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/in$\hat{3}$. 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)

	$\mathbf{m = 18}$	$\mathbf{m = 17}$	$\mathbf{m = 16}$	$\mathbf{m = 15}$
$\mathbf{n = 0}$	1.5495	1.4696	1.3895	1.3093
$\mathbf{n = 1}$	1.9612	1.8755	1.7896	1.7032
$\mathbf{n = 2}$	2.2823	2.1932	2.1036	2.0137
$\mathbf{n = 3}$	2.5772	2.4855	2.3932	2.3005
$\mathbf{n = 4}$	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

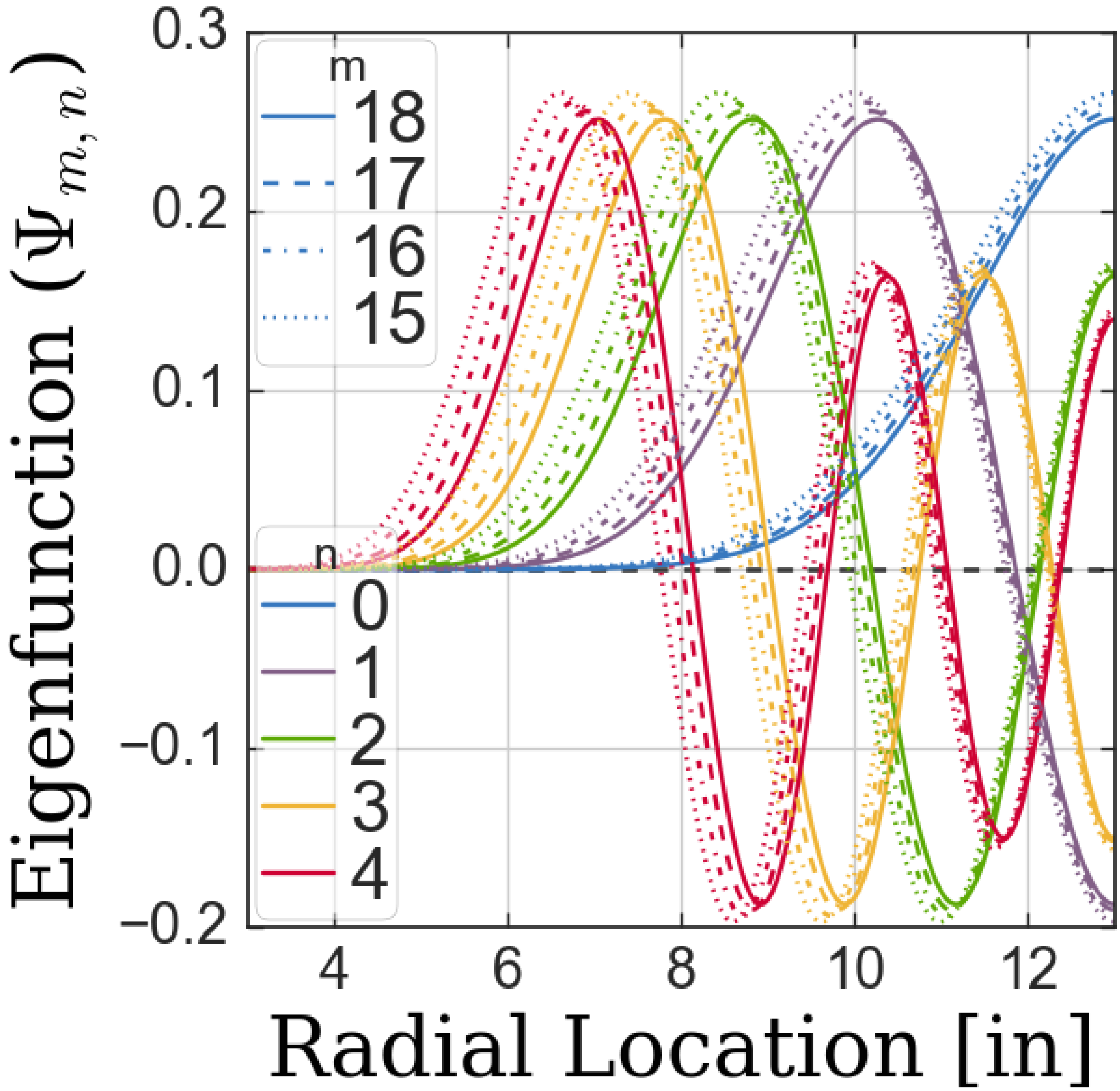

Figure 1: Eigenfunctions of all modes showing zero crossings corresponding to *n*

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 âĂŞz

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

```python
"""GLOBAL VARIABLES STORAGE
Logan Halstrom
MAE 298 AEROACOUSTICS
HOMEWORK 4 - TURBOMACHINERY NOISE
CREATED: 17 NOV 2016
MODIFIY: 06 DEC 2016

DESCRIPTION: Provide global variables for all scripts including wrapper.
"""

import numpy as np

#INPUTS

#CFD Engine Model Parameters
scale = 1/6 #CFD engine model scale
Nblade = 18 #Number of blades in engine fan
Nvane  = 1  #Uneven gust loading modeled as single vane
Ro = 13     #Model engine (outer) radius [in]
Ri = 3      #Model hub (inner) radius [in]
Linlet = 4  #Distace between inlet and z=0 plane [in]

#Flow Parameters
M = 0.525        #Engine flow mach number
a = 13503.937009 #Speed of Sound [in/s]
rho = 1.4988e-5 #density [slug/in^3]
RPM = 8326.3042  #Fan RPM
omega = RPM * (2 * np.pi / 60) * Nblade #angular frequency [rad/s]

#DATA OVERWRITE SWITCHES
datoverwrite = 1 #Overwrite data = 1

#LOAD/SAVE DIRECTORIES
datadir = 'Data'    #Source and processed data storage directory
savedir = 'Results' #

picdir = 'Plots' #Plot storage directory
pictype = 'png'      #Plot save filetype
# pictype = 'pdf'      #Plot save filetype

sigfigs = 4 #number of sig figs to save in data files
```

Appendix B: Main Data Processing and Plotting Script

```python
"""DATA PROCESSING PROGRAM
Logan Halstrom
MAE 298 AEROACOUSTICS
HOMEWORK 4 - TURBOMACHINERY NOISE
CREATED: 17 NOV 2016
MODIFIY: 06 DEC 2016

DESCRIPTION:

NOTE:
"""

#IMPORT GLOBAL VARIABLES
from hw4_98_globalVars import *

import numpy as np
import pandas as pd

from scipy.special import jn, yn #bessel functions
from scipy.special import jv, yv #bessel functions
from scipy.optimize import fsolve #linear solver
from scipy.optimize import broyden1 #non-linear solver



#CUSTOM PLOTTING PACKAGE
import matplotlib.pyplot as plt
import sys
sys.path.append('/Users/Logan/lib/python')
from lutil import df2tex
from lplot import *
from seaborn import color_palette
import seaborn as sns
UseSeaborn('xkcd') #use seaborn plotting features with custom colors
colors = sns.color_palette() #color cycle
markers = bigmarkers         #marker cycle


def AxialEigenvalFunc(x, m, ri, ro):
    """Determinant of axial flow boundary condition eigenfunctions, which is
    equal to zero.  Use scipy solver to get eigenvalues
    x  --> dependent variable (eigenvalue mu in axial flow eigenfunction)
    m  --> Order of bessel function
    ri --> inner radius of jet engine
    ro --> ourter radius of jet engine
    """
```

```python

    X = x * ro
    Y = x * ri
    return ( 0.5 * (jv(m-1, X) - jv(m+1, X)) * 0.5 * (yv(m-1, Y) - yv(m+1, Y))
           - 0.5 * (jv(m-1, Y) - jv(m+1, Y)) * 0.5 * (yv(m-1, X) - yv(m+1, X))
            )

def AxialEigenfunction(r, ri, m, mu):
    """Axial flow eigenfunction
    r  --> radial location to evaluate eigenfunction at
    ri --> inner radius of axial jet engine
    m  --> circumfrential acoustic mode
    mu --> eigenvalue (dependent on radial mode n)
    """
    X = mu * ri
    return ( jv(m, mu * r) - (0.5 * (jv(m-1, X) - jv(m+1, X)))
                 / (0.5 * (yv(m-1, X) - yv(m+1, X))) * yv(m, mu * r) )


def AxialWavenumber(mu, omega, c, M):
    """Calculate z-direction wavenumber (Kz+) for sound mode in axial flow
    mu --> eigenvalue of current mode
    omega --> angular frequency of flow
    c --> speed of sound
    M --> mach number of flow
    """
    K = omega / c #flow wave number
    Kz = (-M + np.emath.sqrt(1 - (1 - M ** 2) * (mu / K) ** 2 )) / (1 - M ** 2) * K
    return Kz

def CutOnCondition(Kz):
    """Determine if mode is cuton based on wave number
    Must be real to propagate (pos. imag.: decay exponential, neg.: expand)
    Must be negative for upstream propagation
    Kz --> axial wavenumber for current mode
    """
    cuton = 'No'
    #Test if no imaginary part (propagation)
    if Kz.imag == 0:
        #Test if upstream propagation
        if Kz.real < 0:
            cuton = 'Yes'
    return cuton


def GetGammaAmn(m, n, mu, p, r):
    """Get Gamma_mn and Amn from eigenfunction and acoustic pressure
    m     --> current circumfrential mode
```

```python
    n      --> current radial mode
    mu     --> eigenvalue for current m,n
    p   --> acoustic pressure at z=0 plane
    r   --> radius vector associated with p (goes from Ri --> Ro)
    """
    ri, ro = min(r), max(r) #inner/outer radii of engine

    #Calculate Gamma_mn for non m=n=0 case:
    Gam = (0.5 * (ro ** 2 - m ** 2 / mu ** 2)
                * AxialEigenfunction(ro, ri, m, mu) ** 2
          - 0.5 * (ri ** 2 - m ** 2 / mu ** 2)
                * AxialEigenfunction(ri, ri, m, mu) ** 2 )
    #Calculate Amn
    # Psi = lambda r: AxialEigenfunction(r, ri, m, mu)
    Psi = AxialEigenfunction(r, ri, m, mu)
    Amn = 1 / Gam * np.trapz( p * Psi * r, r)


    return Gam, Amn

def main():
    """Perform calculations for frequency data processing
    """

    ####################################################################
    ### PROB 1 - FIND EIGENVALUES ######################################
    ####################################################################


    eigenvals = pd.DataFrame() #solutions for eigenvalues

    ms = [18, 17, 16, 15] #circumfrential modes to solve for
    Nn = 5 #number of radial modes to solve for
    ns = range(Nn) #find first five radial modes

    # #tweak for round off error
        #     #values for calculating Wmn are very near zero.  small round-off
        #     #errors can cause Wmn=0 and PWL=infinity.
        #     #Change number of sigfigs to get correct Wmn for all cases
    rounds = [] #round eigenvalue to deal with floating point error
    for m in ms:
        rounds.append([None, None, None, None, None])
    rounds[0] = [8, None, None, None, None]

    for i, m in enumerate(ms):

        #guesses for root solutions
        x0s = np.linspace(0.1, 3, 300)
```

```
        mus = []
        for j, x0 in enumerate(x0s):

            #try each solution guess to see if it returns a good result
            try:
                #find solution corresponding to current guess
                mu = broyden1( lambda x: AxialEigenvalFunc(x, m, Ri, Ro), x0)
                mu = float(mu)
                #add to solution list if no error message
                mus.append( mu )
            except Exception:
                #Skip any solution errors
                pass

        #GET ONLY UNIQUE VALUES OF MU
        df = pd.DataFrame({'long' : list(mus), 'short' : list(mus)})
        df = df.round({'short': 6}) #shorten values to find unique ones
        df = df.drop_duplicates(subset='short') #drop duplicates in short vals
        df = df.sort_values('long') #sort eigen values from least to greatest
        df = df.reset_index() #reset indicies to same as n

        #Asign desired number of eigen values to solution dataframe
        mus = df['long'][:Nn]
        #Deal with floating point error
        for j, mu in enumerate(mus):
            if rounds[i][j] != None:
                mus[j] =  round(mu, rounds[i][j])
        eigenvals[m] = mus
        # eigenvals = eigenvals.round({m : 8})

    #SAVE EIGENVALUES
    #columns are m, rows are n
    eigenvals.to_csv( '{}/eigenvalues.dat'.format(datadir), sep=' ',
                        index=True )
    df2tex(eigenvals, '{}/eigenvalues'.format(datadir), dec=sigfigs)



    ####################################################################
    ### PROB 2 - PLOT EIGENFUNCTIONS ###################################
    ####################################################################

    R = np.linspace(Ri, Ro, 201) #Radial vector in engine
    line = ['-', '--', '-.', ':']

    fig, ax = plt.subplots(len(ms), sharex=True, figsize=[3, 12])
    for i, m in enumerate(ms):
        for j, n in enumerate(ns):
```

```
            ax[i].plot(R, AxialEigenfunction(R, Ri, m, eigenvals[m][j]),
                        label='n={}'.format(j)
                        )
            ax[i].set_ylabel('$\\Psi_{{{},n}}$'.format(m))
            ax[i].set_xlim([Ri, Ro])
    ax[i].set_xlabel('Radial Location [in]') #label x-axis on last subplot

    savename = '{}/2_eigenfunctions_subplot.{}'.format(picdir, pictype)
    SavePlot(savename)



    ###############

    _,ax = PlotStart(None, 'Radial Location [in]',
                            'Eigenfunction ($\\Psi_{{m,n}}$)', figsize=[6, 6])

    ax.plot([Ri, Ro], [0, 0], color='black', linestyle='--', linewidth=2, alpha=0.7  #zero
    mhandles = []
    mlabels = []
    nhandles = []
    nlabels = []
    labels = []
    for i, m in enumerate(ms):
        for j, n in enumerate(ns):
            h, = ax.plot(R, AxialEigenfunction(R, Ri, m, eigenvals[m][j]),
                        label='n={}'.format(j), color=colors[j],
                        linestyle=line[i],
                        )
            if j == 0:
                mhandles.append(h)
                mlabels.append(str(m))
            if i == 0:
                nhandles.append(h)
                nlabels.append(str(n))
            labels.append(j)
    ax.set_xlim([Ri, Ro])

    leg1 = PlotLegendLabels(ax, mhandles, mlabels, loc='upper left', title='m')
    # leg1 = ax.legend(mhandles, mlabels, loc='upper left', title='m',
    #                 # fancybox=True, frameon=True, framealpha=0.5,
    #                 # numpoints=1, scatterpoints=1, prop={'size':28},
    #                 # borderpad=0.1, borderaxespad=0.1, handletextpad=0.2,
    #                 # handlelength=1.0, labelspacing=0
    #                 )

    leg2 = PlotLegendLabels(ax, nhandles, nlabels, loc='lower left', title='n')
    plt.gca().add_artist(leg1)
```

```
savename = '{}/2_eigenfunctions.{}'.format(picdir, pictype)
SavePlot(savename)


###################################################################
### PROB 3 - WAVE NUMBER/CUT-OFF CONDITION ########################
###################################################################


wavenums = eigenvals.copy()

# wavenums = AxialWavenumber(eigenvals, omega, a, M)
wavenums = wavenums.applymap(lambda x: AxialWavenumber(x, omega, a, M))

#SAVE EIGENVALUES
#columns are m, rows are n
wavenums.to_csv( '{}/wavenumbers.dat'.format(datadir), sep=' ',
                    index=True )



#DETERMINE CUTON CONDITION
curNs = [0, 1, 2]
#sort by m
for m in ms:
    Kzs = []
    for n in curNs:
        Kzs.append(wavenums[m][n])

    df = pd.DataFrame({'m' : [m for x in curNs], 'n' : curNs, 'Kz' : Kzs})
    df['Cut-on'] = [CutOnCondition(Kz) for Kz in df['Kz']]
    #save wavenumber and cuton table
    df = df[['m', 'n', 'Kz', 'Cut-on']]
    df2tex(df, '{}/wavenumbers_m{}'.format(datadir, m), dec=sigfigs)


###################################################################
### PROB 4 - SPL OF REAL PRESSURE FIELD ###########################
###################################################################

#READ EXPERIMENTAL PRESSURE DISTRIBUTION AT Z=0 PLANE
    #Columns: Radius [in], Real pressure [psi], Imaginary pressure [psi]
df = pd.read_csv('{}/pressure_input.dat'.format(datadir), sep='\t',
                names=['R', 'pRe', 'pIm'])
#Create complex values of acoustic pressure
df['p'] = df['pRe'] + df['pIm'] * 1j
```

```python

    m, n = 18, 0

    PWLs = []

    for i, n in enumerate([0, 1, 2]):
        mu = eigenvals[m][n] #eigenvalue for curent (m,n)
        Gam, Amn = GetGammaAmn(m, n, mu, df['p'], df['R'])

        # print(Gam)
        # print(eigenvals[m][n])
        # print(Amn)

        #MODAL POWER
        Kz = wavenums[m][n]
        frac = Kz / (omega / a - Kz * M)
        Wmn = np.pi / (rho * a) * Gam * Amn * np.conj(Amn) * (
                (1 + M ** 2) * frac.real + M * (abs(frac) ** 2 + 1) )

        #SOUND POWER LEVEL
        PWL = 10 * np.log10( abs(Wmn) ) - 10* np.log10(7.3756e-13)

        PWLs.append(PWL)

    #SAVE POWERLEVELS
    PWLs = pd.DataFrame({m : PWLs})
    PWLs.to_csv( '{}/powerlevels.dat'.format(datadir), sep=' ', index=True )
    df2tex(PWLs, '{}/powerlevels'.format(datadir), dec=sigfigs)



if __name__ == "__main__":


    main()
```