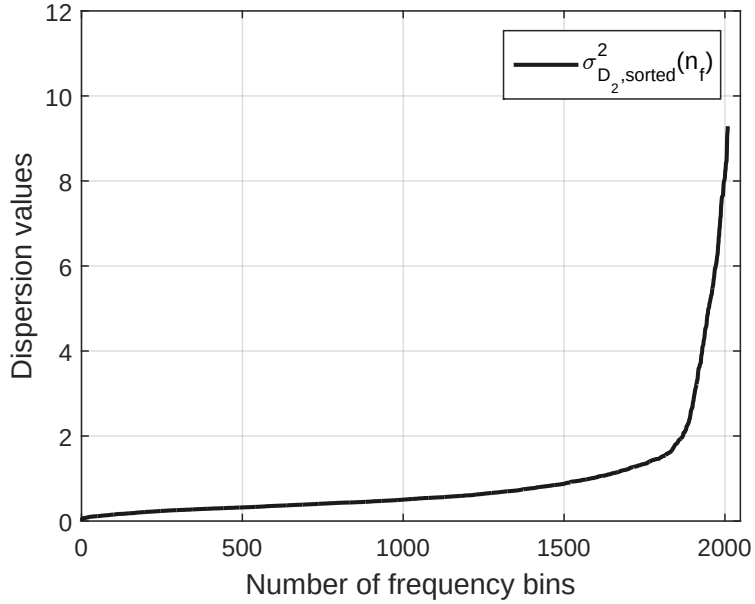

Figura 3.10: Exemplo de $\sigma^2_{D_2,\text{sort}}(f)$, com mesmos parâmetros de simulação que a Fig. 3.9

abaixo de um determinado valor de dispersão, em função das dispersões. A partir da Fig. 3.10 podemos observar que para pequenos valores de dispersão, a curva apresenta um comportamento exponencial com baixa taxa de crescimento, sendo aproximadamente linear. Mas em um determinado ponto, uma variação mais pronunciada na inclinação da curva é observada. Este comportamento ocorre especialmente quando o número de picos relacionados a permutações não é considerável se comparado com o número total de componentes de frequência F. Caso contrário, o número de pontos de mínima dispersão diminui devido à alta densidade de picos, o que diminui a distância entre os pontos de máximo $\sigma^2_{D_2}(f)$ e seus "valores de base" e, consequentemente, reduz a taxa de taxa de variação de $\sigma^2_{D_2,\text{sort}}(n_f)$. Quanto menor o número de permutações a serem corrigidas, mais enfatizada tende a ser a mudança na inclinação da curva. Essa mudança indica variação na contagem de pontos de dados à medida que os valores de dispersão aumentam, o que é uma pista de onde os picos de permutação começam e se distinguir de picos aleatórios de menor valor e não relacionados a permutações. Essa variação na inclinação é o que queremos detectar para encontrar um valor de limiar T_h adequado.

Em princípio, a derivada de $\sigma^2_{D_2,\text{sort}}(n_f)$ poderia nos fornecer informações sobre o ponto de maior variação. Entretanto, testes mostraram que o método é sensível a pequenas variações locais nos dados. Um método que se mostra menos impactado por ruído é realizado da seguinte maneira. Primeiramente, computamos uma soma cumulativa normalizada $C(n_f)$ de $\sigma^2_{D_2,\text{sort}}(n_f)$, como:

$$C(n_f) = \gamma \sum_{n_k=1}^{n_f} \sigma^2_{D_2,\text{sort}}(n_k), \tag{3.26}$$

onde γ é o fator de normalização dado por:

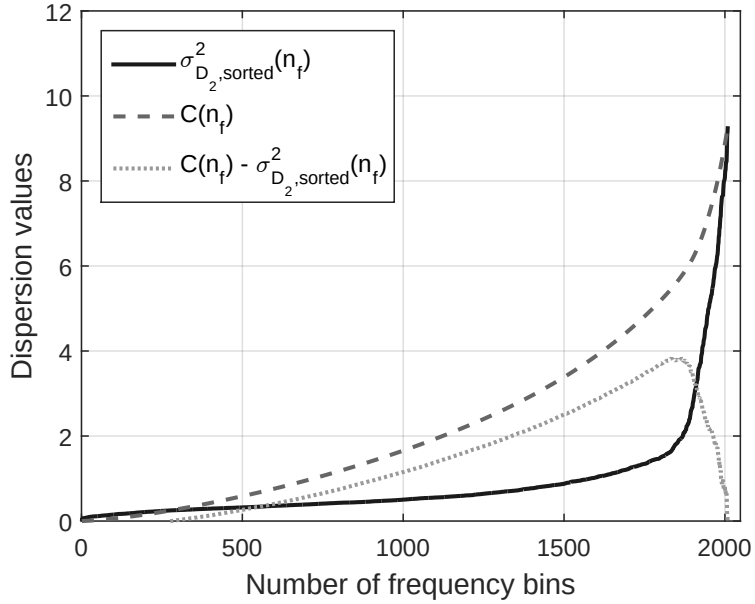

Figura 3.11: Exemplo de $\sigma^2_{D_2,\text{sort}}(f)$, $C(n_f)$ e a diferença entre elas, utilizando os mesmo parâmeros para a simulação que a Fig. 3.7

$$\gamma = \frac{\max\left(\sigma^2_{D_2,\text{sort}}\right)}{\sum_{n_p=1}^{F-2M} \sigma^2_{D_2,\text{sort}}(n_p)}. \tag{3.27}$$

A soma cumulativa na Eq. 3.26 é uma aproximação da função distribuição acumulada - do inglês, *Cumulative Distribution Function* (CDF) da variável aleatória $\sigma^2_{D_2,\text{sort}}(n_f)$. Na Fig. 3.11 exemplos de $\sigma^2_{D_2,\text{sort}}(n_f)$ e $C(n_f)$ podem ser visualizados. Note que, como resultado do fator de normalização $\gamma < 1$ aplicado na Eq. (3.26), as curvas geradas por $\sigma^2_{D_2,\text{sort}}(n_f)$ e $C(n_f)$ têm a mesma escala e sempre convergem no último ponto de dado, sendo este comum às duas curvas. Na Fig. 3.11 também pode ser observado que a diferença entre $C(n_f)$ e $\sigma^2_{D_2,\text{sort}}(n_f)$ tem valor máximo próximo do ponto de maior variação de $\sigma^2_{D_2,\text{sort}}(n_f)$. Essa relação entre $\sigma^2_{D_2,\text{sort}}(n_f)$ e $C(n_f)$ nos fornece informações úteis para o nosso objetivo, de tal forma que um valor preliminar T_p para o valor limiar pode ser obtido por:

$$T_p = \max\left[C(n_f) - \sigma^2_{D_2,\text{sort}}(n_f)\right]. \tag{3.28}$$

A existência para esse valor de máximo é garantida pelas premissas desta técnica para o segundo estágio da correção das permutações. Essas premissas são resumidas no primeiro parágrafo desta seção. A não existência de $\max\left[C(n_f) - \sigma^2_{D_2,\text{sort}}(n_f)\right]$ implicaria $\sigma^2_{D_2,\text{sort}}(f) = C(n_f)$. Por exemplo, se $\sigma^2_{D_2,\text{sort}}(f)$ gerasse uma linha reta. Porém, isso implica os sinais das fontes serem iguais ou proporcionais, quebrando as premissas de não correlação e não estacionaridade. Essa técnica para detecção da deflexão em $\sigma^2_{D_2,\text{sort}}(f)$ funciona de forma equivalente ao mencionado em (CHRISTOPOULOS, 2012), Lemma 1.4. Sendo que aqui é considerada a distância de $\sigma^2_{D_2,\text{sort}}(n_f)$ à soma cumulativa $C(n_f)$, ao invés da distancia de $\sigma^2_{D_2,\text{sort}}(n_f)$ à uma reta que liga seus pontos ex-

tremos. Todavia, dessa forma obtemos diretamente o valor a ser utilizado para o limiar preliminar T_p.

Finalmente, como mencionado anteriormente (na página 28), à medida que o número de permutações aumenta consideravelmente quando comparadas a F, a variação em $\sigma^2_{D_2,\mathrm{sort}}(n_f)$ tende a ser menos pronunciada, causando a aproximação das curvas plotadas na Fig. 3.11 para $\sigma^2_{D_2,\mathrm{sort}}(n_f)$ e $C(n_f)$. Neste cenário, o valor preliminar para limiar T_p encontra-se próximo dos pontos de mínima dispersão de $\sigma^2_{D_2}(f)$, podendo interceptar um pico ou valor de máximo local que não está relacionado com uma permutação. Idealmente, o limiar deve compreender todos os picos relacionados a permutações existentes, estando de forma segura longe dos outros picos de ruído, como mostrado na Fig. 3.9. Assim, uma informação adicional é necessária para contemplar cenários como este.

Um parâmetro estatístico de $\sigma^2_{D_2}(f)$ tem um comportamento oposto a T_p que pode explorado de forma adicional. À medida que o número de pontos de máxima dispersão (ou picos) aumenta, a mediana das dispersões $\sigma^2_{D_2}(f)$ também aumenta, variando de um valor negligenciável - não interferindo em T_p - até um valor de mesma ordem de grandeza quando o número de picos é considerável, nos fornecendo uma fronteira mínima. Portanto, utilizando a mediana de $\sigma^2_{D_2}(f)$ como uma fator adicional, o valor de limiar T_h, para detecção dos picos relacionados à permutações, pode ser calculado como:

$$T_h = \max\big(C(n_f) - \sigma^2_{D_2,\mathrm{sort}}(n_f)\big) + \tilde{\sigma}^2_{D_2}(f), \tag{3.29}$$

onde o operador $\tilde{(\cdot)}$ denota a mediana. Este valor é utilizado para definir as zonas de frequências mencionadas na Seção 3.3.1 (página 26). Como o número de permutações remanescentes pode mudar entre iterações consecutivas, de acordo com o passo 9 da Fig. 3.8, um novo valor de T_h é calculado em cada iteração.

3.3.4 Algoritmo para o método proposto

Com os procedimentos descritos nas Seções de 3.3.1 a 3.3.3, o pseudo-algoritmo para o método proposto para o segundo estágio da solução para o problema de ambiguidade da permutação pode ser resumido da seguinte forma:

1. Construa os perfis centralizados das fontes $E'(f, b, i)$ a partir da equação Eq. (3.13). Para isso, utilize o canal inverso estimado e com permutações preliminarmente corrigidas $\hat{\mathbf{W}}_{\mathrm{stg1}}(f)$, que é saída do primeiro estágio da correção (descrito na Seção 3.1);
2. Compute as dispersões $\sigma^2_{D_2}(f)$, a partir de $H_y(f, b, i)$ e $D_2(f, b)$, pelas Equações (3.19), (3.17) e (3.18), respectivamente.
3. Verifique a existência de permutações:

(a) Se esta é a primeira iteração, siga em frente para o passo 4 e encontre as frequências onde permutações potencialmente ocorrem. Caso contrário:

(b) Verifique se: $\bar{\sigma}^2_{D_2,i_t}(f) - \bar{\sigma}^2_{D_2,i_t-1}(f) < 0$, onde $\bar{\sigma}^2_{D_2,i_t}(f)$ é dado pela Eq. (3.23). Neste caso, armazene o vetor de frequências encontradas na iteração anterior $\mathbf{f}_{l,i_t-1}$ e siga para o passo 4 para procurar por mais candidatos . Caso contrário:

(c) Se $\bar{\sigma}^2_{D_2,i_t}(f) - \bar{\sigma}^2_{D_2,i_t-1}(f) > 0$, descarte o vetor de frequências encontrado na última iteração $\mathbf{f}_{l,i_t-1}$ e pare o processo iterativo. Finalmente, para cada frequência de permutação f_l encontrada (elementos de $\mathbf{f}_l$, Eq. 3.22), permute as linhas de $\hat{\mathbf{W}}_{\text{stg1}}(f)$ para todo $f = f_l, \ldots, F$, obtendo a matriz inversa dos canais estimada (final) $\hat{\mathbf{W}}(f)$.

4. Procure por potenciais frequências $\mathbf{f}_{l,i_t}$ onde permutações ocorrem :

   (a) Compute o limiar T_h, como definido na Eq. (3.29).

   (b) Defina Z zonas de busca delimitadas por $[f_{z,start}, f_{z,end}]$ onde $\sigma^2_{D_2}(f) > T_h$.

   (c) Para cada zona z, procure a potencial frequência onde uma permutação ocorre a partir da Eq. (3.21). Considerando Z zonas de frequência nesta iteração i_t, obtemos o vetor $\mathbf{f}_{l,i_t} \in \{\mathbb{N}^*\}^{Z\times 1}$.

5. Permute as fontes de $E'(f,b,i)$ de acordo com o vetor de frequências $\mathbf{f}_{l,i_t}$. Isto é, para cada frequência f_l encontrada, permute as linhas de $E'(f,b) \in \mathbb{R}^{2\times 1}$ para todas frequências iguais ou superiores a f_l. Assim, considerando cada f_l, $E'_{novo}(f,b,1) = E'(f,b,2)$ e $E'_{novo}(f,b,2) = E'(f,b,1)$, para todo $f = f_l, \ldots, F$. Em seguida, inicie uma nova iteração a partir do passo 2, utilizando os perfis centralizados atualizados $E'_{novo}(f,b,i)$.

4 SIMULAÇÕES E RESULTADOS

Neste capítulo, resultados experimentais comparando diferentes soluções para o problema da ambiguidade permutação são mostrados. O método proposto é comparado com duas soluções no estado da arte: (SERVIÈRE; PHAM, 2006), na qual o método proposto é baseado; e (PHAM; SERVIÈRE; BOUMARAF, 2003c), com modificações propostas por (NION et al., 2010) para cálculo dos, assim chamados, centroides dos perfis da fontes com base em um algoritmo de agrupamento do tipo *k-means*.

Os experimentos mostrados neste capítulo são realizados de forma que a influência de variáveis externas seja reduzida e os resultados das soluções para o problema de ambiguidade da permutação sejam mais direta e objetivamente avaliados. Dessa forma, a matriz dos canais de mistura estimada via AJD, é fornecida como entrada às soluções de correção da permutação supondo estimação perfeita pelos processos de AJD. Logo, é utilizada uma matriz dos canais de mistura $\mathbf{H}(f) \in \mathbb{C}^{J \times I}$ contendo permutações aleatórias de suas colunas entre seus F componentes de frequência. Essas permutações são inseridas artificialmente escolhendo-se de forma arbitrária uma matriz de permutação $\mathbf{\Pi}(f)$ (incluindo a matriz identidade) para cada componente de frequência. Devido à natureza aleatória deste processo, o número de permutações se aproxima de $F/2$. Juntamente com a matriz dos canais, as matrizes espectrais estimadas $\hat{\mathbf{S}}_x(f, b, i)$, calculadas como descrito na seção 2.2, são fornecidas como entrada para as soluções de correção da permutação. Portanto, as medidas de desempenho mostradas, como o tempo de cálculo, consideram este ponto de partida comum.

4.1 DESCRIÇÃO DOS EXPERIMENTOS

Os resultados das simulações são divididos em dois grupos. Os parâmetros comuns entre esses dois grupos são mostrados na Tabela 4.1.

Tabela 4.1: Parâmetros comuns entre os experimentos.

Número de fontes (I) e de microfones (J)	2 e 2, respectivamente.
Sinal da fonte 1	'poem male 30s.wav'. Fonte: (NION et al., 2010)
Sinal da fonte 2	'sentence female 28s'. Fonte: (NION et al., 2010)
Taxa de amostragem	11,025 kHz
Canais de mistura	'h256'. Fonte: (SERVIÈRE; PHAM, 2006)
Comprimento dos canais (L)	256
Núm. de blocos (m) para cálculo de $\hat{\mathbf{S}}_\mathbf{x}(b, f)$	1
Sobreposição entre os blocos no tempo (α)	0,75

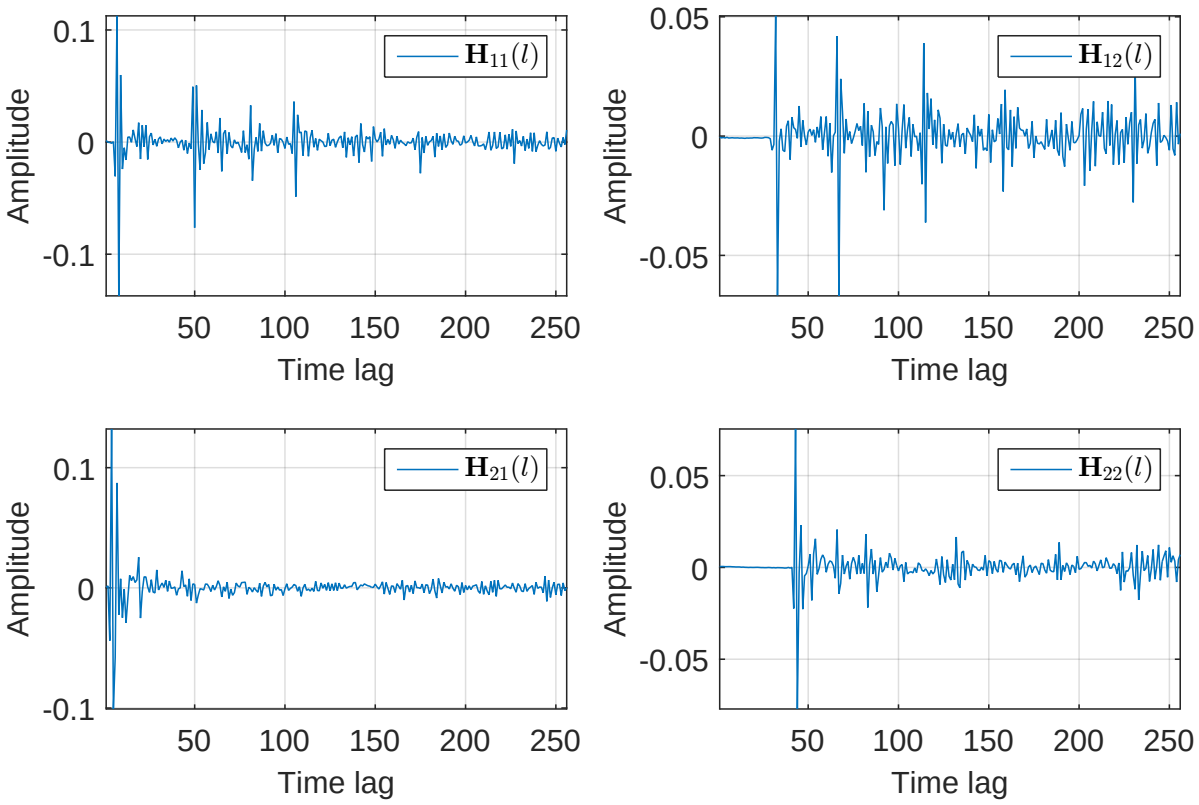

Figura 4.1: Respostas ao impulso utilizadas nas simulações. Comprimento $L = 256$.

Note que a matriz dos canais utilizada contém respostas ao impulso medidas em um ambiente real, sendo este uma sala de reuniões. Estes canais são mostrados na Fig. 4.1 e são os mesmos utilizados em (SERVIÈRE; PHAM, 2006). Além disso, ambos conjuntos utilizam o parâmetro de largura de banda de suavização $M = 20$, com exceção do método (PHAM; SERVIÈRE; BOUMARAF, 2003c) com modificações propostas por (NION et al., 2010), já que a não aplicação da suavização das respostas de frequência do perfis das fontes gera resultados mais consistentes neste método (especialmente na computação dos centroides). As Figuras 4.2, 4.4 e 4.6 mostram resultados considerando o comprimento dos blocos no tempo $N = 2048$. Os resultados nas Figuras 4.3, 4.5 e 4.7 consideram $N = 4096$. Logo, a diferença entre os dois grupos de simulações está no tamanho do bloco N (e consequentemente no número de componentes de frequência $F = N/2 + 1$).

As Figuras. 4.2 e 4.3 mostram o tempo de cálculo considerando apenas o segundo estágio da correção para a ambiguidade de permutação. Em ambas, os tempos de cálculo para diferentes durações dos sinais das fontes são comparados entre o segundo estágio descrito em (SERVIÈRE; PHAM, 2006) e o método proposto. Os resultados mostrados na Fig. 4.2 consideram $N = 2048$, enquanto na Fig. 4.3, $N = 4096$ é usado.

Nas Figuras 4.4 e 4.5, o tempo total de cálculo para o método proposto, considerando a solução completa de correção da ambiguidade de permutação, é comparada entre (SERVIÈRE; PHAM,

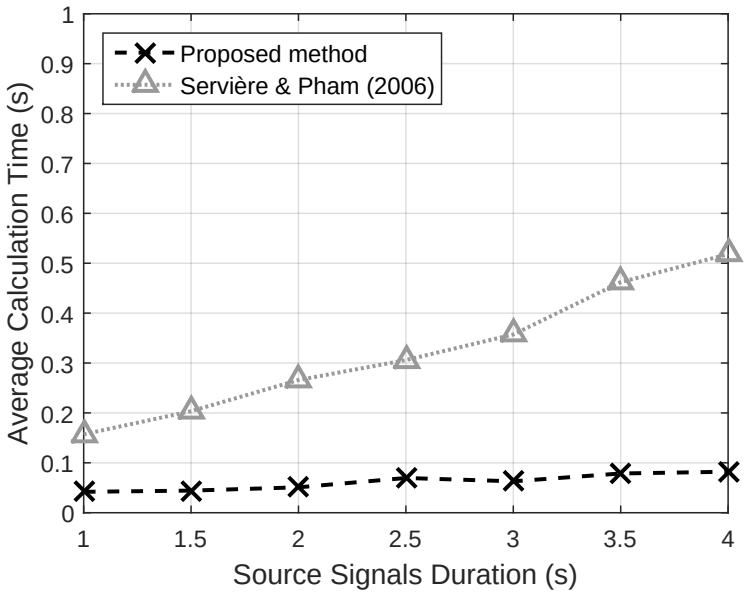

Figura 4.2: Tempo de cálculo considerando apenas o segundo estágio da correção da ambiguidade da permutação, para diferentes durações dos sinais das fontes. Para: $N = 2048$.

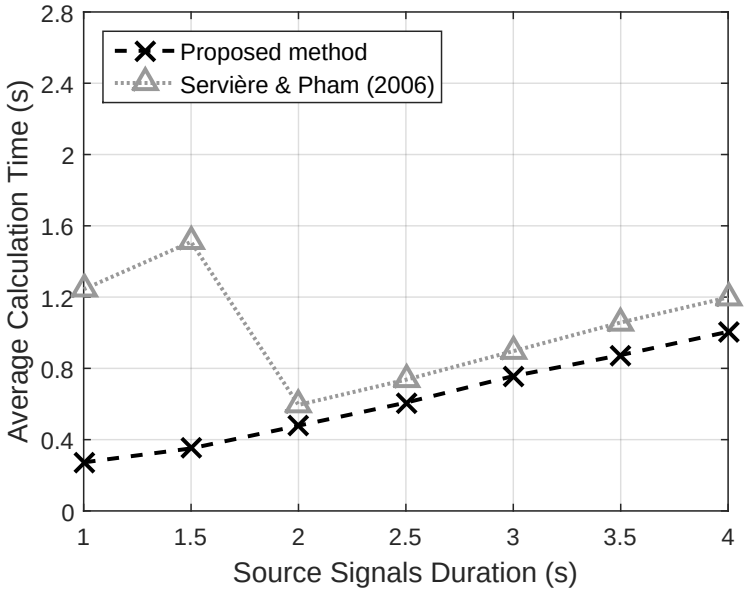

Figura 4.3: Tempo de cálculo considerando apenas o segundo estágio da correção da ambiguidade da permutação, para diferentes durações dos sinais das fontes. Para: $N = 4096$

2006) e (PHAM; SERVIÈRE; BOUMARAF, 2003c) com modificações propostas por (NION et al., 2010). De forma análoga, na primeira figura mencionada $N = 2048$ e na segunda $N = 2096$.

Finalmente, nas Figuras 4.6 e 4.7 é mostrado o número de componentes de frequência corretamente alinhados com relação ao número total de componentes F, indicando o percentual de sucesso (NION et al., 2010) da correção da ambiguidade de permutação para diferentes durações dos sinais das fontes. Note que apenas o problema da ambiguidade de permutação dependente da frequência é considerado pelo métodos mencionados aqui. O que significa que, mesmo após o problema ser solucionado, ainda há uma permutação global das fontes nas saídas do sistema de

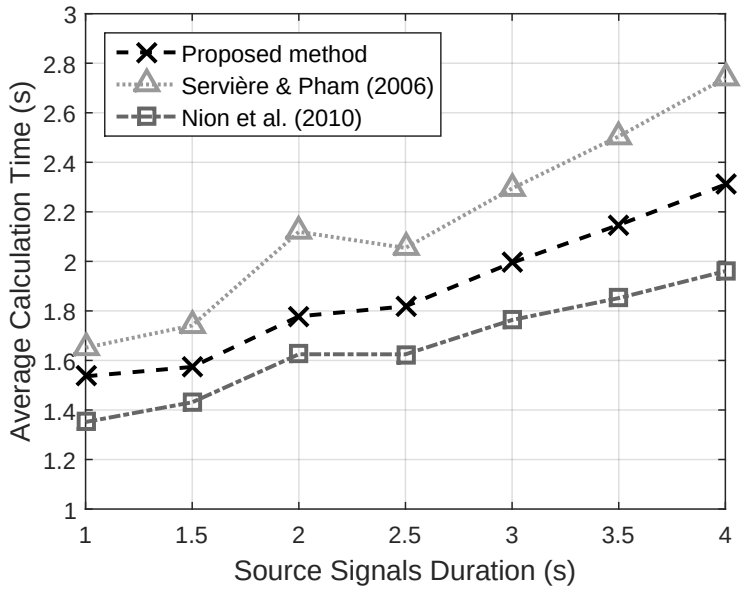

Figura 4.4: Tempo total de cálculo considerando a solução completa da correção da ambiguidade da permutação, para diferentes durações dos sinais das fontes. Para $N = 2048$.

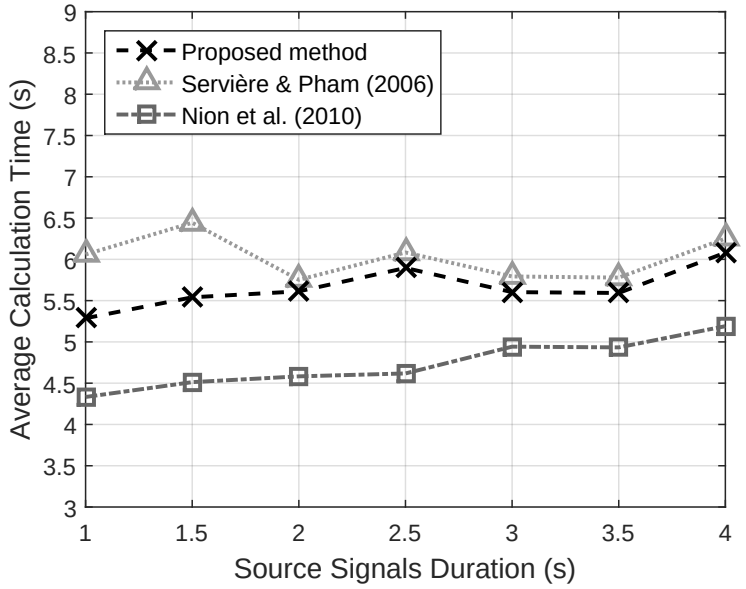

Figura 4.5: Tempo total de cálculo considerando a solução completa da correção da ambiguidade da permutação, para diferentes durações dos sinais das fontes. Para $N = 4096$.

separação. Assim, antes de computar o percentual de sucesso, um procedimento é realizado para remover essa permutação global. A Fig. 4.6, mostra resultados considerando o comprimento dos blocos no tempo $N = 2048$. Os resultados na Fig. 4.7 consideram $N = 4096$.

4.2 ANÁLISE DOS RESULTADOS

As Figuras 4.2 e 4.3 indicam que o segundo estágio proposto apresenta menores tempos de cálculo do que o segundo estágio em (SERVIÈRE; PHAM, 2006). Entretanto, como pode ser

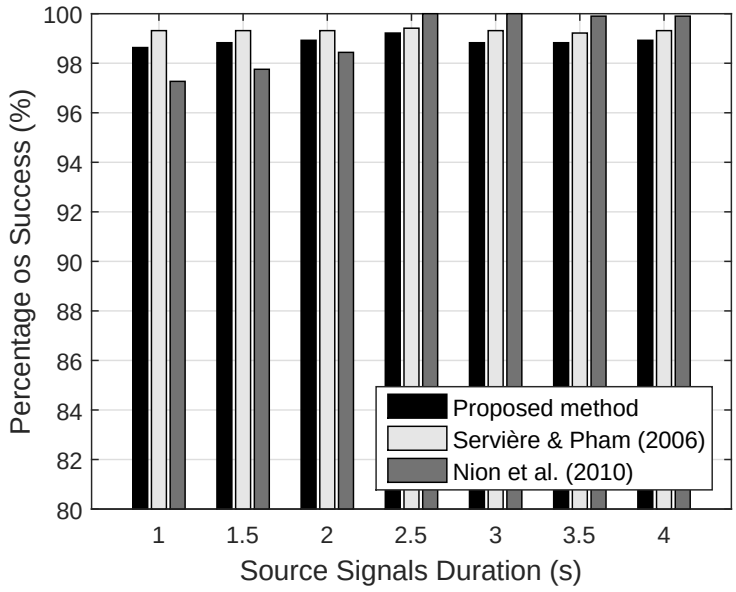

Figura 4.6: Percentage of success: number of frequency bins correctly aligned over total number of frequency bins. For $N = 2048$.

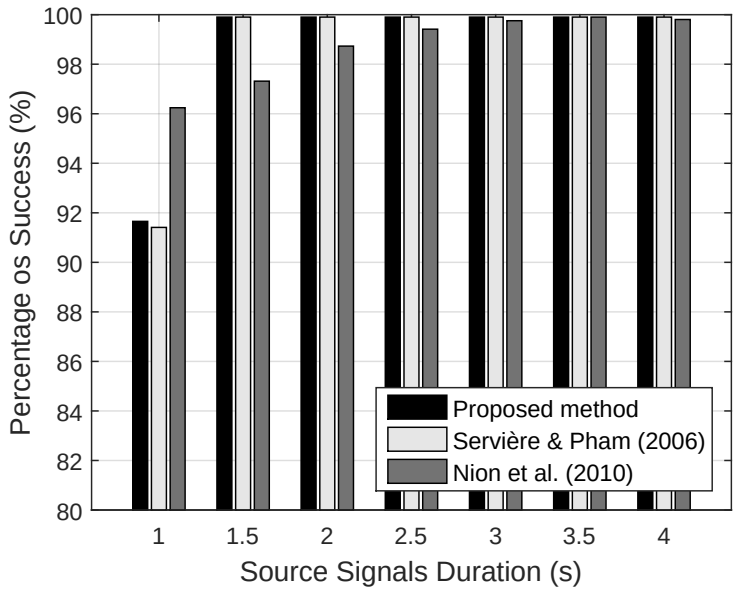

Figura 4.7: Percentage of success: number of frequency bins correctly aligned over total number of frequency bins. For $N = 4096$.

inferido da Fig. 4.3, a diferença entre os métodos tende a diminuir para quando o comprimento N dos blocos, e consequentemente número de componentes de frequências F, têm tamanho considerável.

Comparando os tempos de cálculo para o segundo estágio (apenas), mostrados nas Figuras 4.2 e 4.3, com o tempo total de cálculo para a solução completa, mostrados nas Figuras 4.4 e 4.5, podemos notar que o primeiro estágio da correção domina em termos de esforço computacional. Isso está relacionado com o fato de que a grande maioria das permutações são corrigidas neste estágio. Entretanto, a técnica em sí tem baixa complexidade computacional. Na verdade, o maior

esforço acontece após as permutações serem encontradas, quando as linhas da matriz inversa dos canais estimada $\hat{\mathbf{W}}_{\mathrm{ajd}}(f)$ são permutadas de acordo com as matrizes de permutação encontradas. Este processo, descrito no último parágrafo da seção 3.2.1 (página 23), é utilizado em todos os métodos descritos aqui para atualização das matrizes inversas dos canais. Não obstante, esse processo pode ser otimizado, aumentado os ganhos computacionais.

Nas figuras 4.3 e 4.5, a curva cinza apresenta um aumento peculiar no tempo de cálculo para curtas durações da sinais das fontes. O comportamento aparece quando a energia dos sinais das fontes não varia o suficiente com o tempo, em determinados componentes de frequência. Neste caso $\sigma^2_{D_2}(f) > \sigma^2_{D_1}(f)$ em frequências onde permutações não acontecem, prevenindo o algoritmo de parar o processo iterativo da forma esperada. Assim, um número máximo de iterações, arbitrariamente definido, é alcançado algum tempo depois. Entretanto, como mencionado na seção 3.3 (página 24), e como mostrado pela Fig. 4.7, em cenários como este, a solução para a correção das permutações de (SERVIÈRE; PHAM, 2006) ainda é capaz de apresentar um alto percentual de sucesso. As Figuras. 4.3 e 4.7 indicam que o método proposto neste trabalho é mais tolerante, sendo capaz de fornecer resultados similares sem comprometer o tempo de cálculo. O que significa que ele é uma alternativa ao método de (SERVIÈRE; PHAM, 2006).

Através de observações de resultados de diferentes simulações, e apesar do fato de que nos experimentos aqui documentados $m = 1$ é utilizado, o método proposto se mostra robusto também para pequenas variações de m. Além disso, foi observado que o parâmetro de largura de banda de suavização M tem melhores resultados entre 15 e 25 pontos. Se M for muito pequeno, os perfis das fontes não variam de forma suave o suficiente com a frequência, como esperado pelas premissas do método, e picos em $\sigma^2_{D_2}$ relacionados a permutações são indistinguíveis de outros pontos de máximo locais criados de forma aleatória. Se M é muito grande, permutações entre componentes de frequências próximos podem não ser detectadas.

Finalmente, cabe ressaltar que nestes experimentos, caso o segundo estágio (dos métodos que o utilizam) não seja empregado, os percentuais de sucesso caem para valores em torno de 80%, não sendo aceitáveis. Isso ocorre principalmente devido a reflexões com alta energia nas respostas ao impulso utilizadas, quebrando o algoritmo do primeiro estágio em algumas frequências. Ao atenuar essas reflexões (o que não é exatamente viável em aplicações práticas), a correção do primeiro estágio é bem sucedida, como mostrado em (PHAM; SERVIÈRE; BOUMARAF, 2003b).

4.2.1 Considerações sobre dois métodos alternativos

Por fim, é importante mencionar que outros dois métodos alternativos baseados no segundo estágio (SERVIÈRE; PHAM, 2006) foram encontrados. Estas abordagens são conseguidas através de rápidas modificações do método original, funcionando tão bem quanto ele, porém com melhor performance em termos de tempo de cálculo. Todavia, eles não foram extensivamente testados. Os métodos são brevemente descritos a seguir.

Para o primeiro método, procedemos como em (SERVIÈRE; PHAM, 2006), entretanto, no processo iterativo, ao invés de procurar por um valor de mínimo global em $\sigma^2_{D_1}(f)$, aproveitamos os pontos onde as curvas geradas por $\sigma^2_{D_1}(f)$ e $\sigma^2_{D_2}(f)$ se cruzam para definir zonas de busca. Essas zonas são delimitadas pelas seção onde $\sigma^2_{D_2}(f) > \sigma^2_{D_1}(f)$, de forma análoga ao descrito na subseção 3.3.1. Assim, diferentemente de (SERVIÈRE; PHAM, 2006) - onde apenas uma permutação é corrigida por iteração - procuramos pelo ponto de mínimo global de $\sigma^2_{D_1}(f)$ em cada zona, de forma que várias frequências onde permutações ocorrem são encontradas em uma iteração.

Alternativamente, podemos proceder como o proposto neste trabalho, entretanto corrigindo apenas uma permutação por iteração. Isso é feito, em cada iteração, procurando pelo ponto de máximo global de $\sigma^2_{D_2}(f)$, sem a definição de zonas de frequência mencionadas na subseção 3.3.1. Veja que, neste caso também não computamos $\sigma^2_{D_1}(f)$. A existência de permutações remanescentes é determinada como descrito na subseção 3.3.2, através da verificação da variação da média de $\sigma^2_{D_2}(f)$ entre iterações. Assim, o método funciona fundamentalmente como (SERVIÈRE; PHAM, 2006), todavia evitando o cálculo de $\sigma^2_{D_1}(f)$ ou de um valor de limiar como descrito na subseção 3.3.3. Isto diminui o custo computacional e aumenta a robustez do algoritmo, já que neste caso picos em $\sigma^2_{D_2}(f)$ relacionados a permutações podem estar tão perto quanto possível de pontos de máximo aleatórios em $\sigma^2_{D_2}(f)$. Além disso, como $\sigma^2_{D_1}(f)$ não é computado, não há possibilidade de o processo iterativo não parar.

5 CONCLUSÕES

Neste trabalho foi proposto um método para solução da ambiguidade de permutação na separação de misturas convolutivas de sinais de fala. Trata-se, principalmente, de uma alternativa ao segundo estágio da solução no estado da arte proposta por (SERVIÈRE; PHAM, 2006). É mostrado que o algoritmo apresenta boa performance sob as mesmas premissas do método original no qual ele é baseado, tais como; não estacionaridade dos sinais das fontes, conferindo variação suficiente das energias dos sinais com o tempo, para seus diferentes componentes de frequência; não correlação mútua entre os sinais das fontes; e não variação dos canais de mistura durante as gravações dos sinais no ambiente acústico. O método proposto se mostra de menor complexidade computacional, sendo também mais tolerante a variações na premissa sobre a estacionaridade dos sinais das fontes. Para validações dos resultados, o método proposto é comparado não só com a técnica original mas também o outro método no estado da arte (NION et al., 2010). Finalmente, neste trabalho também foram elencadas duas abordagens alternativas ao método de (SERVIÈRE; PHAM, 2006), considerando pequenas modificações nestes para fins de ganhos em termos de custo computacional.

Finalmente, podemos ressaltar que apesar do método proposto apresentar resultados comparáveis a (NION et al., 2010), este contém menos parâmetros livres a serem definidos por um possível usuário. Por exemplo, não é necessário definir um valor para o parâmetro de largura de banda de suavização M utilizado para cálculo de $F_y(f, b, i)$. Isso faz a técnica ser menos subjetiva e mais automatizada, sendo menos propensa a definições iniciais incorretas destes parâmetros.

5.1 SUGESTÕES DE TRABALHOS FUTUROS

Como sugestões para continuação deste estudo e de trabalhos futuros, podemos elencar os seguintes pontos a serem abordados:

1. Realização de testes mais detalhados utilizando os métodos alternativos citados na seção 4.2.1. Em especial, a técnica que parece ser a mais robusta e mesmo assim com tempos de cálculo comparáveis ao método principal detalhado neste trabalho. Esta utiliza apenas as dispersões $\sigma^2_{D_2}(f)$, encontra apenas uma permutação por iteração através do ponto de máximo global de $\sigma^2_{D_2}(f)$ e utiliza a variação da média para detecção de permutações existentes.

2. Implementação e testes para cálculo do limiar utilizando uma reta que passa pelos pontos iniciais e finais de $\sigma^2_{D_2}(f)$, ao invés da soma cumulativa normalizada $C(n_f)$ descrita neste trabalho, com intuito de detectar o ponto de maior deflexão de $\sigma^2_{D_2,sort}(f)$.

3. Expansão do algoritmo de (SERVIÈRE; PHAM, 2006) implementados, bem como dos mé-

todos neste trabalho, para correção das permutações para $I > 2$ (um maior número de fontes). Neste trabalho, experimentos foram realizados para $I = 2$.

4. Verificar a influência de uma estimativa errada do tamanho L para os canais de mistura. Em especial, utilizando um comprimento para os canais estimados mais curto do que o efetivamente utilizado na mistura. Dessa forma, deixando de compreender reflexões com alta e baixa intensidade, fazendo uma análise comparativa da relação sinal-ruído em cada caso. Para isso, canais de separação conhecidos podem ser utilizados em primeira instância. Ou seja, os sinais das fontes podem ser estimados através de versões truncadas dos canais de mistura utilizados. Assim, inicialmente uma avaliação ideal é realizada para determinar o potencial de queda da relação sinal ruído nos sinais estimados ao utilizar-se canais de separação com tamanhos menores.

5. Avaliação dos mapeamentos dos sinais para o domínio da frequência e de volta para o domínio do tempo, apenas. Isso visa avaliar qualitativamente esses processos, como a aplicação da janela de Hann e a sobreposição de blocos consecutivos, de forma que se possa medir a influência deles sobre os sinais no final do processo (em princípio, assumindo separação ideal).

REFERÊNCIAS BIBLIOGRÁFICAS

CHRISTOPOULOS, D. T. Developing methods for identifying the inflection point of a convex/concave curve. p. 10, 2012. Disponível em: <http://arxiv.org/abs/1206.5478>.

DENK, F.; COSTA, J. P. C. L. da; SILVEIRA, M. A. Enhanced Forensic Multiple Speaker Recognition in the Presence of Coloured Noise. *8th International Conference on Signal Processing and Communication Systems, ICSPCS 2014, 2014, Gold Coast, Australia*, 2014.

GOLUB, G. H.; Van Loan, C. F. *Matrix Computations*. [S.l.: s.n.], 1996. v. 10. 48 p. ISSN 00036935. ISBN 0801854148.

KINSLER, L. E.; FREY, A. R.; COPPENS, A. B.; SANDERS, J. V. *Fundamentals of Acoustics*. 4. ed. [S.l.]: John Wiley & Sons, 2010.

MATSUOKA, K. Minimal distortion principle for blind source separation. *Proceedings of the 41st SICE Annual Conference. SICE 2002.*, v. 4, p. 5–7, 2002.

MATSUOKA, K. Elimination of filtering indeterminacy in blind source separation. *Neurocomputing*, v. 71, n. 10-12, p. 2113–2126, 2008. ISSN 09252312. Disponível em: <http://www.elsevier.com/locate/neucom>.

MIRANDA, R. K. Métodos para melhoria da qualidade de separação cega de fontes sonoras em ângulos oblíquos de radiação. 2013.

NION, D.; MOKIOS, K. N.; SIDIROPOULOS, N. D.; POTAMIANOS, A. Batch and adaptive PARAFAC-based blind separation of convolutive speech mixtures. *IEEE Transactions on Audio, Speech and Language Processing*, v. 18, n. 6, p. 1193–1207, 2010. ISSN 15587916.

PARRA, L.; SPENCE, C. Convolutive blind separation of non-stationary sources. *IEEE Transactions on Speech and Audio Processing*, v. 8, n. 3, p. 320–327, 2000. ISSN 10636676.

PEDERSEN, M. S. Source Separation for Hearing Aid Applications. *Ph.D. dissertation, Informatics and Mathematical Modelling, Technical University of Denmark, DTU*, 2006. Disponível em: <http://eprints.pascal-network.org/archive/00001490/>.

PEDERSEN, M. S.; LARSEN, J.; KJEMS, U.; PARRA, L. C. A Survey of Convolutive Blind Source Separation Methods. *Springer Handbook on Speech Processing and Speech Communication*, Springer, p. 1–34, 2007.

PHAM, D.-T.; CARDOSO, J.-F. Blind Separation of Instantaneous Mixtures of Non Stationary Sources. *Signal Processing*, v. 81, n. 9, p. 855–870, 2001. ISSN 1053587X.

PHAM, D.-T.; SERVIÈRE, C.; BOUMARAF, H. Blind separation of convolutive audio mixtures using nonstationarity. *Proceeding of ICA 2003 Conference*, p. 975–980, 2003b. Disponível em: <http://bsp.teithe.gr/members/downloads/bssaudio/references/sepa-audioR.pdf>.

PHAM, D.-T.; SERVIÈRE, C.; BOUMARAF, H. Blind Separation of Speech Mixtures Based on Nonstationarity. *Proceeding of ISSPA 2003 Conference.*, 2003c. Disponível em: <http://www-ljk.imag.fr/membres/Dinh-Tuan.Pham/BSS/sepa-speech.pdf>.

QUINLAN, A.; BARBOT, J. P.; LARZABAL, P.; HAARDT, M. Model order selection for short data: An Exponential Fitting Test (EFT). *Eurasip Journal on Advances in Signal Processing*, v. 2007, 2007. ISSN 11108657.

SCHOBBEN, D. W. E. *Real-time Adaptive Concepts in Acoustics*: Blind signal separation and multichannel echo cancellation. 1. ed. [S.l.]: Kluwer Academic Publishers, 2001.

SERVIÈRE, C.; PHAM, D. T. Permutation correction in the frequency domain in blind separation of speech mixtures. *Eurasip Journal on Applied Signal Processing*, v. 2006, p. 1–16, 2006. ISSN 11108657.

SILVEIRA, M. A.; SCHROEDER, C. P.; COSTA, J. P. C. L. da; OLIVEIRA, C. G. de; Apolinário Junior, J. A.; SERRANO, A. M. R.; QUINTILIANO, P.; de Sousa Júnior, R. T. Convolutive ICA-Based Forensic Speaker Identification Using Mel Frequency Cepstral Coefficients and Gaussian Mixture Models. *The International Journal of Forensic Computer Science*, v. 8, n. 1, p. 27–34, 2013. ISSN 18099807. Disponível em: <http://www.ijofcs.org/abstract-v08n1-pp04.html>.

WAX, M.; KAILATH, T. Detection of signals by information theoretic criteria. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, v. 33, n. 2, p. 387–392, 1985. ISSN 0096-3518.

YEREDOR, A. Non-orthogonal joint diagonalization in the least-squares sense with application in blind source separation. *IEEE Transactions on Signal Processing*, v. 50, n. 7, p. 1545–1553, 2002. ISSN 1053587X.

APÊNDICES

I.1 CÓDIGOS EM MATLAB® RELACIOANDOS AO MÉTODO PROPOSTO

Códigos para solução da ambiguidade da permutação de acordo com (PHAM; SERVIÈRE; BOUMARAF, 2003b)

A função é chamada através do arquivo: “`solve_permutation_SerPh2003b.m`". Arquivo único que não dependende de programas externos.

```
function [Gf_hat_perm,permFreqs] = solve_permutation_SerPh2003b(Hf_hat,Gf_hat)
%% This function solves the permutation ambiguity problem.
% This is a typical problem in blind source separation (BSS) of convolutive speech mixtures
% in the frequency domain.
%
% According to:
%
%   D. Pham, C. Serviere, H. Boumaraf: BLIND SEPARATION OF CONVOLUTIVE AUDIO MIXTURES USING
%   NONSTATIONARITY. Proceeding of ICA 2003. Conference, Nara, Japan April 2003(b). Available
%   at http://ljk.imag.fr/membres/Dinh-Tuan.Pham/ on 19/12/2015.
%
%   The method rely on the continuity of the frequency response H(f) of the mixing filter to
%   solve the permutation ambiguity.
%
% Inputs:
%   Hf_hat: estimated mixing channel (tensor Hf_hat of size FxJxI, where F = #Frequency bins;
%           J = #Microphones; and I = #Sources. Note that #sources must be equal #mics (I=J=K),
%           for any J => H(f) must be square...
%   Gf_hat: OPTIONAL. If not provided, it will be calculated (inv(Hf_hat(f))). Size F x I x J.
%           This is the (estimated) inverse channel matrix.
%
```

```
% Output:
%   Hf_hat and Gf_hat after permutation correction (same dimensions as the inputs).
%   allFreqsFound: row vector with the frequencies where permutation jumps where found.
%
% *For now assuming Hf_hat is non-singular.

%% Check input arguments and put them in a more suitable format:

% Check if J == I, as expected:
if size(Hf_hat,2) ~= size(Hf_hat,3)
error('This method assumes the number of microphones and sources are equal.')
end

% Get key dimensions from input arguments:
[F,J,I] = size(Hf_hat);                     % #frequency bins F, #microphones J, #sources.

% Put Hf_hat in a more suitbale format for the calculations that will come => J x I x F:
Hf_hat = permute(Hf_hat,[2 3 1]);

% Check if Gf_hat was provided as input argument:
switch nargin
% If not, calculate Gf_hat from Hf_hat:
case 1
Gf_hat = zeros(I,J,F);              % Initialize Gf_hat. Size I x J x F.
for f = 1:F                         % For each freq. bin, compute Gf_hat = Hf_hat^(-1):
Gf_hat(:,:,f) = inv(Hf_hat(:,:,f));
end
```

```
% If it was provided, just reshape it to make it better suitable:
case 2
Gf_hat = permute(Gf_hat,[2 3 1]);   % Now, size I x J x F (previously: F x I x J).
end

%% (i) Compute the I x I matrices R(f,f-1) = Gf_hat(f).Hf_hat(f-1) for each frequency bin:
% The data will be stored in a tensor (array) of size I x I x F. Note that the first frequency
% bin has a dumb value (and will not be used). We just add it to avoid shifting the
% frequencies by 1 bin, since R(f,f-1) actually starts at f = 2... (Hf_hat(1-1) does not exist).
R = zeros(I,I,F);                           % Initialize R.
for f = 2:F
R(:,:,f) = Gf_hat(:,:,f)*Hf_hat(:,:,f-1);
end

%% Check for what frequencies f we can skip the search for all possible permutations:
R_abs = abs(R);                 % Absolute value of R (|R(f,f-a)|). Size I x I x F.

%%%
% Get the index of the element of highest magnitude of each row of |R(f,f1)| and store it in
% the permutationIndices matrix (size I x F), for each frequency bin:
[~,permIndices_f] = max(R_abs,[],2);
permIndices_f = squeeze(permIndices_f);     % I x F

%%%
% Determine if, for each frequency bin, the permutation indices found are distinct (if there
% are no repeated indices) and thus define precisely the permutation we are looking for.
% Note: the sentence below (as it is written) is not very intuitive, but it is really fast!
```

```
areIndicesDistinct_f = all(diff(sort(permIndices_f)),1);    % 0 or 1, for each f. Size 1 x F.

%%%
% If, for a frequency bin, the permutation indices are all different from each other, then
% they already define the permutation matrix. Therefore, for these frequencies, we can skip
% the search for all possible permutations.

%%%
% Nonetheless, in this group, we still need to check which of these permutation matrices are
% not the identity matrix. If, for a frequency f, the permIndices define an identity matrix, we
% do not need to permute the channel at f. For all other frequency bins in this group, we then
% have the indices that define the permutation matrices we were looking for.

freqs_indAreDistinct = find(areIndicesDistinct_f); % Frequencies where the permutation
% indices are distinct from each other.
nFreqIndDist = length(freqs_indAreDistinct);       % Number of frequency bins where permutation
% indices that are distinct from each other.
eyeIndices = (1:I).';                              % Vector I x 1 with indices that can build
% an identity matrix.

% Matrix I x nFreqIndDist (#freqs. where indices are distinct) that holds (replicates) the
% indices for the identity matrix for each frequency.
eyeIndicesRep = repmat(eyeIndices,1,nFreqIndDist);

% In this group, where permutation indices are distinct, check for what frequencies the
% permutation indices are different from the identity matrix indices. This is the first
% set of frequencies found, where permutation corrections occur:
```

```
id = find(any(permIndices_f(:,freqs_indAreDistinct) ~= eyeIndicesRep,1));
permFreqs = freqs_indAreDistinct(id).';

% Finally (for this step) neglect the first frequency bin, if it was collected previsouly:
permFreqs = permFreqs(permFreqs~=1);
nPermFreqs = length(permFreqs);

%%%
% Along with the frequencies where the permutations occur (permFreqs), store also the
% corresponding permutation matrices (permFreqs_Pi):
permFreqs_Pi = zeros(I,I,nPermFreqs);   % Init.
% Assemble matrices permFreqs_Pi:
for fl = 1:nPermFreqs
for j = 1:I
permFreqs_Pi(j,permIndices_f(j,permFreqs(fl)),fl) = 1;
end
end

%% (ii) For the remaining freqs., determine the permutation such that diag(abs(R)) is maximum:

%%%
% Assemble all possible permutation matrices that will be used for the verification:
allPermIndices = perms(1:I);    % Indices for all possible permutations (size I! x I).
nPerms = factorial(I);          % Total number of possible permutations.
Pi = zeros(I,I,nPerms);         % All possible permutation matrices as an array I x I x nPerms.

for i = 1:nPerms
```

```
% Mount the permutation matrix P from the permutation indices especified in the i'th
% row of the matrix allPermIndices.
for j = 1:I
Pi(j,allPermIndices(i,j),i) = 1;
end
end

%%%
% Get remaining frequencies for which this process will be done (those where the permutation
% indices are not distinct from each other):
freqs_indAreNotDistinct = setdiff(1:F,freqs_indAreDistinct);

%%%
% Run iterations for each (remaining) frequency bin:
for f = freqs_indAreNotDistinct

% If this is the first frequency bin, just ignore it:
if f == 1
continue
end

%%%
% Look for maximum product in (ii) among all possible permutations:
maxProduct = 0;              % Reset maximum product (when going to the next frequency bin).

for i = 1:nPerms             % For all possible permutations...
% Calcualte Pi(i)*|R(f,f-1)| for this iteration (same as permuting G(f) by Pi(i)
```

```
% and then calculating R again):
R_abs_i = Pi(:,:,i)*R_abs(:,:,f);

% Calculate product of the magnitude of the diagonal elements of R(f,f-1) and check
% if this permutation provides the maximum product value among all other permutations:
product = prod(diag(R_abs_i));

if maxProduct < product
maxProduct = product;  % Update max product found:
bestPi = Pi(:,:,i);        % Update best permutation found
end

end % end for i = 1:numberOfPermutations

% Check if the permutation matrix Pi found is different from the identity matrix. If so,
% store current frequency bin value and bestPi(i)...
if bestPi ~= eye(I)
permFreqs = [permFreqs; f];
nPermFreqs = nPermFreqs + 1;
permFreqs_Pi(:,:,nPermFreqs) = bestPi;
end

end % end for f = freqs_indAreNotDistinct

%% (iii) Permute rows of G(f)_hat according to the perm frequencies and Pi matrices found:

Gf_hat_perm = Gf_hat; % Initiliaze Gf_hat_perm (I x J x F).
```

```
for i = 1:nPermFreqs
for f = permFreqs(i):F
Gf_hat_perm(:,:,f) = permFreqs_Pi(:,:,i)*Gf_hat_perm(:,:,f);
% FOR TEST PURPOSES ONLY:
% permFreqs_Pi(:,:,i)
% Gf_hat(:,:,f)
% Gf_hat_perm(:,:,f)
end
end

% Put G(f) in the desirable final format (size F x I x J):
Gf_hat_perm = permute(Gf_hat_perm,[3 1 2]);

%%%
% Also permute the columns of H(f)_hat according to Pi(f)^-1 ?
% Hf_hat_perm = Hf_hat; % Initiliaze Hf_hat_perm (J x I x F).
% for i = 1:nPermFreqs
%     for f = permFreqs(i):F
%         Hf_hat_perm(:,:,f) = Hf_hat_perm(:,:,f)/permFreqs_Pi(:,:,i); % H(f)*Pi(f)^(-1)
%         % FOR TEST PURPOSES ONLY:
% %         permFreqs_Pi(:,:,i)
% %         Hf_hat(:,:,f)
% %         Hf_hat_perm(:,:,f)
%     end
% end
%
% % Put H(f) in the desirable final format (size F x J x I):
```

```
% Hf_hat_perm = permute(Hf_hat_perm,[3 1 2]);

end % end function.
```

Códigos para o segundo estágio da solução da ambiguidade da permutação de acordo com o método proposto

A função é chamada através do arquivo: “`solve_permutation_SerPh2006_2ndStage_Lima.m`". Todos os outros 4 scripts mencionados nessa seção são necessários para que essa função seja executada.

Função principal

Arquivo “`solve_permutation_SerPh2006_2ndStage_Lima.m`":


```
function [Wf_perm,allFreqsFound,finalDispD2] = ...
solve_permutation_SerPh2006_2ndStage_Lima(xt,N,m,overlap,Wf,M,maxIter)
%% This function solves the permutation ambiguity problem according to Ser, Pham and Lima:
% This performs the SECOND STAGE of the correction, proposed in Pham s paper, but also
% considering the enhancements/modifications by Pedro Lima described below.
%
% It iss based on the continuity of the time-varying spectrums of the reconstructed sources.
% For the first stage, based on the continuity of the unmixinfg filter Wf,
% see solve_permutation_SerPh2003b.m. In general, this must be applied after the first stage.
%
%   D.-T. Pham,and Ch. Serviere, "Permutation correction in the frequency domain in blind
%   separation of speech mixtures", in Eurasip Journal on Applied Signal Processing, Volume
%   2006.
%
```

```
%   AND %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Pedro Lima: enhancements on the method from Serviere and Pham are done the following way:
%       1. Only a single dispersion curve: \sigma^2_{D_2}(f) (dispersion of D_2(f,k).
%          Therefore, there is no need to compute F_y(f,k,i), D_1(f,k) and \sigma^2_{D_1}(f).
%       2. The permutation peaks are detected from dispersion of D_2(f,k) based on a calculated
%          threshold. All permutations are corrected in a much smaller number of iterations
%          (1, 2, or so...), since detection zones are found and all peaks in these zones are
%          corrected in a single permutation.
%          I expected better performance with little robustness loss as compared
%          to Pham s method.
%
%   See comments on functions below for more info about input arguments.
%
%  Outputs:
%   Wf_perm:     Wf fixed (with permutations solved accordingly). These are the unmixing
%                filters. Here, a tensor of size "nFreqBins x I x J", I is the number of
%                sources and J the number of microphones. W(f) is an I by J matrix. nFreqBins
%                considering single sided spectrums (only positive frequencies: nF = N/2+1)
%
%   freqsFound:  List of permutations jumps found (frequencies where there are permutations).
%   dispD2:      Dispersion of the differences D2(f,k) after permutation corection.

%% Compute parameters for first iteration:
global tic_2ndStage

[Sx_hat] = compute_Sx_hat(xt,N,m,overlap); % Estimate spectral matrix Sx_hat.
```

```
tic_2ndStage = tic;

% Compute centered source profiles (this also initialize Ec_perm, which will store Ec with
% permuted sources, as the permutations are found).
Ec_perm = compute_Ec(Wf,Sx_hat);

iIter = 0;                  % Initialize variable that will store the current iteration number.
allFreqsFound = 0;          % Initialize variable that will store all (index of the) points
% of minimum dispersion found.

%% Start iterations to detect and correct the frequencies where permutation jumps occur:
while iIter <= maxIter

iIter = iIter+1;        % increase iterations counter at each loop.
%disp(['Permutation detection, iteration ',num2str(iIter)]);

% (1) Compute Hy(f,k,i), D_2(f,k) and \sigma^2_{D_2}(f) and mean(\sigma^2_{D_2}(f)): -----

% First, store dispersion curve calculated in the last iteration.
if iIter > 1
lastDispD2 = dispD2;
end

Hy = compute_Hy(Ec_perm,M);          % Compute smoothed profiles with forced permutations.
dispD2 = compute_dispersion_D2(Hy); % Compute dispersion of D_2(f,k)
mu(iIter) = mean(dispD2);           % Compute dispersion mean.
```

```
% ------------------------------------------------------------------------------

% (2) If iIter > 1, check if indeed there were permutations to be corrected in last iter: -

% We correct the permutations in the first iteration assuming they exist - if they did not,
% we will know (the average will increase)...
if iIter > 1
if mu(iIter) > mu(iIter-1)
% If mean increased from last iteration to this one, then there were no
% permutations to be corrected (and the correction induced a peak which increased
% the mean). Therefore:

% Discard frequencies found in the last iteration:
allFreqsFound = allFreqsFound(1:length(allFreqsFound)-length(freqsFound));

% Return dispersion curve from last iteration (before latest correction):
finalDispD2 = lastDispD2;

break                % break while loop - no more permutatiosn to be found.
end
end

% ------------------------------------------------------------------------------
```

```
% (3) Find the permutation frequencies (peaks in the dispersion of D_2{f,k}+1): ----------

% (3.1): Calculate threshold (Th) which will separate the peaks from the noise:

a = sort(dispD2,'ascend');   % Sort dispersion values in ascending order.
%w = 25;
%a = fastsmooth(a,w,1,1);    % Smooth A over a bandwidth w (points).
b = cumsum(a);               % Integrate A.
b = b*(max(a)/max(b));       % Rescale intA to fit A.
c = b-a;
med = median(dispD2);
Th = max(c)+ med;

% Plot some data (for test purposes only)...
%     nFreqBins = length(dispD2);
%     x = 1:nFreqBins;
%     figure
%     plot(x,a,x,b);
%     legend('sorted dispD2','cumsum(sorted dispD2)')

% nFreqBins = length(dispD2);
% x = 1:nFreqBins;
% p1Color = [0.1 0.1 0.1];
% p2Color = [0.4 0.4 0.4];
% p3Color = [0.6 0.6 0.6];
% figure
% p=plot(x,a,x,b,'--',x,c,':','LineWidth',2);
```

```
% legend('\sigma^2_{D_2,sorted}(n_f)','C(n_f)','C(n_f) - \sigma^2_{D_2,sorted}(n_f)','Interpreter','late
% xlabel('Number of frequency bins','FontSize',14)
% ylabel('Dispersion values','FontSize',14)
% ylim([0 12])
% xlim([0 2048])
% set(legend,'FontSize',12,'Location','northwest')
% set(gcf,'render','painters')
% set(gca, 'FontSize',12)          % Ticks font size
% p(1).Color = p1Color;
% p(2).Color = p2Color;
% p(3).Color = p3Color;
% grid on

%     figure
%     plot(x,c);
%     legend('(sorted dispD2) - cumsum(sorted dispD2)')
%     figure
%     plot(x,dispD2,'k',x,mu(iIter)*ones(nFreqBins,1),'b',x,Th*ones(nFreqBins,1),'r',...
%          x,med*ones(nFreqBins,1),'g')
%     legend('dispD2','mean','threshold','median')

% Plot to generate picture...
%     figure
%     p1Color = [0.1 0.1 0.1];
%     p2Color = [0.1 0.1 0.1];
%     p3Color = [0 0 0];
%     p = plot(21:nFreqBins+20,dispD2,1:2048,Th*ones(2048,1),'--',...
```

```
%         1:2048,mu(iIter)*ones(2048,1),'-.','LineWidth',1);
%     legend('\sigma^2_{D_2}(f)','{\rm Threshold}','{\rm Mean of} \sigma^2_{D_2}(f) ','Interpreter','la
%     xlabel('Frequency bins','FontSize',14)
%     ylabel('Dispersion values','FontSize',14)
%     ylim([0 14])
%     xlim([0 2048])
%     set(legend,'FontSize',12)
%     set(gcf,'render','painters')
%     set(gca, 'FontSize',12)          % Ticks font size
%     p(1).Color = p1Color;
%     p(2).Color = p2Color;
%     p(3).Color = p3Color;
%     grid on

% (3.2) Separate the peaks above the threshold into different zones:

% Look for crossing zones (between dispD2 and Th):
% First check first element of dispD2:
nZones = 0;                         % Number of zones to be found in this iteration.
if dispD2(1) < Th                   % Check if the first element of dispD2 is below Th.
flag_dispD2_isBelow = 1;
else
nZones = nZones+1;              % Update number of zones found.
zoneIn(nZones) = 1;             % Mark zone begin.
flag_dispD2_isBelow = 0;        % Mark that dispD2 is above Th.
end
% Then check remaining elements:
```

```
for i = 2:length(dispD2)
if flag_dispD2_isBelow == 1 && dispD2(i) > Th
nZones = nZones+1;              % Update number of zones found.
zoneIn(nZones) = i-1;           % Mark zone begin.
flag_dispD2_isBelow = 0;        % Mark that curve dispD2 is above Th.
elseif flag_dispD2_isBelow == 0 && dispD2(i) < Th
zoneOut(nZones) =  i;           % Mark zone end.
flag_dispD2_isBelow = 1;        % Mark that curve dispD2 below Th.
end
end

% (3.1) Look for global maximum in each zone (of dispD2) found above:

freqsFound = 0;         % Initialize variable that will store (index of the) points
% of minimum dispersion found in this iteration.
for i = 1:nZones
[~,fl] = max(dispD2(zoneIn(i):zoneOut(i)));
fl = fl + zoneIn(i)-1;  % Adjust value, since maximum was looked for inside a
% zone/range of values.
fl = fl + M;            % The spectrum had 2*M (end) points removed during smoothing
% process (to calculate Fy). Hence, here we adjust fl to its
% actual value (as if no points were subtracted from spectrum).
fl = fl + 1;            % Finally, consider the fact that the dispersion peak will
% appear one data point before the actual permutation
% frequency - due to the way Hy is calculated (and impacts
% the dispersion of D2).
freqsFound(i) = fl;     % Store permutation frequencies.
```

```
end
allFreqsFound = [allFreqsFound freqsFound];

% ------------------------------------------------------------------------------


% (4) Permute sources in Ec(f,k,i) according to the permutation frequencies found: -------

Ec_perm = permute_Ec_outputs(Ec_perm,freqsFound); % Permute profiles (Ec)
% outputs for all frequencies
% higher than fl for new iteration.

% ------------------------------------------------------------------------------


% Just a caution meaure in case the algorithm does not stop...
% Check if maximum number of iterations has been reached:
if iIter == maxIter
disp(['Maximum number of iterations (',num2str(maxIter),') reached.'])
break
end


end % end while.

allFreqsFound = allFreqsFound(2:length(allFreqsFound));

```

```
% Permute the rows of the demixing channel W(f) for all frequencies higher than fl.
Wf_perm = permute_rows_of_Wf(Wf,allFreqsFound);

end % end function.
```

Função para estimação da matriz espectral variante no tempo dos sinais gravados $\hat{S}_x(b, f)$

Arquivo de função “`compute_Sx_hat.m`":

```
function [Sx_hat,nBlocks] = compute_Sx_hat(xt,N,m,overlap)
%%
% This function estimates $\mathbf{\hat{S}}_x(f,k)$, the $J \times J$ (time varying) spectral
% matrix of a (multivariate) signal $\mathbf{x}(t)=[x_1(t) \enspace ... \enspace x_J(t)]^T$.
%
%  According to:
%   D.-T. Pham,and Ch. Serviere, "Permutation correction in the frequency domain in blind
%   separation of speech mixtures", in Eurasip Journal on Applied Signal Processing, Volume
%   2006.
%
%  Inputs:
%    xt:        Observed sequences. Matrix of size "J x nTimeSamples", where J is the number
%               of microphones/signals. x(t)=[x_1(t) ... x_J(t)]^T (T denoting the transpose).
%
%    N:         Length of the consecutive time blocks the data sequence (x) will be subdivided
%               to. Preferably a power of 2 to take advantage of the Fast Fourier Transform.
%
%    m:         Number of periodograms to be averaged over time. Must be an odd positive
```

```
%                integer. If m = 1, no timeaverage is performed.
%
%    overlap:    Overlap between consecutive blocks (betwenn 0 and 1).
%
%  Outputs:
%    nBlocks:    Number of time blocks the sequence data (x) has been subdivided to. Calculated
%                based on provided block length (N) and overlap.
%
%    Sx_hat:     Estimate of the spectrum of a multivariate signal (time-varying spectral
%                matrix). Tensor of size "J x J x nBlocks x nFrequencyBins". If data is composed
%                of the observed sequences, the output is Sx_hat (spectral matrix of the
%                observed signals xt). But, In pactrice, the function will output the estimated
%                spectrum of any input (single or multivariate) signal xt. Only positive
%                frequencies are returned (single sided spectrum).
%
%  Author: Pedro Lima (04/2016).

%% Check input arguments:

% Check if "m" is a positive integer. Otherwise, display error.
if mod(m,1)
error('Input "m" must be a positive integer.')
end

% Check if "m" is odd. Otherwise, display error.
if rem(m,2)
% OK
```

```
else
error('Input "m" must be an odd number.')
end

% Check if the block length (N) is even. Otherwise, display error.
if rem(N,2)
error('Please provide an even block length "N". Preferably a power of 2.')
end

%% Subdivide x into consecutive (overlapping) blocks and apply a Hanning window to each block.

%%%
% Calculate number of data blocks based on desired block length (N) and overlap:
[J,nTimeSamples] = size(xt);                          % Get key dimensions from xt.
nBlocks = round((nTimeSamples-N)/((1-overlap)*N)) + 1;      % Calc. number of blocks.

if nBlocks < 1                                                % Check its validity.
error('The block length "N" must be less than the number of time samples of "xt".');
end

%%%
% Compute N-points Hann window. In the cited paper, the author did not divide by 2 as in the
% original formula for the Hann window. Hence, multipliying by 2 here so it matches the paper.
hannWindow = hann(N,'periodic')*2;

%%%
% Subdivide data into consecutive blocks, apply window to each block and store the data in the
```

```
% tensor xt_blocks:
xt_blocks = zeros(N,J,nBlocks);% Tensor to store the time blocks of xt. Size "N x J x nBlocks".
xt = xt.';                     % Put xt in a more suitable format for below (nTimeSamples x J).
blockEnd = round((0:nBlocks-1)*(nTimeSamples-N)/(nBlocks-1));         % End of each time block.
for i=nBlocks:-1:1
xt_blocks(:,:,i)=hannWindow(:,ones(J,1)).*xt(blockEnd(i)+1:blockEnd(i)+N,:);
end

%% Apply Fast Fourier Transform to each (windowed) data block:
xf_blocks = fft(xt_blocks);              % Tensor of size "N x J x nBlocks".
xf_blocks = permute(xf_blocks,[2 1 3]); % Put in a more suitable format
% "J x nFrequencyBins x nBlocks".

%% Compute the modifyed periodogram (periodogram using a Hanning window):
Px = zeros(J,J,nBlocks,N);             % Initialization of Px(k,f).
for k = 1:nBlocks                      % for each data block k, calculate the periodogram:
for f = 1:N
Px(:,:,k,f) = xf_blocks(:,f,k)*xf_blocks(:,f,k)'/(3*N/2);
% Px: tensor of size J x J x nBlocks x nFrequencyBins.
end
end
nFreqBins = N/2+1;                     % Number of frequency bins considering only positive
% frequencies (single sided spectrum).
Px = Px(:,:,:,1:nFreqBins);            % Keep only positive frequencies.
% Size "J x J x nBlocks x nFreqBins"

%% Average the periodogram over m consecutive blocks to get the estimated spectral matrix.
```

```
Px = permute(Px,[3 4 1 2]);     % Put in a more suitable format for the calculation
% below (nBlocks x nFrequencyBins x J x J).

if m == 1                       % if m = 1 skip average.
Sx_hat = Px;
else
Sx_hat = zeros(size(Px));   % Initialize tensor of size "nBlocks x nFrequencyBins x J x J".
for j1 = 1:J                        % For each signal...
for j2 = 1:J
for iFreq = 1:nFreqBins    % And each frequency bin...
% Compute average between "m" consecutive blocks (using fastsmooth.m):
Sx_hat(:,iFreq,j1,j2) = fastsmooth(Px(:,iFreq,j1,j2),m,1,0);
end
end % end for j2
end % end for j1
end

%%%
% Crop first and last $(m-1)/2$ points, which are zero. These blocks are neglected, since
% there are not enough points to compute the above average for them (see fastsmooth.m).
% Also, computing them could actually cause the curves $\sigma^2_{D_1}(f)$ and
% $\sigma^2_{D_2}(f)$, dispersions of $D_1(f,k)$ and $D_2(f,k)$ to touch each each other
% prematurely at low frequencies.

if m~=1
nBlocks = nBlocks - (m-1);
firstBlock = (m-1)/2 + 1;
```

```
lastBlock = firstBlock+nBlocks-1;
Sx_hat = Sx_hat(firstBlock:lastBlock,:,:,:); % nBlocks x nFrequencyBins x J x J
end

% Put Sx_hat in the desirable final format(size "J x J x nBlocks x nFrequencyBins)
Sx_hat = permute(Sx_hat,[3 4 1 2]);

end
```

Função para computação dos perfis centralizados das fontes $E'(f,b,i)$

Arquivo de função "`compute_Ec`":

```
function [Ec] = compute_Ec(Wf_hat,Sx_hat)
%% This function computes the centered source "profiles" E'(f,k,i) (Ec).
%
%  According to:
%    D.-T. Pham,and Ch. Serviere, "Permutation correction in the frequency domain in blind
%    separation of speech mixtures", in Eurasip Journal on Applied Signal Processing, Volume
%    2006; and
%
%    D. T. Pham, Ch. Serviere and H. Boumaraf, Blind separation of speech mixtures based on
%    nonstationarity, Proceeding of ISSPA (2003c) Conference, Paris, Japan July 2003.
%
%  Inputs:
%    Wf_hat:     Unmixing filters, tensor of size "nFrequencyBins x I x J", I is the number of
%                sources and J the number of microphones. W(f) is an I by J matrix. Assuming
```

```
%                only positive frequencies (single sided) are kept for W(f). That is, nFreqbins
%                = L/2+1, where L is the number of time lags of W(t).
%
%    Sx_hat:     Time varying spectral matrix of size "J x J x nBlocks x nFrequencyBins", where
%                nBlocks is the number of time blocks the source signals have been previously
%                subdived into. Also considering only positive frequencies.
%
%  Outputs:
%    Ec:         Centered source "profiles" E' as defined in the papers above. The array
%                has dimensions I x nBlocks x nFrequencyBins (single sided - only positive
%                frequencies).
%
%  Author: Pedro Lima (04/2016).

%% Get key dimensions from the input arguments and give them a check:
[~,I,J] = size(Wf_hat);
[~,~,nBlocks,nFreqBins] = size(Sx_hat);

% Check if the dimensions match:
if size(Wf_hat,1) ~= nFreqBins || size(Sx_hat,1) ~= J || size(Sx_hat,2) ~= J
error('Please check the dimensions of the input arguments. They do not match.')
end

%% Compute spectrum of the reconstructed sources: Sy_hat(t,f) = diag( G(f)Sx_hat(t,f)G'(f) ):
Wf_hat = permute(Wf_hat,[2 3 1]);   % Put Wf_hat in a more adequate format (I x J x nFreqBins).

Sy_hat = zeros(I,nBlocks,nFreqBins);% Initialize Sy_hat (I x nBlocks x nFreqBins)
```

```
for q = 1:nBlocks                          % For each data block q...
for f = 1:nFreqBins                      % And frequency bin f, calculate Sy_hat(t,f).
Sy_hat(:,q,f) = diag(  Wf_hat(:,:,f)*Sx_hat(:,:,q,f)*Wf_hat(:,:,f)'  );
end
end

%% Compute the source profiles E(f,k) defined as the logarithm of Sy_hat(t,f):
E = log(real(Sy_hat)); % Taking only real part - assuming img part is neglectable/null.
% Size: I x nBlocks x nFrequencyBins.

%% Compute the centered profiles: Ec(f,k,i) = E(f,k,i) - time average of E(f,k,i):
E = permute(E,[2 3 1]);            % Put E in a more adequate format to calculate the time mean.
% Size: nBlocks x nFreqBins x I.

E_mean = mean(E);                  % Compute mean (time average) of E(f,k,i) (1 x nFreqBins x I).
E_mean = permute(E_mean,[2 3 1]); % Put data in a more suitable format (nFreqBins x I x 1)
% to perform the loop below.
E = permute(E,[2 3 1]);            % Also put E in a more suitable format for below.
% (nFreqBins x I x nBlocks).

Ec = zeros(nFreqBins,I,nBlocks);  % Initialize Ec.
for iBlock = 1:nBlocks             % For each block, subtract the time average.
Ec(:,:,iBlock) = E(:,:,iBlock) - E_mean;
end

Ec = permute(Ec,[2 3 1]);         % Put E' back in the desired format (I x nBlocks x nFreqBins).
```

```
end
```

Função para computaçao do perfis suavizados com permutações impostas $H_y(f, b, i)$

Arquivo de função “compute_Hy":

```
function [Hy] = compute_Hy(Ec,M)
%% This function computes the smoothed source profiles Hy, which contains forced permutations.
%
%  According to:
%   D.-T. Pham,and Ch. Serviere, "Permutation correction in the frequency domain in blind
%   separation of speech mixtures", in Eurasip Journal on Applied Signal Processing, Volume
%   2006.
%   -> Equation (11).
%
%  For this, the rectangular or unweighted sliding-average smooth is used. It simply
%  replaces each point in the freq. resp. with the average of BW adjacent points, where BW is a
%  positive integer called the smooth width (bandwidth). However, in this case, the bins from
%  the right band are taken from the second source (forced permutation) for each fl.
%
%  Here, we crop the firsts/lasts M points (which are zero). These blocks are neglected,
%  since there are not enough points to compute the above average for them (see fastsmooth.m).
%  We could also use progressively smaller bandwidths to consider these points, but this
%  had a side effect in subsequent calculations (in the dispersions).
%
%  Input arguments:
```

```
%   Ec:          Centered source "profiles" E' as defined above in the paper above. The array
%                must have dimensions I x nBlocks x nFreqBins.
%   M:           Bandwidth parameter. Where the bandwidth BW has [fl-M:fl+M] data points. Given
%                fl the frequency bin which value will be replaced by the average of
%  Output:
%   Hy:          Smoothed profiles with forced permutations size (I x nBlocks x nFreqBins).

%% Get key dimensions from the input argument:
% I:             Number of sources;
% nBlocks:       Number of time blocks the source signals have been previously subdived into;
% nFreqBins:     Number of frequency bins of the time-varying spectrum.
[I,nBlocks,nFreqBins] = size(Ec);

%% Check input arguments:

if mod(M,1)      % Check if "m" is integer. Otherwise, display error.
error('Input "m" must be a non-negative integer.')
end

if M < 1         % Check if "m" is integer. Otherwise, display error.
error('M must be greater than 0.')
end

%%
% Compute Hy:
BW = 2*M+1;                      % Bandwidth(#consecutive bins for the average process).
```

```
Ec = permute(Ec,[3 1 2]);        % Put Ec in a more suitable format for the calculation
% below (nFreqBins x I x nBlocks).

Hy = zeros(nFreqBins,I,nBlocks); % Initialize Hy (nFreqBins x I x nBlocks).
for iBlock = 1:nBlocks
% Average the (time-varying) frequency responses over the bandwidth BW [fl-M,fl+M]
% imposing a permutation on the second part of the band [fl+1,fl+M]. For this, use the
% modified fastmooth function below.
Hy(:,:,iBlock) = fastsmooth_permuting(Ec(:,:,iBlock),BW);
end

% Crop firsts/lasts (BW-1)/2 = M points (which are zero). These blocks are neglected, since
% there are not enough points to compute the above average for them (see fastsmooth.m).
nFreqBins = nFreqBins - (BW-1);
firstFrequency = M + 1;
lastFrequency = firstFrequency + nFreqBins - 1;
Hy = Hy(firstFrequency:lastFrequency,:,:); % nBlocks x nFrequencyBins x J x J


Hy = permute(Hy,[2 3 1]);        % Put Hy in a more desirable format (I x nBlocks x nFreqBins).

end
```


Função para computação das dispersões $\sigma^2_{D_2}(f)$

Arquivo de função “compute_dispersion_D2":

```
function [dispD2,D2] = compute_dispersion_D2(Hy)
%% This function computes the differences D2(f,k) and its dispersion (sigma^2).
%
%  According to:
%   D.-T. Pham,and Ch. Serviere, "Permutation correction in the frequency domain in blind
%   separation of speech mixtures", in Eurasip Journal on Applied Signal Processing, Volume
%   2006.
%
%  Input:
%   Hy:          "New" averaged profiles with an imposed permutation after a given frequency fl.
%                Also an array of size I x nBlocks x nFrequencyBins.
%  Outputs:
%   D2:          D2(f,k) = Hy(1,k,f) - Hy(2,k,f). Difference between the new averaged profiles
%                with imposed permutation (nFreqBins x nBlocks).
%   dispD2:     Dispersion of D2 (vector of length nFrequencyBins).
%
% *This function was developed having in mind the case: number of sources I = 2.

%% Get key dimensions from the inputs and check their validity.
[I,nBlocks,nFreqBins] = size(Hy);

if I ~= 2
error('Number of sources (first dimension of Fy) must be equals 2.')
end

%% Compute differences D_2(f,k).
Hy = permute(Hy,[3 2 1]);   % Put Hy in a more suitable format (nFreqBins x nBlocks x I)
```

```
D2 = Hy(:,:,1) - Hy(:,:,2); % Compute differences D_2 (nFreqBins x nBlocks)

%% Compute dispersion of D2:
dispD2 = zeros(nFreqBins,1);   % Initialize vector.
for iFreq = 1:nFreqBins
dispD2(iFreq) = (D2(iFreq,:)*D2(iFreq,:).')/(nBlocks);
end

end % End of function.
```