Angle Check

This program is a one time program that was used to plot the angle versus the zpC.

BOA

This method is the main way that the Automated Equivalent Source Method finds the optimal solution. It calls simAnnealing and FGS. The easiest way to use this template is to plug in the parameters that it asks for.

errorFunction

This function determines the differences between the measured data and the calculated data. This will be the weighting device for the BOA.

function [ errorMap, errorTotal ] = errorFunction( SPLc, SPLm, Pref, range, type )

errorMap = the returned grid that is that takes the difference for each point between the measured and the calculated data.

errorTotal = the total error according the used equation

type = if 0, then use equation from Jessica Morgan’s paper, given as

if 1, then use a least squares equation

SPLm = the Sound Pressure Level Map for the measured data

SPLc = the Sound Pressure Level Map for the calculated data

Pref = the reference microphone pressure, usually 20 microPascals.

range = the amount db from the maximum db in SPLm to consider in the error formula. For example, if my maximum db is 140, and range is 10, anything that is below 130 db is not considered when calculating errorTotal

ESM

This stands for Equivalent Source Method, not to be confused with an Automated Equivalent Source Method. This script simulates the sound field on a set of microphones given a set of parameters for an Equivalent Sound Source (see code for descriptions of parameters). It then displays, the resulting source, sound field, and errormap. This is what will be used for analysis once the optimal equivalent source is obtained.

F22Holloman2009Geometry7

This is where the microphone positions are retrieved from.

FGS

This is the fast gibbs sampler that is used as part of the Bayesian analysis. It takes the parameters that are outputted from the simAnnealing. It outputs the accepted values which are used by BOA to generate the PPD’s.

initSources

This function returns two sets of monopole line sources: one with correlated properties and one with uncorrelated properties. This function is used to initialize the monopole sources based on the variables that are given for input. It uses a Rayleigh distribution when initializing.

function [ sourcesU, sourcesC ] = initSources(x,y,zpU,sdU,AmaxU,zpC,...

sdC,AmaxC,n,dist,angle,freqs,c

sourcesU = The line of uncorrelated sources

sourcesC = The line of correlated sources. This is a [n,5,nf] matrix, where n is the number of sources and nf is the number of frequencies that correspond to freqs.

x = initial x position in meters

y = initial y position in meters

zpU = peak position of the uncorrelated sources.

sdU = standard deviation of the Rayleigh distribution for the uncorrelated sources.

AmaxU = The maximum amplitude for the a source of the uncorrelated sources.

zpC = peak position of the correlated sources.

sdC = standard deviation of the Rayleigh distribution for the correlated sources.

AmaxC = The maximum amplitude for the a source of the correlated sources.

n = number of sources in each line. 100 has been the most commonly used.

dist = The distance between each source in meters. 0.1 is commonly used.

angle = The angle in radians that the correlated sources project at. 130 \* Pi / 180 is commonly used.

freqs = The frequencies that are considered in the band. 10 frequencies are usually listed for each band.

c = The speed of sound in meters. 343 is most commonly used.

loadData

This function is still under construction. Once all data sets are available, this function will be able to easily select which plane and frequency and engine speed you want to look at.

localSearch

This function looks for the optimal solution in a given range for each variable. This function searches for solution at a certain step size, and then a smaller step size, until it finds the best solution.

function [ zpU,zpC, Amax, sdU,sdC, certainty, error] = localSearch(zpUmin, zpUmax,zpCmin,zpCmax, A1min,A1max,sdUmin,sdUmax,sdCmin,sdCmax,x,y,n, dist,Pref, angles, freqs, c, micPositions, SPLm, k, stepnumber, refinenumber, recursenumber)

zpU,zpC,sdU,sdC are all there equivalents from initSources.

Amax = the ratio of

Certainty = how far +- the answer the actual solution could be

Error = the total error as calculated by errorFunction

~min = the minimum value that the variable could be

~max = the maximum value that the variable could be

x,y,n,dist,angles,freqs,c are the same as initSources

Pref = the reference sound pressure, usually 20 micro pascals

micPositions = the array that describes the positions of all microphones

SPLm = the Sound Pressure Levels of the measured data.

k = the 2\*Pi \* frequency of all the frequencies listed in freqs.

Stepnumber = the number of steps that localSearch does on a single variable

Refinenumber = the number of iterations localSearch uses to find a common solution

Recursenumber = the number of times the range shrinks

matchSPL

It multiplies the calculated SPL until it matches the measured SPL. This function is usually called before using the errorFunction, so that the results are within the same range.

function [ SPL ] = matchSPL( SPLc, SPLm, percentile )

SPL = the new calculated Sound Pressure Level

SPLc = the original calculated Sound Pressure Level

SPLm = the measured SPL

Percentile = the ratio to match the maximum of SPLc to the maximum of SPLm. i.e.

plotDistribution

A script that makes it really easy to plot a source of monopole distribution. Takes in sourcesU and sourcesC.

plotSPL

A script that makes it really easy to plot a SPL map. Takes in the SPL map, the microphone positions, a title for the graph, and a type.

If type = 0, then it uses the default color scheme

If type = 1, then it uses Alan Wall’s color scheme

Plane2ABtall\_nov2011

This is data taken from the F22 Raptor on plane2 during AfterBurner. This is where the initial test cases are being taken from. More data will be tested in the future.

simAnnealing

This function is the function that finds the best solution after a given amount of time, by using simulated annealing principles.

Boundaries = [zpUmin,zpUmax,zpCmin,zpC,max,AmaxMin,Amax,Max,sdUmin,sdUmax,sdCmin,sdCmax,distMin,distMax]

Geometry = [x,y,angle,c,Pref,n]

//Annealing Parameters

T = starting temperature

Tfinal = ending temperature

Rate = how the temperature drops between each step, T = Rate \* T

Num\_steps = How many times to run before dropping temperature

SPL\_Range = how much db is considered when calculating the error

The total time for the simulated annealing to run is

Where is the average of time for one simulation

simulation

This function takes the two lines of monopole sources and simulates the pressure levels at the microphone positions. It returns .

function [ measurement ] = simulation(sourcesU,sourcesC,micPositions,k)

measurement =

sourceU = the line of uncorrelated sources taken from initSources

sourceC = the line of correlated sources taken from initSources

micPositions = the positions of all the microphones that need to be simulated at.

k = 2\*pi \*freqs

takeSPL

A simple converter function from to SPL, with a reference pressure Pref

TestPlane2

A calculated test case used previous to having the plane2AB data.

TestSuite

This script has been used to test all the functions. As functions are confirmed to work, the code is commented out. This script is continually changed as more tests are conducted.

VariableMapper

This script is used to help determine the shape of the variable space. This will help in determining how the BOA will function. It works by holding 3 variables constant and then varying two variables to see how the two variables relate to each other. This script will only need to be ran once, to collect. After data is collected, it can be analyzed without further use of the script.