`amiet_tools`: um pacote em Python para predição de ruído de interação turbulência-aerofólio
EM CONSTRUÇÃO

Fabio Casagrande Hirono
fchirono@gmail.com

Janeiro 2021

Resumo

O pacote `amiet_tools` (AmT), escrito em linguagem Python, é uma implementação do modelo analítico de Amiet [1] para predição de ruído de interação turbulência-aerofólio, com extensões. As funções permitem o cálculo do "salto" de pressão na superfície do aerofólio (i.e. a distribuição da fonte acústica) gerado em resposta à turbulência incidente, e do campo acústico radiado por esta interação. A turbulência incidente pode ser uma única rajada senoidal, ou uma soma de rajadas incoerentes com amplitudes definidas por um espectro energético. Outras funções includem o modelamento de efeitos de convecção e refração na propagação sonora para modelar medições acústicas realizadas em túneis de vento abertos ou fechados. Uma publicação de referência está disponível [2], e o pacote pode ser encontrado através do link https://github.com/fchirono/amiet_tools.

1 Introdução

1.1 Contexto

O ruído gerado pela interação entre um escoamento turbulento e uma superfície sólida, tal como um aerofólio, é uma das principais fontes de ruído banda larga na indústria de aviação moderna. Apesar dos mecanismos físicos fundamentais desta fonte de ruído estarem bem estabelecidos, métodos eficientes para redução de ruído de interação turbulência-aerofólio ainda são objeto de estudos e pesquisas. Devido ao seu baixo custo computacional, diversos modelos analíticos são frequentemente adotados para analisar os comportamentos físicos fundamentais e escalas do problema. Entre os modelos analíticos mais populares para predição de ruído de aerofólio, o modelo de Amiet [1] para um aerofólio do tipo placa plana em campo livre é frequentemente considerado um dos mais importantes devido a sua simplicidade formal e extensibilidade.

Em resumo, o modelo de Amiet decompõe um escoamento turbulento em uma soma de Fourier, onde cada componente constitui uma "rajada senoidal", e calcula um "salto de pressão" distribuído sobre a superfície do aerofólio em resposta à interação com cada rajada. Seguindo a analogia acústica de Curle, a radiação acústica pode ser calculada usando fontes equivalentes - no caso fontes tipo dipolo distribuídas sobre a superfície do aerofólio - com a amplitude dos dipolos dada pelo salto de pressão. O campo acústico radiado é então calculado através de uma integral de radiação, levando-se em conta os efeitos de convecção e/ou refração do som na propagação entre a fonte e o(s) observador(es).

Ao assumir que o aerofólio possui envergadura infinita, pode-se mostrar que apenas uma rajada senoidal será responsável por todo o ruído recebido por um observador distante; a expressão resultante desta simplificação relaciona a densidade espectral de potência (*power spectral density* - PSD) do ruído visto pelo observador diretamente às propriedades do escoamento turbulento e à geometria do aerofólio, evitando o cálculo da distribuição de fonte acústica sobre a superfície

do aerofólio e o cálculo da integral de radiação completa. Esta forma simplificada do modelo de Amiet é a mais popular, e frequentemente é suficiente para se obter tendências gerais do ruído radiado.

Porém, problemas mais avançados podem não permitir que tais aproximações sejam feitas, e exigir o cálculo explícito do campo acústico próximo ao aerofólio ou a inclusão de efeitos de envergadura finita. Sob estas condições, o modelo de Amiet requer o cálculo separado da distribuição de fontes e da integral de radiação completa, aumentando significativamente a complexidade do modelo de Amiet. Neste caso, é válido buscar uma implementação de referência do modelo de Amiet completo, permitindo que os pesquisadores possam rapidamente obter resultados confiáveis e não perder tempo com detalhes de implementação.

Este documento busca introduzir o pacote `amiet_tools` (abreviado por "AmT"), uma implementação de referência do modelo de Amiet completo para ruído de interação turbulência-aerofólio, e explicar o seu uso. O pacote AmT é desenvolvido em linguagem Python e disponibilizado com uma licença de código aberto, permitindo a sua distribuição, modificação e uso por diversos usuários em múltiplos sistemas operacionais. O pacote encontra-se hospedado na plataforma GitHub através do link:

- https://github.com/fchirono/amiet_tools .

1.2 O modelo de Amiet

Esta seção busca descrever sucintamente o modelo analítico de Amiet para o problema de ruído de interação turbulência-aerofólio. Para mais detalhes, consulte a próxima subseção para sugestões de referências teóricas.

As condições experimentais do problema estão indicadas na Figura 1: um aerofólio do tipo placa plana está imerso em um escoamento turbulento subsônico, de número de Mach $M_x = U_x/c_0$ e na direção $+x$. O aerofólio possui corda $c = 2b$ e envergadura $L = 2d$, e a origem do sistema de coordenadas é colocado no centro do aerofólio, com o eixo $+z$ apontando para cima ("fora da página" na Fig. 1). Pontos no espaço ao redor do aerofólio são indicados por $\mathbf{r} = (x, y, z)$, e pontos na superfície do aerofólio são indicados com um caractere subescrito "$_s$", $\mathbf{r}_s = (x_s, y_s, z_s)$. O aerofólio encontra-se no plano $z_s = 0$.

Um contínuo de rajadas turbulentas senoidais "congeladas" movem-se no plano $z = 0$ com velocidade de convecção U_x, e cada rajada é denotada por um par de números de onda hidrodinâmicos (isto é, vorticais) longitudinal (*chordwise* - na direção da corda) e transversal (*spanwise* - na direção da envergadura) (k_χ, k_ψ). Assumindo a dependência temporal da forma $e^{+j\omega t}$, um "salto" de pressão de superfície Δp irá se formar sobre a superfície do aerofólio em resposta à turbulência incidente, e será da forma

$$\Delta p(x_s, y_s, \omega) = 2\pi\rho_0 \int_{-\infty}^{+\infty} w(\kappa_\chi, k_\psi)\ g(x_s, \kappa_\chi, k_\psi) e^{-jk_\psi y_s}\, \mathrm{d}k_\psi, \tag{1}$$

onde $w(k_\chi, k_\psi)$ é a amplitude da rajada, $\kappa_\chi = \omega/U_x$ é o número de onda longitudinal da rajada na frequência $\omega = 2\pi f$, e g é o salto de pressão não-dimensional sobre a corda x_s devido a cada componente (κ_χ, k_ψ).

Porém, como turbulência é um fenômeno aleatório, esta deve ser analisada de forma estatística. A densidade espectral de potência cruzada (*cross-power spectral density* - CSD) do salto de pressão de superfície entre dois pontos (x_s, y_s) e (x'_s, y'_s) é dado por

$$S_{\Delta p \Delta p'}(x_s, x'_s, y_s, y'_s, \omega) = \lim_{T\to\infty} \left[\frac{\pi}{T} \mathrm{E}\left\{\Delta p(x_s, y_s, \omega) \Delta p^*(x'_s, y'_s, \omega)\right\}\right] \tag{2}$$

$$= (2\pi\rho_0)^2 U_x \int_{-\infty}^{+\infty} \Phi_{ww}(\kappa_\chi, k_\psi) g(x_s, \kappa_\chi, k_\psi) g^*(x'_s, \kappa_\chi, k_\psi) e^{-jk_\psi(y_s - y'_s)}\, \mathrm{d}k_\psi \tag{3}$$

onde $\Phi_{ww}(k_\chi, k_\psi)$ é a densidade espectral de número de onda da turbulência incidente.

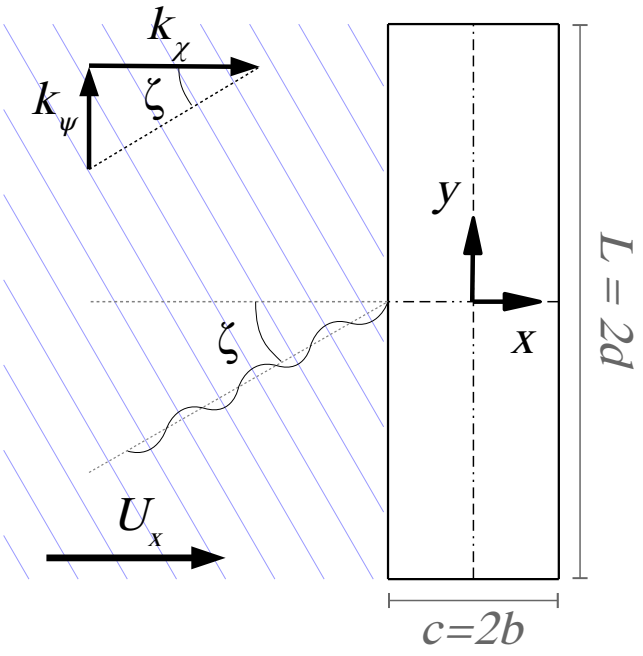

Figura 1: Experimental setup diagram, with gust incidence angle $\zeta = \arctan(k_\psi/k_\chi)$.

1.3 Referências Teóricas

- R. Amiet, "*Acoustic radiation from an airfoil in a turbulent stream*", Journal of Sound and Vibration, Vol. 41, No. 4:407–420, 1975 [1];
- G. Reboul, "*Modélisation du bruit à large bande de soufflante de turboréacteur*", PhD Thesis, Laboratoire de Mécanique des Fluides et d'Acoustique - École Centrale de Lyon, Lyon - France, 2010 [3];
- M. Roger, "*Broadband noise from lifting surfaces: Analytical modeling and experimental validation*", in: R. Camussi (Ed.), "*Noise Sources in Turbulent Shear Flows: Fundamentals and Applications*", Springer-Verlag, 2013 [4];
- L. de Santana, "*Semi-analytical methodologies for airfoil noise prediction*", PhD Thesis, Faculty of Engineering Sciences - Katholieke Universiteit Leuven, Leuven, Belgium, 2015 [5];
- F. Casagrande Hirono, "*Far-Field Microphone Array Techniques for Acoustic Characterisation of Aerofoils*", PhD Thesis, Institute of Sound and Vibration Research, University of Southampton, Southampton - UK, 2018 [6].

2 Usando o pacote

2.1 Requerimentos

O pacote `amiet_tools` (abreviado por "AmT") foi desenvolvido em Python 3.x, e tem como dependências os pacotes `numpy` e `scipy`. Para plotar resultados, esse tutorial usa o pacote `matplotlib`, mas esta não é uma dependência do AmT. Estes pacotes estão inclusos na "*Anaconda Python Distribution*"[1], uma distribuição Python gratuita e de código aberto. A distri-

[1]Disponível em https://www.anaconda.com/products/individual.

buição Anaconda é usada para desenvolver o pacote AmT, e é a distribuição recomendada para usar AmT.

2.2 Inserindo dados experimentais

O pacote AmT utiliza dois arquivos de texto contendo as variáveis relacionadas às condições do experimento e as variáveis relacionadas à geometria do aerofólio. Usaremos como exemplo os dois arquivos de configuração do teste "DARP2016", realizado no túnel de vento DARP da Universidade de Southampton, Reino Unido, e disponíveis no repositório GitHub do projeto.

O primeiro arquivo chama-se `DARP2016_TestSetup.txt`, e os seus conteúdos estão indicados na listagem 1 abaixo; note que linhas vazias são ignoradas, e o caractere "#" indica comentários. O programa lerá os valores em cada linha do arquivo, e os interpretará conforme indicado nas respectivas linhas.

```
# amiet_tools Test Setup file
#
# DARP2016 test setup, ISVR, Univ. of Southampton, UK

# Acoustic characteristics
340.     # c0          Speed of sound [m/s]
1.2      # rho0        Air density [kg/m**3]
20e-6    # p_ref       Ref acoustic pressure [Pa RMS]

# turbulent flow properties
60       # Ux                  flow velocity [m/s]
0.025    # turb_intensity      turbulence intensity = u_rms/U
0.007    # length_scale        turb length scale [m]

# shear layer height
-0.075   # z_sl           Shear layer height (aerofoil is at z=0) [m]
```

Listing 1: Arquivo `DARP2016_TestSetup.txt`

O segundo arquivo chama-se `DARP2016AirfoilGeom.txt`, e contém variáveis relacionadas à geometria do aerofólio. Seus conteúdos estão indicados na listagem 2 abaixo.

```
# amiet_tools AirfoilGeom file
#
# DARP2016 flat plate airfoil, ISVR, Univ. of Southampton, UK

0.075    # b          airfoil half chord [m]
0.225    # d          airfoil half span [m]
100      # Nx         number of chordwise points (non-uniform sampl)
101      # Ny         number of spanwise points (uniform sampl)
```

Listing 2: Arquivo `DARP2016_AirfoilGeom.txt`

Para carregar os valores descritos nos arquivos de texto, usa-se o programa descrito na listagem 3. Primeiramente, o pacote AmT é importado pelo script. A função `AmT.loadTestSetup` é usada para carregar as variáveis relacionadas às condições do experimento a partir do arquivo de texto fornecido como argumento, e cria um objeto - aqui chamado de `DARP2016Setup` - para armazenar as variáveis. Este objeto é uma instância da classe `TestSetup`, definida no pacote AmT; mais informações sobre as classes usadas no pacote AmT podem ser encontradas na Seção 5.

Similarmente, usa-se a função `AmT.loadAirfoilGeom` para carregar variáveis relacionadas à geometria do aerofólio a a partir de um arquivo de texto. Esta criará um objeto - aqui chamado de `DARP2016Airfoil` - para armazenar as dimensões do aerofólio e as configurações de amostragem de sua superfície. Este objeto é uma instância da classe `AirfoilGeom`, definida no pacote AmT; mais informações sobre as classes usadas no pacote AmT podem ser encontradas na Seção 5.

```
import amiet_tools as AmT

# load test setup from file
DARP2016Setup = AmT.loadTestSetup('../DARP2016_TestSetup.txt')

# load airfoil geometry from file
DARP2016Airfoil = AmT.loadAirfoilGeom('../DARP2016_AirfoilGeom.txt')
```

Listing 3: Importando pacotes e carregando variáveis

Alternativamente, estas funções podem ser chamadas sem nenhum argumento de entrada; o objeto resultante irá conter as configurações do teste "DARP2016" por padrão, e estes podem ser modificados posteriormente dentro do script. Um exemplo deste método alternativo encontra-se na listagem 4, onde três variáveis são modificadas e recebem novos valores. Note, porém, que é necessário executar os métodos internos `MyTestSetup._calc_secondary_vars()` e `MyAirfoil._calc_grid()` para recalcular as variáveis secundárias depois das variáveis primárias serem redefinidas.

```
import amiet_tools as AmT

# Inicializando os objetos para armazenar variaveis
MyTestSetup = AmT.loadTestSetup()
MyAirfoil = AmT.loadAirfoilGeom()

# Definir novos valores
MyTestSetup.Ux = 120
MyTestSetup.turb_intensity = 0.05
MyAirfoil.b = 0.15

# IMPORTANTE: recalcular as variaveis secundarias em ambos objetos
MyTestSetup._calc_secondary_vars()
MyAirfoil._calc_grid()
```

Listing 4: Método alternativo para carregar variáveis

Desta forma, as diversas variáveis relacionadas ao setup experimental e ao aerofólio utilizado podem ser importadas e combinadas; por exemplo, pode-se simular dois aerofólios diferentes dentro das mesmas condições experimentais. As variáveis podem ser acessadas como atributos dos objetos criados (por exemplo, `DARP2016Setup.c0` para acessar a velocidade do som), ou ainda podem ser exportadas diretamente para o "namespace" atual utilizando o método `export_values` (pertencente às duas classes), conforme indicado na listagem 5. Desta forma, as variáveis agora existem tanto como atributos dos objetos criados quanto como variáveis independentes dentro do namespace atual, facilitando certos cálculos.

```
# export variables to current namespace
(c0, rho0, p_ref, Ux, turb_intensity, length_scale, z_sl, Mach, beta,
flow_param, dipole_axis) = DARP2016Setup.export_values()

(b, d, Nx, Ny, XYZ_airfoil, dx, dy) = DARP2016Airfoil.export_values()
```

Listing 5: Exportando as variáveis

Finalmente, as variáveis relacionadas à frequência temporal f_0 (em Hz) são criadas dentro do próprio código, dependendo do interesse do usuário. Neste exemplo inicial, vamos definir uma única frequência f_0 a partir da frequência normalizada pela corda $k_0 c = 5$. Para tal, definimos o valor desejado de f_0 (a partir da frequência não-dimensional $k_0 c$), e passamos esta variável e o objeto `DARP2016TestSetup` como argumentos para a função `AmT.FrequencyVars`, resultando no objeto aqui chamado `FreqVars`. Este objeto é uma instância da classe `FrequencyVariables`, definida no pacote AmT; mais informações sobre as classes usadas no pacote AmT podem ser

encontradas na Seção 5. Similarmente, os valores aqui calculados podem ser exportados para o namespace atual através do método `FreqVars.export_values`.

```
# frequency of operation
kc = 5                              # chordwise normalised frequency = k0*(2*b)
f0 = kc*c0/(2*np.pi*(2*b))          # approx 1.8 kHz

FreqVars = AmT.FrequencyVars(f0, DARP2016Setup)
(k0, Kx, Ky_crit) = FreqVars.export_values()
```

Listing 6: Criando variáveis relacionadas à frequência

3 Exemplos

3.1 Exemplo 1: Interação de Rajada Única

Como primeiro exemplo, vamos usar o pacote AmT para calcular a distribuição do "salto" de pressão $\Delta p(x_s, y_s)$ na superfície do aerofólio em resposta a uma única rajada turbulenta, e as funções de diretividade do campo acústico nas direções da corda e da envergadura.

4 Pseudocódigo

O pseudocódigo usado para calcular o espectro cruzado $S_{\Delta p \Delta p'}$ da pressão de superfície do aerofólio e/ou o espectro cruzado $S_{pp'}$ do ruído radiado é mostrado abaixo, retirado de [2]. A maior parte das instruções mostradas abaixo possuem funções Python equivalentes ou métodos associados no pacote AmT, e não precisam ser implementadas pelo usuário final. Para mais detalhes sobre o uso destas funções, veja a documentação e os scripts de exemplo no pacote.

Algorithm 1 Pseudocódigo para o modelo de predição de ruído de interação turbulência-aerofólio do pacote `amiet_tools`

Definir as variáveis do ambiente de teste: $c_0, \rho_0, p_{ref}, U_x, \overline{w^2}, \Lambda, z_{sl}, M_x, \beta$
Definir as variáveis da geometria do aerofólio: $b, d, N_x, N_y, \Delta x_s, \Delta y_s$
Calcular a amostragem da corda do aerofólio $x_s[n],\ n \in [1, \ldots, N_x]$
Calcular a amostragem da envergadura do aerofólio $y_s[n],\ n \in [1, \ldots, N_y]$
Definir as coordenadas dos observadores $\mathbf{r}_m = (x_m, y_m, z_m),\ m \in [1, \ldots, M]$
if usando correção de camada cisalhante **then**
    for all pares (m, n) de pontos observador-aerofólio $(\mathbf{r}_m|\mathbf{r}_s)$ **do**
        Calcular os pontos de cruzamento da camada cisalhante $\mathbf{r}_l(m, n)$ e tempos de propagação $\tau_{sm}(m, n)$
    end for
end if
for cada frequência $f \in [f_{min}, \ldots, f_{max}]$ **do**
    Calcular $k_0, \kappa_\chi, k_\psi^{crit}$
    Calcular valores de números de onda de rajada $k_\psi \in [-k_\psi^{max}, k_\psi^{max}]$
    Calcular intervalo de amostragem dos números de onda de rajada Δk_ψ
    Calcular espectro de energia dos números de onda de rajada $\Phi_{ww}(k_\psi)$
    Calcular matriz $\partial G(\mathbf{r}_m|\mathbf{r}_s)/\partial z_s$ para todos os pares (m, n) de pontos observador-aerofólio;
    Inicializar $S_{\Delta p \Delta p'} \leftarrow 0$
    Inicializar $S_{pp'} \leftarrow 0$
    for cada rajada $k_\psi \in [-k_\psi^{max}, k_\psi^{max}]$ **do**
        $w(k_\psi) \leftarrow \sqrt{\Phi_{ww}(k_\psi)}$
        Calcular $\Delta p(x_s, y_s, k_\psi)$ usando $w(k_\psi)$
        $S_{\Delta p \Delta p'}(\mathbf{r}_s, \mathbf{r}'_s) \leftarrow S_{\Delta p \Delta p'}(\mathbf{r}_s, \mathbf{r}'_s) + [\Delta p(x_s, y_s, k_\psi) \cdot \Delta p^*(x'_s, y'_s, k_\psi)] \cdot U_x \cdot \Delta k_\psi$
    end for
    $S_{pp'}(\mathbf{r}_m, \mathbf{r}_{m'}) \leftarrow 4\pi \left[S_{\Delta p \Delta p'} \cdot (\Delta x_s \cdot \Delta y_s) \cdot (\Delta x'_s \cdot \Delta y'_s)\right] \cdot [\partial G(\mathbf{r}_m|\mathbf{r}_s)/\partial z_s] \cdot [\partial G(\mathbf{r}_{m'}|\mathbf{r}'_s)/\partial z_s]$
end for

5 Descrição das classes

O pacote `amiet_tools` utiliza classes - i.e. estruturas de abstração usadas em programação orientada a objetos - para armazenar dados relacionados aos diferentes aspectos da simulação. As três classes utilizadas em AmT estão descritas abaixo, seguidas de uma lista de seus atributos:

- Classe `TestSetup`:
  - `TestSetup.c0`: velocidade do som em um meio em repouso c_0 (em m/s);
  - `TestSetup.rho0`: densidade do ar ρ_0 (em kg/m^3);
  - `TestSetup.p_ref`: pressão acústica de referência $p_{ref} = 20\mu$Pa RMS;
  - `TestSetup.Ux`: velocidade média do escoamento U_x (em m/s);
  - `TestSetup.turb_intensity`: intensidade da turbulência $\overline{w^2}/U_x$;
  - `TestSetup.length_scale`: escala de comprimento de turbulência Λ (em m);
  - `TestSetup.z_sl`: altura da camada cisalhante z_{sl} (em m);
  - `TestSetup.flow_dir`: caractere único indicando a direção do escoamento (i.e. '`x`' para indicar escoamento na direção $+x$);
  - `TestSetup.dipole_axis`: caractere único indicando a direção do eixo dos dipolos (i.e. '`z`' para dipolos apontando "para cima", na direção $+z$);
  - `TestSetup.Mach`: número de Mach do escoamento $M_x = U_x/c_0$;
  - `TestSetup.beta`: fator de Prandtl-Glauert $\beta = \sqrt{1 - M_x^2}$.
- Classe `AirfoilGeom`:
  - `AirfoilGeom.b`: semi corda do aerofólio $b = c/2$ (em m);
  - `AirfoilGeom.d`: semi envergadura do aerofólio $d = L/2$ (em m);
  - `AirfoilGeom.Nx`: número de amostras sobre a corda N_x, amostragem não-uniforme;
  - `AirfoilGeom.Ny`: número de amostras sobre a envergadura N_y, amostragem uniforme;
  - `AirfoilGeom.dx`: intervalos de amostragem sobre a corda Δx_s (calculado internamente)
  - `AirfoilGeom.dy`: intervalo de amostragem sobre a envergadura Δy_s (calculado internamente).
- Classe `FrequencyVariables`:
  - `FrequencyVariables.freq`: frequência f (em Hz);
  - `FrequencyVariables.k0`: número de onda acústico $k_0 = 2\pi f/c_0$;
  - `FrequencyVariables.Kx`: número de onda de rajada na direção da corda $\kappa_\chi = \omega/U_x$;
  - `FrequencyVariables.Ky_crit`: número de onda crítico de rajada na direção da envergadura k_ψ^{crit}.

Uma instância de cada classe é criada para armazenar os valores das variáveis relacionados às condições de um experimento e para passá-las às diferentes funções no pacote. Valores numéricos podem ser atribuídos diretamente através de um script em Python, ou podem ser lidos em um arquivo externo através das funções `loadTestSetup` e `loadAirfoilGeom`.

Uma quarta classe chamada `MicArrayCsmHDF5` também está inclusa no pacote, e permite a leitura e escrita de dados de CSM de arranjos de microfones no formato HDF5 adotado pela comunidade "Array Methods" [7, 8] para compartilhar dados de "beamforming".

Referências

[1] R. K. Amiet. Acoustic radiation from an airfoil in a turbulent stream. *Journal of Sound and Vibration*, 41, No. 4:407–420, 1975.

[2] Fabio Casagrande Hirono, Phillip Joseph, and Filippo Fazi. An open-source implementation of analytical turbulence-airfoil interaction noise model. In *26th AIAA/CEAS Aeroacoustics Conference*, 2020. URL https://doi.org/10.2514/6.2020-2544. AIAA Paper 2020-2544.

[3] Gabriel Reboul. *Modélisation du bruit à large bande de soufflante de turboréacteur*. PhD Thesis, Laboratoire de Mécanique des Fluides et d'Acoustique - École Centrale de Lyon, Lyon - France, 2010.

[4] Michel Roger. Broadband noise from lifting surfaces: Analytical modeling and experimental validation. In Roberto Camussi, editor, *Noise Sources in Turbulent Shear Flows: Fundamentals and Applications*. Springer-Verlag Wien, 2013.

[5] Leandro de Santana. *Semi-analytical methodologies for airfoil noise prediction*. PhD Thesis, Faculty of Engineering Science - Katholieke Universiteit Leuven, Leuven - Belgium, 2015.

[6] Fabio Casagrande Hirono. *Far-Field Microphone Array Techniques for Acoustic Characterisation of Aerofoils*. PhD Thesis, Institute of Sound and Vibration Research - University of Southampton, Southampton - UK, 2018.

[7] Christopher Bahr, William Humphreys Jr., Daniel Ernst, Thomas Ahlefeldt, Carsten Spehr, Antonio Pereira, Quentin Leclere, Christophe Picard, Ric Porteous, Danielle Moreaux, Jeoffrey Fischer, and Con Doolan. A comparison of microphone phased array methods applied to the study of airframe noise in wind tunnel testing. In *23rd AIAA/CEAS Aeroacoustics Conference*, 2017. AIAA Paper AIAA 2017-3718.

[8] Ennes Sarradj, Gert Herold, Pieter Sijtsma, Roberto Merino-Martinez, Anwar Malgoezar, Mirjam Snellen, Thomas Geyer, Christopher Bahr, Ric Porteous, Danielle Moreau, and Con Doolan. A microphone array method benchmarking exercise using synthesized input data. In *23rd AIAA/CEAS Aeroacoustics Conference*, 2017. AIAA Paper AIAA 2017-3719.