

MAE 298 Aeroacoustics – Homework#4

Turbomachinery Noise

Logan D. Halstrom

Graduate Student

Department of Mechanical and Aerospace Engineering

University of California, Davis, CA 95616

Problem Statement

You are designing a new aircraft engine and analyzing acoustic propagation generated by a non-uniform flow with angle of attack interacting with rotating fans. You obtained flow fields from CFD for a one-sixth small scale of the engine. The radius for the small scale engine is 13 in and the hub radius is 3 in. CFD provides the velocity gust information as a function of its circumferential modes. The circumferential mode for acoustics can be expressed as $m = nBkV$ where B is the number of blades, V is the number of vanes, n stands for the harmonic of BPF and k is the integer (1, 2, 3...). You consider only positive k at this time (this is related the rotation direction of gust). The number of blades is 18. The number of vanes is considered to be 1 since there is no physical vanes but there is one revolution difference. The Mach number is 0.525 and the fan RPM is 8326.3042, the speed of sound is 13503.937009 in/s and the density is 1.4988E-5 slug/in³. The dominant noise is generated at the 1st BPF or $n = 1$ in which the angular frequency ω is given as $RPM \times \frac{2\pi}{60} \times B$. We are interested in the propagation through the inlet of the engine so that sound propagates to z direction assuming the $+z$ direction is in the flow direction.

I. Eigenvalues

1. [20 points] Determine the first five eigenvalues of acoustics for $m=18, 17, 16, 15$ or ($k=1, 2, 3, 4$) or $(m,n)=(18,0), (18,1), (18,2), (18,3), (18,4), (17,0), (17,1), (17,2), (17,3), (17,4), (16,0), (16,1), (16,2), (16,3), (16,4), (15,0), (15,1), (15,2), (15,3), (15,4)$

	m = 18	m = 17	m = 16	m = 15
n = 0	1.5495	1.4696	1.3895	1.3093
n = 1	1.9612	1.8755	1.7896	1.7032
n = 2	2.2823	2.1932	2.1036	2.0137
n = 3	2.5772	2.4855	2.3932	2.3005
n = 4	2.8585	2.7646	2.6702	2.5753

Table 1: Eigenvalues μ for all modes

II. Eigenfunctions

2. [20 points] Plot the five eigenfunctions (radial modes, $n=0, 1, 2, 3, 4$) for $m=18, 17, 16, 15$ or ($k=1, 2, 3, 4$) and verify n describes the number of zero crossings in the radial direction

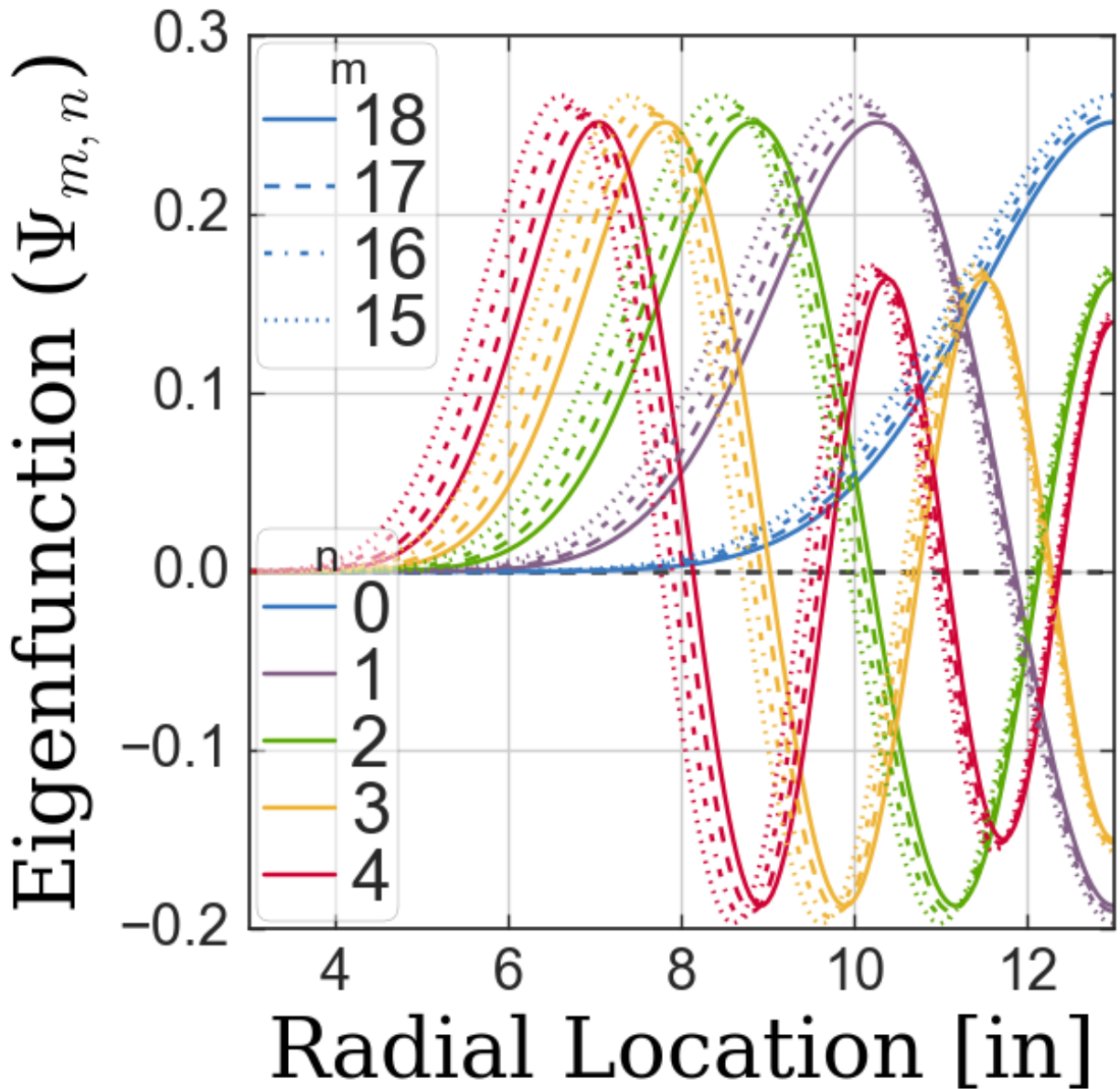


Figure 1: Eigenfunctions of all modes showing zero crossings corresponding to n

III. Wavenumbers

3. [20 points] Determine the wavenumbers in the z direction for $(m,n)=(18,0), (18,1), (18,2), (17,0), (17,1), (17,2), (16,0), (16,1), (16,2), (15,0), (15,1), (15,2)$. Indicate whether the mode is cut-on (propagating) or cut-off (exponentially decaying). Consider only the propagation in the \hat{z} direction

direction. Exclude the exponentially growing solution and include only the propagating solutions or exponentially decaying solutions.

m	n	K_z	Cut – on
18	0	(-0.842342752252+0.860467931211j)	No
18	1	(-0.842342752252+1.6539375966j)	No
18	2	(-0.842342752252+2.14865069168j)	No
17	0	(-0.842342752252+0.63804009925j)	No
17	1	(-0.842342752252+1.5105548176j)	No
17	2	(-0.842342752252+2.01642513534j)	No
16	0	(-0.842342752252+0.301624659942j)	No
16	1	(-0.842342752252+1.35896420335j)	No
16	2	(-0.842342752252+1.8801224576j)	No
15	0	(-0.386366560427+0j)	Yes
15	1	(-0.842342752252+1.19608312922j)	No
15	2	(-0.842342752252+1.73881279925j)	No

Table 2: Axial wavenumbers K_z for certain modes and associated cut-on condition

IV. Modal Power Levels

4. [30 points] The pressure distribution file at $z=0$ plane for $m=18$ or 1 BPF is provided. The first column is the dimensional radius [in], the second column the real part of the pressure [psi], and the third column is the imaginary part of the pressure [psi]. Using this boundary condition, compute the sound power level for $(m,n)=(18,0)$, $(18,1)$, $(18,2)$. This noise is considered for blade self noise that is not associated with the gust response since $k=0$. Note that the $z=0$ plane is not the same as the engine inlet. Use the conversion for the unit for the sound power as follows: $PWL(dB) = 10 * \log_{10}(W_{mn}) - 10 * \log_{10}(7.3756E - 13)$

n	PWL(dB)
0	-10.4015
1	-13.3776
2	-22.3880

Table 3: Sound Power Levels in dB of modal power of $m = 18$ circumferential mode

Appendix A: Global Variable Storage Script

```

0  """GLOBAL VARIABLES STORAGE
1  Logan Halstrom
2  MAE 298 AEROACOUSTICS
3  HOMEWORK 4 - TURBOMACHINERY NOISE
4  CREATED: 17 NOV 2016
5  MODIFIY: 06 DEC 2016
6
7  DESCRIPTION: Provide global variables for all scripts including wrapper.
8  """
9
10 import numpy as np
11
12 #INPUTS
13
14 #CFD Engine Model Parameters
15 scale = 1/6 #CFD engine model scale
16 Nblade = 18 #Number of blades in engine fan
17 Nvane = 1 #Uneven gust loading modeled as single vane
18 Ro = 13 #Model engine (outer) radius [in]
19 Ri = 3 #Model hub (inner) radius [in]
20 Linlet = 4 #Distance between inlet and z=0 plane [in]
21
22 #Flow Parameters
23 M = 0.525 #Engine flow mach number
24 a = 13503.937009 #Speed of Sound [in/s]
25 rho = 1.4988e-5 #density [slug/in^3]
26 RPM = 8326.3042 #Fan RPM
27 omega = RPM * (2 * np.pi / 60) * Nblade #angular frequency [rad/s]
28
29
30 #DATA OVERWRITE SWITCHES
31 datoverwrite = 1 #Overwrite data = 1
32
33 #LOAD/SAVE DIRECTORIES
34 datadir = 'Data' #Source and processed data storage directory
35 savedir = 'Results' #
36
37 picdir = 'Plots' #Plot storage directory
38 pictype = 'png' #Plot save filetype
39 pictype = 'pdf' #Plot save filetype
40
41 sigfigs = 4 #number of sig figs to save in data files
42
43
44 #CONVERSIONS
45 volt2pasc = -116.0 #volts to pascals

```

Appendix B: Main Data Processing and Plotting Script

```
0 """DATA PROCESSING PROGRAM
1 Logan Halstrom
2 MAE 298 AEROACOUSTICS
3 HOMEWORK 4 - TURBOMACHINERY NOISE
4 CREATED: 17 NOV 2016
5 MODIFIY: 06 DEC 2016
6
7 DESCRIPTION:
8
9 NOTE:
10 """
11
12 #IMPORT GLOBAL VARIABLES
13 from hw4_98_globalVars import *
14
15 import numpy as np
16 import pandas as pd
17
18 from scipy.special import jn, yn #bessel functions
19 from scipy.special import jv, yv #bessel functions
20 from scipy.optimize import fsolve #linear solver
21 from scipy.optimize import broyden1 #non-linear solver
22
23
24
25 #CUSTOM PLOTTING PACKAGE
26 import matplotlib.pyplot as plt
27 import sys
28 sys.path.append('/Users/Logan/lib/python')
29 from lutil import df2tex
30 from lplot import *
31 from seaborn import color_palette
32 import seaborn as sns
33 UseSeaborn('xkcd') #use seaborn plotting features with custom colors
34 colors = sns.color_palette() #color cycle
35 markers = bigmarkers #marker cycle
36
37
38 def AxialEigenvalFunc(x, m, ri, ro):
39     """Determinant of axial flow boundary condition eigenfunctions, which is
40     equal to zero. Use scipy solver to get eigenvalues
41     x --> dependent variable (eigenvalue mu in axial flow eigenfunction)
42     m --> Order of bessel function
43     ri --> inner radius of jet engine
44     ro --> ourter radius of jet engine
45     """
```

```

46
47 X = x * ro
48 Y = x * ri
49 return ( 0.5 * (jv(m-1, X) - jv(m+1, X)) * 0.5 * (yv(m-1, Y) - yv(m+1, Y))
50         - 0.5 * (jv(m-1, Y) - jv(m+1, Y)) * 0.5 * (yv(m-1, X) - yv(m+1, X))
51         )
52
53 def AxialEigenfunction(r, ri, m, mu):
54     """Axial flow eigenfunction
55     r --> radial location to evaluate eigenfunction at
56     ri --> inner radius of axial jet engine
57     m --> circumfrential acoustic mode
58     mu --> eigenvalue (dependent on radial mode n)
59     """
60     X = mu * ri
61     return ( jv(m, mu * r) - (0.5 * (jv(m-1, X) - jv(m+1, X)))
62            / (0.5 * (yv(m-1, X) - yv(m+1, X))) * yv(m, mu * r) )
63
64
65 def AxialWavenumber(mu, omega, c, M):
66     """Calculate z-direction wavenumber (Kz+) for sound mode in axial flow
67     mu --> eigenvalue of current mode
68     omega --> angular frequency of flow
69     c --> speed of sound
70     M --> mach number of flow
71     """
72     K = omega / c #flow wave number
73     Kz = (-M + np.emath.sqrt(1 - (1 - M ** 2) * (mu / K) ** 2 )) / (1 - M ** 2) * K
74     return Kz
75
76 def CutOnCondition(Kz):
77     """Determine if mode is cuton based on wave number
78     Must be real to propagate (pos. imag.: decay exponential, neg.: expand)
79     Must be negative for upstream propagation
80     Kz --> axial wavenumber for current mode
81     """
82     cuton = 'No'
83     #Test if no imaginary part (propagation)
84     if Kz.imag == 0:
85         #Test if upstream propagation
86         if Kz.real < 0:
87             cuton = 'Yes'
88     return cuton
89
90
91 def GetGammaAmn(m, n, mu, p, r):
92     """Get Gamma_mn and Amn from eigenfunction and acoustic pressure
93     m --> current circumfrential mode

```



```

104 n    --> current radial mode
105 mu   --> eigenvalue for current m,n
106 p    --> acoustic pressure at z=0 plane
107 r    --> radius vector associated with p (goes from Ri --> Ro)
108 """
109 ri, ro = min(r), max(r) #inner/outer radii of engine
110
111 #Calculate Gamma_mn for non m=n=0 case:
112 Gam = (0.5 * (ro ** 2 - m ** 2 / mu ** 2)
113         * AxialEigenfunction(ro, ri, m, mu) ** 2
114       - 0.5 * (ri ** 2 - m ** 2 / mu ** 2)
115         * AxialEigenfunction(ri, ri, m, mu) ** 2 )
116 #Calculate Amn
117 # Psi = lambda r: AxialEigenfunction(r, ri, m, mu)
118 Psi = AxialEigenfunction(r, ri, m, mu)
119 Amn = 1 / Gam * np.trapz( p * Psi * r, r)
120
121 return Gam, Amn
122
123 def main():
124     """Perform calculations for frequency data processing
125     """
126
127     #####
128     ### PROB 1 - FIND EIGENVALUES #####
129     #####
130
131     eigenvals = pd.DataFrame() #solutions for eigenvalues
132
133     ms = [18, 17, 16, 15] #circumferential modes to solve for
134     Nn = 5 #number of radial modes to solve for
135     ns = range(Nn) #find first five radial modes
136
137     # #tweak for round off error
138     #     #values for calculating Wmn are very near zero.  small round-off
139     #     #errors can cause Wmn=0 and PWL=infinity.
140     #     #Change number of sigfigs to get correct Wmn for all cases
141     rounds = [] #round eigenvalue to deal with floating point error
142     for m in ms:
143         rounds.append([None, None, None, None, None])
144     rounds[0] = [8, None, None, None, None]
145
146     for i, m in enumerate(ms):
147
148         #guesses for root solutions
149         x0s = np.linspace(0.1, 3, 300)

```

```

142     mus = []
143     for j, x0 in enumerate(x0s):
144
145         #try each solution guess to see if it returns a good result
146         try:
147             #find solution corresponding to current guess
148             mu = broyden1( lambda x: AxialEigenvalFunc(x, m, Ri, Ro), x0)
149             mu = float(mu)
150             #add to solution list if no error message
151             mus.append( mu )
152         except Exception:
153             #Skip any solution errors
154             pass
155
156     #GET ONLY UNIQUE VALUES OF MU
157     df = pd.DataFrame({'long' : list(mus), 'short' : list(mus)})
158     df = df.round({'short': 6}) #shorten values to find unique ones
159     df = df.drop_duplicates(subset='short') #drop duplicates in short vals
160     df = df.sort_values('long') #sort eigen values from least to greatest
161     df = df.reset_index() #reset indicies to same as n
162
163     #Assign desired number of eigen values to solution dataframe
164     mus = df['long'][:Nn]
165     #Deal with floating point error
166     for j, mu in enumerate(mus):
167         if rounds[i][j] != None:
168             mus[j] = round(mu, rounds[i][j])
169     eigenvals[m] = mus
170     # eigenvals = eigenvals.round({m : 8})
171
172     #SAVE EIGENVALUES
173     #columns are m, rows are n
174     eigenvals.to_csv( '{}/eigenvalues.dat'.format(datadir), sep=' ',
175                     index=True )
176     df2tex(eigenvals, '{}/eigenvalues'.format(datadir), dec=sigfigs)
177
178
179
180     #####
181     ### PROB 2 - PLOT EIGENFUNCTIONS #####
182     #####
183
184     R = np.linspace(Ri, Ro, 201) #Radial vector in engine
185     line = ['-','--','-.',':']
186
187     fig, ax = plt.subplots(len(ms), sharex=True, figsize=[3, 12])
188     for i, m in enumerate(ms):
189         for j, n in enumerate(ns):

```

```

190         ax[i].plot(R, AxialEigenfunction(R, Ri, m, eigenvals[m][j]),
191                    label='n={}'.format(j)
192                    )
193         ax[i].set_ylabel('$\\Psi_{{{}},n}}$'.format(m))
194         ax[i].set_xlim([Ri, Ro])
195     ax[i].set_xlabel('Radial Location [in]') #label x-axis on last subplot
196
197     savename = '{} / 2_eigenfunctions_subplot.{}'.format(picdir, pictype)
198     SavePlot(savename)
199
200
201
202     #####
203
204     _, ax = PlotStart(None, 'Radial Location [in]',
205                        'Eigenfunction ($\\Psi_{{m,n}}$)', figsize=[6, 6])
206
207     ax.plot([Ri, Ro], [0, 0], color='black', linestyle='--', linewidth=2, alpha=0.7 #zero
208     mhandles = []
209     mlabels = []
210     nhandles = []
211     nlabels = []
212     labels = []
213     for i, m in enumerate(ms):
214         for j, n in enumerate(ns):
215             h, = ax.plot(R, AxialEigenfunction(R, Ri, m, eigenvals[m][j]),
216                         label='n={}'.format(j), color=colors[j],
217                         linestyle=line[i],
218                         )
219             if j == 0:
220                 mhandles.append(h)
221                 mlabels.append(str(m))
222             if i == 0:
223                 nhandles.append(h)
224                 nlabels.append(str(n))
225             labels.append(j)
226     ax.set_xlim([Ri, Ro])
227
228     leg1 = PlotLegendLabels(ax, mhandles, mlabels, loc='upper left', title='m')
229     # leg1 = ax.legend(mhandles, mlabels, loc='upper left', title='m',
230     #                  # fancybox=True, frameon=True, framealpha=0.5,
231     #                  # numpoints=1, scatterpoints=1, prop={'size':28},
232     #                  # borderpad=0.1, borderaxespad=0.1, handletextpad=0.2,
233     #                  # handlelength=1.0, labelspace=0
234     #                  )
235
236     leg2 = PlotLegendLabels(ax, nhandles, nlabels, loc='lower left', title='n')
237     plt.gca().add_artist(leg1)

```

```

238
239 savename = '{}/2_eigenfunctions.{}'.format(picdir, pictype)
240 SavePlot(savename)
241
242
243 #####
244 ### PROB 3 - WAVE NUMBER/CUT-OFF CONDITION #####
245 #####
246
247
248 wavenums = eigenvals.copy()
249
250 # wavenums = AxialWavenumber(eigenvals, omega, a, M)
251 wavenums = wavenums.applymap(lambda x: AxialWavenumber(x, omega, a, M))
252
253 #SAVE EIGENVALUES
254 #columns are m, rows are n
255 wavenums.to_csv( '{}/wavenumbers.dat'.format(datadir), sep=' ',
256                 index=True )
257
258
259 #DETERMINE CUTOFF CONDITION
260 curNs = [0, 1, 2]
261 #sort by m
262 for m in ms:
263     Kzs = []
264     for n in curNs:
265         Kzs.append(wavenums[m][n])
266
267
268 df = pd.DataFrame({'m' : [m for x in curNs], 'n' : curNs, 'Kz' : Kzs})
269 df['Cut-on'] = [CutOnCondition(Kz) for Kz in df['Kz']]
270 #save wavenumber and cuton table
271 df = df[['m', 'n', 'Kz', 'Cut-on']]
272 df2tex(df, '{}/wavenumbers_m{}'.format(datadir, m), dec=sigfigs)
273
274
275 #####
276 ### PROB 4 - SPL OF REAL PRESSURE FIELD #####
277 #####
278
279 #READ EXPERIMENTAL PRESSURE DISTRIBUTION AT Z=0 PLANE
280 #Columns: Radius [in], Real pressure [psi], Imaginary pressure [psi]
281 df = pd.read_csv('{} /pressure_input.dat'.format(datadir), sep='\t',
282                 names=['R', 'pRe', 'pIm'])
283 #Create complex values of acoustic pressure
284 df['p'] = df['pRe'] + df['pIm'] * 1j
285

```

```

286
287 m, n = 18, 0
288
289 PWLs = []
290
291 for i, n in enumerate([0, 1, 2]):
292     mu = eigenvals[m][n] #eigenvalue for curent (m,n)
293     Gam, Amn = GetGammaAmn(m, n, mu, df['p'], df['R'])
294
295     # print(Gam)
296     # print(eigenvals[m][n])
297     # print(Amn)
298
299     #MODAL POWER
300     Kz = wavenums[m][n]
301     frac = Kz / (omega / a - Kz * M)
302     Wmn = np.pi / (rho * a) * Gam * Amn * np.conj(Amn) * (
303         (1 + M ** 2) * frac.real + M * (abs(frac) ** 2 + 1) )
304
305     #SOUND POWER LEVEL
306     PWL = 10 * np.log10( abs(Wmn) ) - 10* np.log10(7.3756e-13)
307
308     PWLs.append(PWL)
309
310     #SAVE POWERLEVELS
311     PWLs = pd.DataFrame({m : PWLs})
312     PWLs.to_csv( '{} /powerlevels.dat'.format(datadir), sep=' ', index=True )
313     df2tex(PWLs, '{} /powerlevels'.format(datadir), dec=sigfigs)
314
315
316
317 if __name__ == "__main__":
318
319
320     main()

```