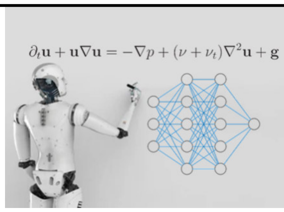


Hands on Machine Learning for Fluid Dynamics

7 – 11 February 2022

von KARMAN INSTITUTE FOR FLUID DYNAMICS

Lecture 8
Tutorial Exercise 2

Dominique Joachim

Joachim.dominique@vki.ac.be

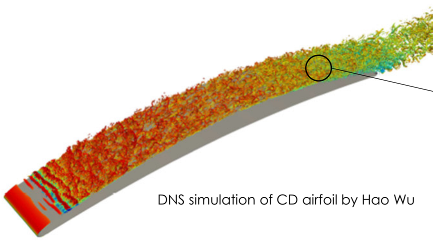
Table of Contents

Table of Contents

Wall pressure spectral model

Applications :

- Fatigue loading
- Vibro-acoustics
- **Aero-acoustics**



DNS simulation of CD airfoil by Hao Wu

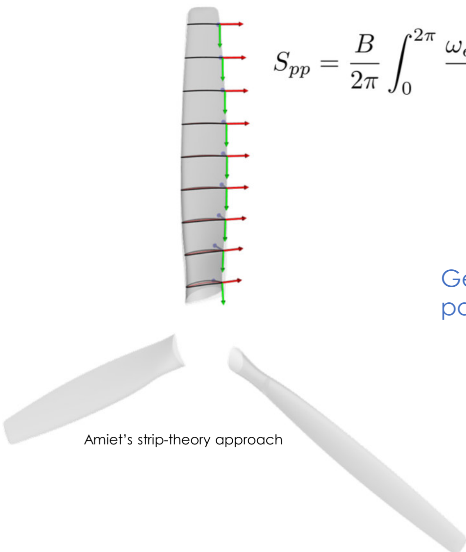
$$p'(\mathbf{x},t) = p(\mathbf{x},t) - \bar{p}(\mathbf{x},t)$$

Wall pressure fluctuations

$$\boxed{\Phi(\omega) = \frac{1}{2\pi}\int_0^\infty \overline{p'(\mathbf{x},t)p'(\mathbf{x},t+\tau)}e^{-i\omega t}d\tau}$$

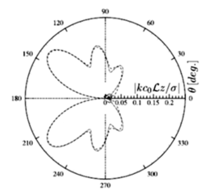
Single point wall pressure spectra

Application : Trailing edge noise



$$S_{pp} = \frac{B}{2\pi}\int_0^{2\pi}\frac{\omega_e(\Psi)}{\omega}\boxed{S_{pp}^{\Psi}}d\Psi$$

$$\boxed{S_{pp}^{\Psi}(\mathbf{x},\omega)} = \underbrace{\left(\frac{\omega c x_3}{2\pi c_o S_0^2}\right)^2 \frac{L}{2}}_{\text{Geometrical parameters}}\underbrace{\left|\mathcal{L}\left(\frac{\omega}{U_c},\frac{\bar{k}x_2}{S_0}\right)\right|^2}_{\text{Aeroacoustic transfer function}}\underbrace{\Phi_{pp}(\omega)l_y(\omega)}_{\text{Data obtained from RANS simulations}}$$

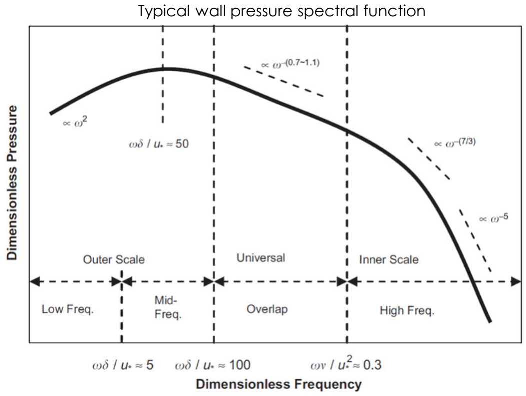


$\Phi_{pp}(\omega)$ Trailing edge wall-pressure spectrum (Goody, Kamruzzaman, Lee)

$$l_y(\omega) = \frac{bU_c}{\omega}$$ Spanwise correlation length (Corco's)

Problem definition

Typical wall pressure spectral function



General form of a semi-empirical model (Goody, Kamruzaman, Lee, …)

$$\Phi(\omega)\,SS = \frac{a\,(\omega FS)^b}{\left[i\,(\omega FS)^c + d\right]^e + \left[(f R_T^g)\,(\omega FS)\right]^h}$$

Parameters
a, b, c, d, e, f, g, h

$a = const.$
or
$a = f\,(\Pi, \beta_c, \Delta, \ldots)$

Scaling
***S**pectrum **S**caling, **F**requency **S**caling*

$\dfrac{\Phi(\omega)\,U_e}{\tau_w^2 \delta^*}$ $\dfrac{\omega\delta^*}{U_e}$

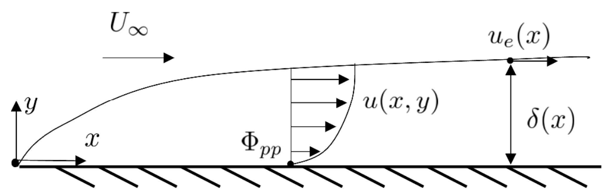
$\Phi(\omega)$ … wall pressure spectra
ω … frequency

Local approach !!!

$$\frac{\Phi(\omega) U_e}{\tau_w^2 \delta^*} = f(\Pi, \beta_c, \ldots)$$

Turbulent boundary layers

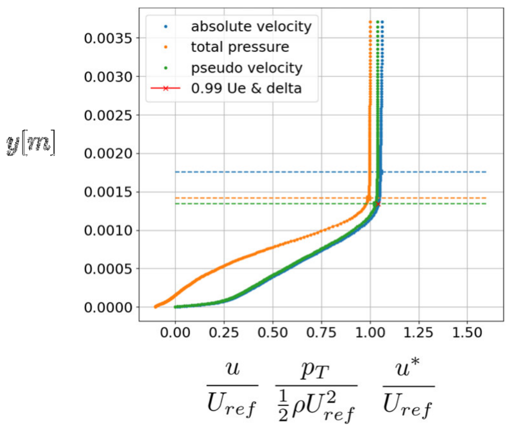
Let's consider a 2D turbulent boundary layer



$$\delta^*(x) = \int_0^\delta \left(1 - \frac{u(x,y)}{u_e}\right) dy\,,$$

$$\theta(x) = \int_0^\delta \frac{u(x,y)}{u_e}\left(1 - \frac{u(x,y)}{u_e}\right) dy\,.$$

Equilibrium velocity and boundary layer thickness



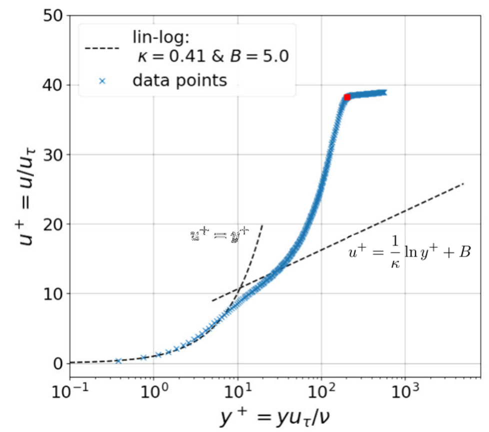
Total pressure

$$p_T = \frac{1}{2}\rho u^2$$

Pseudo-velocity

$$u^*(x,y) = -\int_0^y \Omega_z(x,\xi)d\xi$$

Wall shear stress



Definition : $\tau_w = \mu \left(\frac{\partial u}{\partial y}\right)_{y=0}$ $\quad u_\tau = \sqrt{\tau_w/\rho}$

Methods :

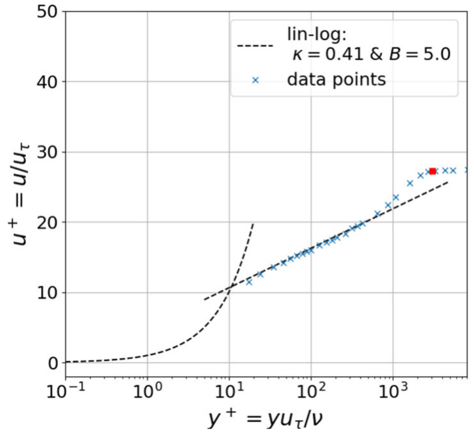
Finite difference $\quad \tau_w = \mu \frac{u_1 - u_0}{y_1 - y_0}$

Linear method $\quad u_\tau = \sqrt{\frac{u_0 \nu}{y_0}}$

Clauser's method $\quad u^+ = \frac{1}{\kappa}\ln y^+ + B$

Can you see a problem with that?

Wall shear stress



Definition : $\tau_w = \mu \left(\frac{\partial u}{\partial y}\right)_{y=0}$ $\quad u_\tau = \sqrt{\tau_w/\rho}$

Methods :

Finite difference $\tau_w = \mu \frac{u_1 - u_0}{y_1 - y_0}$

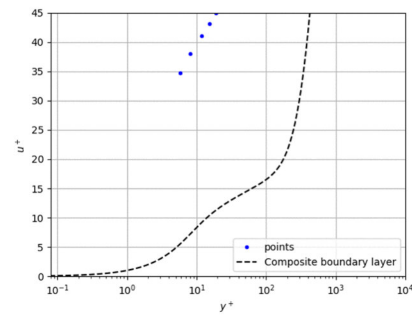
Linear method $u_\tau = \sqrt{\frac{u_0 \nu}{y_0}}$

Clauser's method $u^+ = \frac{1}{\kappa} \ln y^+ + B$

Can you see a problem with that?

Processing of turbulent boundary layers

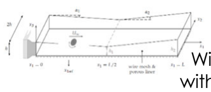
$$u^+(y^+) = \begin{cases} \overbrace{\boxed{u^+_{inner}(y^+)}}^{\text{Musker law of the wall}} + \overbrace{\boxed{\frac{2\Pi}{\kappa} W\left(\frac{y}{\delta}\right)}}^{\text{Chauhan Wake correction}}, & \text{if } y \leq \delta \\ \underbrace{\boxed{u_e/u_\tau}}_{\text{External velocity}} & \text{otherwise} \end{cases}$$



High pressure gradient !!!

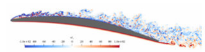
Modification of the law of the wall from Nickels (2004) and Nagib (2008)

Data



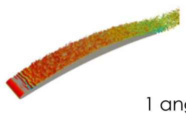
Salze et al. (2014)

EXPERIMENTAL
Wind tunnel, channel flow
with changing ceiling angle,
separate ZPG, APG, FPG data.



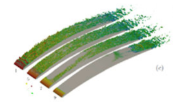
Deuse et al. (2020)

NUMERICAL
LES/DNS
CD airfoil,
1 angle of attack and 1 Re num.



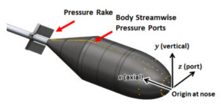
Hao Wu (2020)

NUMERICAL
DNS
CD airfoil,
1 angle of attack and 1 Re num.



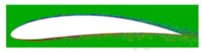
Christophe et al. (2009)

NUMERICAL
LES
CD airfoil,
several angle of attacks.



Balantrapu (2021)

EXPERIMENTAL
Wind tunnel,
axisymmetric tail.
PIV



Winkler et al. (2021)

NUMERICAL
DNS
NACA 6512-63,
1 angle of attack and 1 Re num.

Choice of the terminals

$$f\left(\delta, \delta^*, \theta, U_e, \nu, \rho, \tau_w, \frac{dp}{dx}, c_0, \Pi_{Coles}, \Phi_{pp}, \omega\right) = 0$$

Can you see a problem with that?

Parameters – Fundamental units = N. of non-dim params.

$$i = n - k = 11 - 3 = 8$$

$$\psi\left(\Pi_1, \Pi_2, \Pi_3, \Pi_4, \Pi_5, \Pi_6, \Pi_7, \Pi_8, r\right) = 0 \qquad r = \Pi_{Coles}$$

database	N	$\tilde{\omega}$	Δ	H	M	Π	C_f	R_T	β_C
Salze (2014)	10	$0.03 \to 33.6$	$0.09 \to 0.16$	$0.07 \to 0.12$	$0.07 \to 0.17$	$0.03 \to 0.75$	$0.002 \to 0.004$	$7.60 \to 22.5$	$-0.47 \to 0.57$
Deuse (2020)	13	$0.01 \to 24.2$	$0.15 \to 0.32$	$0.11 \to 0.15$	$0.21 \to 0.27$	$0.14 \to 2.26$	$0.001 \to 0.004$	$2.58 \to 3.54$	$0.36 \to 15.59$
Hao (2020)	16	$0.02 \to 34.4$	$0.13 \to 0.35$	$0.09 \to 0.16$	$0.27 \to 0.34$	$0 \to 2.2$	$0.001 \to 0.006$	$1.56 \to 2.25$	$-0.15 \to 5.93$
Christophe (2011)	78	$0.002 \to 4.65$	$0.11 \to 0.30$	$0.86 \to 0.16$	$0.05 \to 0.06$	$0.0 \to 1.43$	$0.002 \to 0.006$	$1.49 \to 3.48$	$-0.05 \to 7.94$
All	117	$0.01 \to 34.4$	$0.09 \to 0.35$	$0.07 \to 0.16$	$0.05 \to 0.34$	$0 \to 2.26$	$0.001 \to 0.006$	$1.49 \to 22.5$	$-0.47 \to 15.59$

Table of Contents

von KARMAN INSTITUTE FOR FLUID DYNAMICS

Load the code from previous Lecture 7

Modify the code for wall pressure spectra

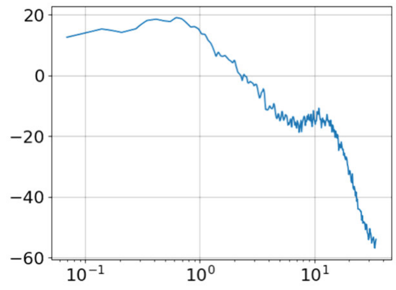
Exercise 1 : plot a single wall pressure spectra from the pandas dataframe

```
############################################################################
# 0 - Problem we are trying to solve (slide 9)
############################################################################

import pandas as pd

df = pd.read_pickle('dataframe')
```

Change this section

Solution

Exercise 1 : plot a single wall pressure spectra from the pandas dataframe




```
############################################################################
# 0 - Problem we are trying to solve (slide 9)
############################################################################
```

```
import pandas as pd

df = pd.read_pickle('dataframe')

name_plot = '2020_hao_case_CD_xc_0.02'
#get name
data = df.loc[df['name'] == name_plot]

plt.figure()
plt.semilogx(data['Pi1'],data['PiF_log'])
plt.grid()
plt.show()
```

Modify the code for wall pressure spectra

Exercise 2 : change the fitness function so that $10\log_{10}\left(\frac{\Phi_{pp}u_e}{\delta^*\tau_w^2}\right) = f(\Pi_1)$

```
##########################################################################
# 2 - Define what is the fitness (slide 11)
##########################################################################
```

Change this section

Solution

Exercise 2 : change the fitness function so that $10\log_{10}\left(\frac{\Phi_{pp}u_e}{\delta^*\tau_w^2}\right) = f(\Pi_1)$

```
###########################################################################
# 2 - Define what is the fitness (slide 11)
###########################################################################

def MSE(individual, points):
    # Transform the tree expression in a callable function
    func = toolbox.compile(expr=individual)

    # Evaluate the mean squared error between the expression
    sqerr = np.zeros(len(points))
    for ii, pt in enumerate(points):
        sqerr[ii] = (func(pt) - data['PiF_log'].iloc[ii])**2
        #print(sqerr[ii])
    return np.sum(sqerr)/len(points),

# we apply the fitness in an operation called evaluate where we compute the error for each points
toolbox.register("evaluate", MSE, points=data['Pi1'])
```

Modify the code for wall pressure spectra

Exercise 3 : change the display and try it out

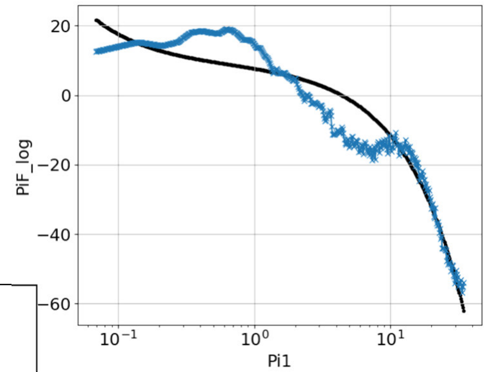
```
########################################################################
# 6 - Display end solution
########################################################################
```

Change this section

Solution

Exercise 3 : change the display and try it out

```
#########################################################################
# 6 - Display end solution
#########################################################################

bestfunct = toolbox.compile(expr=hof[0])
plotpoints = data['Pi1']

plt.figure()
plt.semilogx(data['Pi1'],data['PiF_log'],marker='x',linestyle='-')
for pt in plotpoints:
    plt.scatter(pt,bestfunct(pt),marker='.',color='k')
plt.grid()
plt.xlabel('Pi1')
plt.ylabel('PiF_log')
plt.show()

print(hof[0])
```



Modify the code for wall pressure spectra

Exercise 4 : add more variables

```
#########################################################################
# 2 - Define what is the fitness (slide 11)
#########################################################################
```

Change this section

```
#########################################################################
# 6 - Display end solution
#########################################################################
```

Change this section

Solution

Exercise 4 : add more variables

```
########################################################################
# 2 - Define what is the fitness (slide 11)
########################################################################
def MSE(individual, points):
    # Transform the tree expression in a callable function
    func = toolbox.compile(expr=individual)

    # Evaluate the mean squared error between the expression
    sqerr = np.zeros(len(points))
    for ii in range(len(points)):
        sqerr[ii] = (func(points.iloc[ii,0],points.iloc[ii,1]) - data['PiF_log'].iloc[ii])**2
        #print(sqerr[ii])
    return np.sum(sqerr)/len(points),

 # we apply the fitness in an operation called evaluate where we compute the error for each points
 toolbox.register("evaluate", MSE, points=data[['Pi1','Pi4']])
# 6 - Display end solution
########################################################################


bestfunct = toolbox.compile(expr=hof[0])
plotpoints = data[['Pi1','Pi4']]

plt.figure()
plt.semilogx(data['Pi1'],data['PiF_log'],marker='x',linestyle='-')
for ii in range(len(plotpoints)):
    plt.scatter(plotpoints.iloc[ii,0],bestfunct(plotpoints.iloc[ii,0],plotpoints.iloc[ii,1]),marker='.',color='k')
plt.grid()
plt.xlabel('Pi1')
plt.ylabel('PiF_log')
plt.show()

print(hof[0])
```

Add more data

Exercise 5 : add more data in the training

```
########################################################################
# 0 - Problem we are trying to solve (slide 9)
########################################################################
```

Change this section

Let’s code

Exercise 6 (optional) : Try to get the best fit

Let's code

Exercise 6 (optional) : Try to get the best fit

Let's discuss

How good was your fit ?

What problem did you face ?

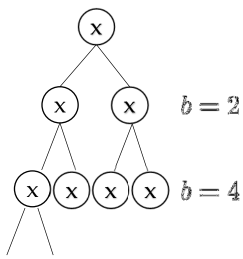
What could be improved ?

Choice of the operators

$$\Phi(\omega)\,SS = \frac{a\,(\omega FS)^{b}}{\left[i\,(\omega FS)^{c} + d\right]^{e} + \left[\left(fR_T^{g}\right)(\omega FS)\right]^{h}}$$

Exponent problem

$\tilde{\omega}^{b}$



...

New terminal type

The pow terminal

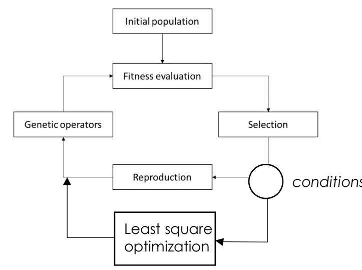
$\tilde{\omega}^{cst}$

R_T^{cst}

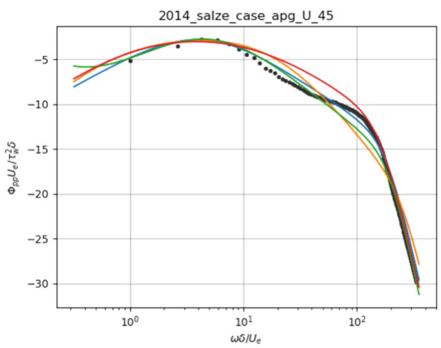
...

GAs have difficulties to find accurate numerical constant

Local optimizer



Results



N	Equation	fitness	trends
Solution 1	$\dfrac{4.65\frac{1+\beta_C}{1+C_f}\tilde{\omega}^{0.72}}{1.38(\tilde{\omega}^{1.58}+0.25)+\frac{5.78\tilde{\omega}^{6.5}}{\Delta R_T^{5.78}}}$	4,78 %	$\dfrac{4.65\boxed{\frac{1+\beta_C}{1+C_f}}\boxed{\tilde{\omega}^{0.72}}}{1.38(\boxed{\tilde{\omega}^{1.58}}+0.25)+\boxed{\frac{5.78\tilde{\omega}^{6.5}}{\Delta R_T^{5.78}}}}$
Solution 2	$\dfrac{\left(\frac{1.26}{R_T^{1.26}}+\Delta(1+(1+\beta_C)^{2.83})\right)\tilde{\omega}^{1.22}}{C_f+H\tilde{\omega}^{1.22}+0.1C_f\tilde{\omega}^{2.78}+\frac{\tilde{\omega}^{7.25}}{R_T^{6.88}}}$	4,94 %	$\dfrac{\left(\frac{1.26}{R_T^{1.26}}+\Delta(1+\boxed{(1+\beta_C)^{2.83}})\right)\boxed{\tilde{\omega}^{1.22}}}{C_f+H\tilde{\omega}^{1.22}+0.1C_f\boxed{\tilde{\omega}^{2.78}}+\boxed{\frac{\tilde{\omega}^{7.25}}{R_T^{6.88}}}}$
Solution 3	$\dfrac{(\Pi_C+C_f)(1+\beta_C)^{4.48}+\beta_C+1.57}{\frac{M}{R_T}+\tilde{\omega}+\frac{0.29\tilde{\omega}}{Cf+\tilde{\omega}^{1.76}}+\frac{\tilde{\omega}^{6.15}}{R_T^5}}$	3,73 %	$\dfrac{(\Pi_C+C_f)\boxed{(1+\beta_C)^{4.48}}+\beta_C+2.57}{\frac{M}{R_T}+\boxed{\tilde{\omega}}+\frac{0.29\tilde{\omega}}{Cf+\boxed{\tilde{\omega}^{1.76}}}+\boxed{\frac{\tilde{\omega}^{6.15}}{R_T^5}}}$
Solution 4	$\dfrac{\left(\Pi_C+R_T^{0.27}+2.76(1+\beta_C)^{2.76}\right)\tilde{\omega}}{R_T+(1+\beta_C)\tilde{\omega}+\tilde{\omega}^2+\frac{\tilde{\omega}^{6.6}}{(1+\beta_C)^{0.6}R_T^{4.06}}}$	5,43 %	$\dfrac{\left(\Pi_C+R_T^{0.27}+2.76\boxed{(1+\beta_C)^{2.76}}\right)\boxed{\tilde{\omega}}}{R_T+(1+\beta_C)\tilde{\omega}+\boxed{\tilde{\omega}^2}+\frac{\boxed{\tilde{\omega}^{6.6}}}{\boxed{(1+\beta_C)^{0.6}}\boxed{R_T^{4.06}}}}$

References

Dominique, J., van den Berghe J., Schram, C., & Mendez, M, A. (2022). **Artificial Neural Networks Modelling of Wall Pressure Spectra Beneath Turbulent Boundary Layers**. *Physics of Fluids (to be published)*.

Dominique, J., Christophe, J., Schram, C., & Sandberg, R. D. (2021). **Inferring empirical wall pressure spectral models with Gene Expression Programming**. *Journal of Sound and Vibration*, *506*, 116162.

Weatheritt, J., & Sandberg, R. (2016). **A novel evolutionary algorithm applied to algebraic modifications of the RANS stress–strain relationship**. *Journal of Computational Physics*, *325*, 22-37.

https://github.com/DominiqueVKI/VKI_researchWPS.

Take Home Messages

Genetic programing is particularly interesting because…

- ☑ Extremely simple coding from existing packages...
- ☒ … but often require modification to be efficient

- ☑ It gives you an equation
- ☒ … but you might be looking for only one good solution

- ☑ It gives works well with limited amount of data
- ☒ … but it does not scale well with large dataset

The end!

Slide 31