OTM 33A Appendix F1

Version 1.3 Change Description:

This Appendix contains information on emission rate assessment procedures using the simple point source Gaussian (PSG) stationary analysis approach. This revision (Version 1.3, July 2016) updates OTM 33A Appendix F1 by providing improved analysis code implementation examples and test files. The original description of the PSG approach and the original MATLAB®-based PSG analysis code (Appendix F1-B) is retained but the latter is no longer recommended for use. Updated and improved versions of the PSG analysis code written in both MATLAB® (Appendix F1-F) and “R” (Appendix F1-G) are provided. The improved versions of the analysis code fix several software issues in the original code that could lead to errors in data processing. These coding issues include errors in batch processing of files, setting cut internal filter levels, and binning of information in the fitting procedure. For files executed using the EPA OTM 33A protocol, these coding issues may produce errors in source strength estimates on the order of 5% on average. If users deviate from the OTM 33A acquisition protocol (e.g. shorter observation times, improper instrument rotation) these code issues could lead to more serious errors so use of the updated versions of the software is recommended. In addition to the updated analysis code, twenty test files from controlled release experiments are provided along with an output summary file. As described in OTM 33A, the original PSG code required a particular orientation of the 3-D sonic requirement. The new versions of the analysis code have additional processing elements that relax this requirement.

TABLE OF CONTENTS

Tables

Figures

Appendices

Standard Operating Procedure for Analysis of US EPA Geospatial Measurement of Air Pollution-Remote Emission Quantification by Direct Assessment (GMAP-REQ-DA) Method Data for Methane Emission Rate Quantification using the Point Source Gaussian Method

Scope: The following describes procedures for analysis of data collected with the US EPA Geospatial Measurement of Air Pollution-Remote Emission Quantification–Direct Assessment (GMAP-REQ-DA) technique OTM 33A to yield emission rate estimates using the Point Source Gaussian (PSG) approach. This procedure provides guidance on preparing input data files for use with the PSG software program, importing the input files into the software, and instructions for processing the data.

Purpose: To provide guidance on analyzing data collected with the GMAP-REQ-DA OTM 33A technique to yield emission rates estimates using the PSG method.

1.0 Overview

OTM 33A is a mobile inspection approach to assess emissions of air pollutants from proximate ground level point sources. The measurement system consists of a fast-response concentration measurement instrument (CMI), global positioning system (GPS), and wind monitors, installed in a vehicle. The OTM 33A source emissions estimates are made from a stationary location downwind of the source. The EPA-developed research systems use custom software to control operation of instrumentation, and acquisition of data. The system has used to assess emissions from upstream oil and gas production sites (well pads), but has utility for monitoring emissions from other emission sources with similar physical characteristics.

Data elements collected with the GMAP-REQ-DA OTM 33A technique include target analyte concentration, GPS data, 3-D Sonic Anemometer data, and AIO Compact Weather Station data. These data elements can be used to estimate site emission rates. The current document describes the procedures for calculating emission rates using PSG approach using data analysis software programs. The programs pre-process raw data collected during the measurement surveys, create data input files for the PSG algorithm, and to calculate methane emission rate estimates from the survey sites. Section 2.0 of this document presents details on the PSG algorithm, and Section 3.0 provides procedures for using the analysis software for calculating emission rates. Section 4.0 describes a quality-assurance tool used to assess the location of detected emission sources.

2.0 Point Source Gaussian Emission Estimation Method

The first step in pre-processing the data is to time-align the raw data to correct for the delay time it takes for the sample to reach the analyzer through the sampling line. The exact delay time is determined by conducting delay tests in the field as part of the in-field calibration checks. The delay time will vary slightly for each site, so the response tests should be conducted on the first day of each measurement campaign. After the delay time has been determined, it is input to the analysis software, and the software performs the time alignment correction (instructions for inputting the delay time to the analysis software are discussed in Section 3).

The second step in pre-processing the data is to rotate the 3-D Sonic Anemometer data to streamlined coordinates. The anemometer outputs velocity data in the *u*, *v*, and *w* coordinates. The coordinate system is first rotated through the angle formed by the average wind velocity in the *u* and *v* directions, as shown in Equation 1 below:

$$\theta_1 = \tan^{-1}\left(\frac{avg\ u}{avg\ v}\right) \tag{1}$$

The rotated average velocity values (*u′*, *v′*) from the original coordinate system are calculated using the following rotation matrix:

$$\begin{bmatrix} v' \\ u' \end{bmatrix} = \begin{bmatrix} \cos(\theta_1) & \sin(\theta_1) \\ -\sin(\theta_1) & \cos(\theta_1) \end{bmatrix} \begin{bmatrix} v \\ u \end{bmatrix} \tag{2}$$

The coordinate system is then rotated through the angle formed by the average wind velocity in the *v* and *w* directions:

$$\theta_2 = \tan^{-1}\left(\frac{avg\ w}{avg\ v'}\right) \tag{3}$$

The rotated average velocity values (*v″*, *w′*) are calculated using the following rotation matrix:

$$\begin{bmatrix} v'' \\ w' \end{bmatrix} = \begin{bmatrix} \cos(\theta_2) & \sin(\theta_2) \\ -\sin(\theta_2) & \cos(\theta_2) \end{bmatrix} \begin{bmatrix} v' \\ w \end{bmatrix} \tag{4}$$

In the revised coordinate system, $v'' = v;\ u' = u;\ w' = w$

The rotated wind speed value is expressed as:

$$WS = \sqrt{v^2 + u^2} \tag{5}$$

The rotated 3-dimensional azimuthal wind direction is expressed as:

$$WD = 180 + \left[a\tan 2\left(\frac{avg\ u}{avg\ v} \right) * \frac{180}{\pi} \right] \tag{6}$$

The rotation calculations are performed automatically by the analysis software after the raw data files are input to the program.

After pre-processing has been completed, the software is used to determine the local stability class of wind conditions during the time of the stationary measurements. The local wind stability class is determined by the analysis software using the standard deviation of 2-dimensional wind direction and turbulent intensity (ratio of the standard deviation of the wind speed to the average wind speed). When determining the local wind stability class based on the standard deviation of 2-dimensional wind direction (*StdWD*), the following relationships are used:

$$Stability\ Class = 1\ if\ Std\ WD > 27.50°$$
$$Stability\ Class = 2\ if\ 23.50° < Std\ WD \le 27.50°$$
$$Stability\ Class = 3\ if\ 19.50° < StdWD \le 23.50°$$
$$Stability\ Class = 4\ if\ 15.50° < StdWD \le 19.50°$$
$$Stability\ Class = 5\ if\ 11.50° < StdWD \le 15.50°$$
$$Stability\ Class = 6\ if\ 7.50° < StdWD \le 11.50°$$
$$Stability\ Class = 7\ if\ StdWD \le 7.50°$$

When determining the local wind stability class based on the turbulent intensity (*turbuint*), the following relationships are used:

$$Stability\ Class = 1\ if\ turbu\text{int} > 0.205$$
$$Stability\ Class = 2\ if\ 0.180 < turbu\text{int} \le 0.205$$
$$Stability\ Class = 3\ if\ 0.155 < turbu\text{int} \le 0.180$$
$$Stability\ Class = 4\ if\ 0.130 < turbu\text{int} \le 0.155$$
$$Stability\ Class = 5\ if\ 0.105 < turbu\text{int} \le 0.130$$
$$Stability\ Class = 6\ if\ 0.08 < turbu\text{int} \le 0.105$$
$$Stability\ Class = 7\ if\ turbu\text{int} \le 0.080$$

The average local wind stability class (*pgi*) is defined as the average of the stability class determinations based on the standard deviation

of 2-dimensional wind direction and the turbulent intensity parameter. The average local wind stability class for a particular stationary survey is classified using the following table, where Class 1 represents extremely unstable wind conditions, and Class 7 represents extremely stable wind conditions.

$$Stability\ Class = 1\ if\ \ pgi < 1.5$$
$$Stability\ Class = 2\ if\ 1.50 \le pgi\ < 2.5$$
$$Stability\ Class = 3\ if\ 2.50 \le pgi < 3.50$$
$$Stability\ Class = 4\ if\ 3.50 \le pgi < 4.50$$
$$Stability\ Class = 5\ if\ 4.50 \le pgi < 5.50$$
$$Stability\ Class = 6\ if\ 5.50 \le pgi < 6.50$$
$$Stability\ Class = 7\ if\ \ pgi \ge 6.50$$

After the local wind stability class has been determined, the values of horizontal (σ_y) and vertical (σ_z) dispersion are determined from point source dispersion tables using the measured source distance and the determined average local wind stability class. The tables used for determining the values of σ_y and σ_z are presented in Appendix A of this document.

Next, the analysis software determines the average background methane concentration by calculating the average of the lowest 5% of all methane concentration values collected during the survey. The background concentration value is then subtracted from all concentration data collected during the survey. The concentration data are then sorted into 5-degree incremental bins based on wind direction during the time of the measurements, and the average methane concentration for each bin is calculated. The plot of concentration versus wind direction is then fitted to a Gaussian function during the stationary survey. Prior to performing the Gaussian fit, the software utilizes a “cut filter” value of 2%. In this approach, data from a particular bin are removed from the analysis if the total number of data points in any particular bin is not greater than 2% of the total number of data points collected during the survey. The peak value of the Gaussian fit is the peak methane concentration (***a1*** parameter), which is used in the calculation of plane-integrated methane concentration. Figure 1 below presents an example plot of methane concentration data as a function of prevailing wind direction, fitted using a Gaussian function. The plot shows the determined ***a1*** value as 2.53 ppm.

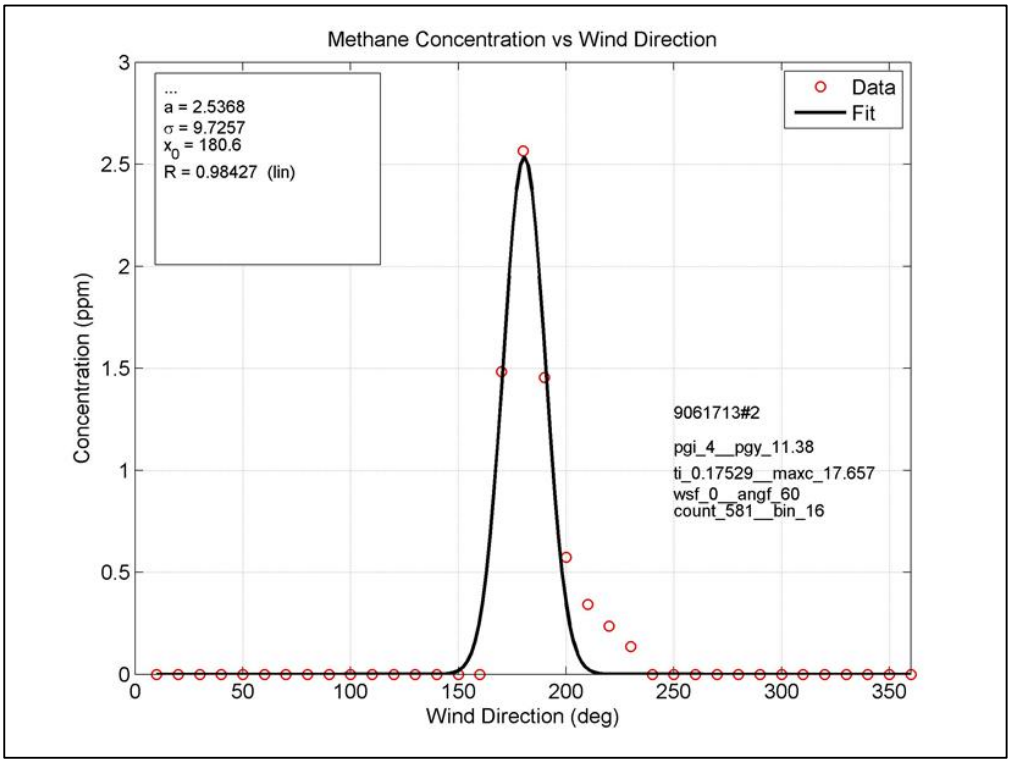

Figure 1. Gaussian Fit of Methane Concentration as a Function of Wind Direction

After the peak methane concentration (***a1***) has been determined, it is converted from units of ppmv to units of grams per cubic meter. The plane-integrated methane concentration is then determined using the following two-dimensional Gaussian integration:

$$G(y,z) = a1 \times \iint \exp - \left(\frac{(y)^2}{2\sigma_y^2} + \frac{(z)^2}{2\sigma_z^2} \right) = 2\pi \times a1 \times \sigma_y \times \sigma_z \qquad (7)$$

Where:

y= crosswind distance (m)

z= vertical distance (m)

a1= peak average methane concentration (g/m^3) as determined by Gaussian fit

σ_y = standard deviation of the crosswind plume concentration (m)

σ_z = standard deviation of the vertical plume concentration (m)

This expression is multiplied by the calculated wind speed during the survey (determined using Equation 5) to yield the estimated emission rate (in grams per second) of the target compound.

3.0 Utilizing the OTM 33A PSG Method to Calculate Emission Rates

The estimation of emision rates using the PSG approach requires a computer program to perform the calculations described in previous sections. Several group have implemented the OTM 33A PSG algorithm using different data analysis computer programs. This appendix contains three example computer program implementations MATLAB® and "R". The first part of this description and Appendix F1-B contains the original version of the PSG analysis code written in MATLAB®. This version is retained for historical purposes but is not recommend for use and several errors in the code where found and repaired in the updated versions of the code contained in Appendices F1-F and F1-G. The versions of the PSG analysis code are described below. These codes sets are contained in a Zip file as part of this Appendix.

Appendix F1-B: MATLAB®-based OTM33_PSG_031913.m – Discontinued

Appendix F1-F: MATLAB®-based OTM33_PSG_082715_Histcount.m

Appendix F1-G: R-based calcPSG.R

3.1 Converting Raw Data Output Files

The EPA version of the OTM 33A system uses a custom data analysis program written in LabView (National Instrument Inc., Austin, TX) that saves data in a proprietary format call *TDMS*. The following file conversation step is only necessary for system using *TDMS* formats. Prior to executing data analysis, the raw acquisition data files ".tdms" format, to STR files in ".csv" format. The file conversion is done accomplished the *TDMS to CSV TranslatorRev 1.2.2. Build 5/28/2013* program. Data files are converted using the following steps:

1. Open the *TDMS to CSV TranslatorRev 1.2.2. Build 5/28/2013* program. The program main screen will be displayed.

2. Select the data file to be converted by clicking the "browse" icon, located to the right of the "Path to Single File" box in the program main screen. Select the desired file and hit OK as shown in Figure 2.

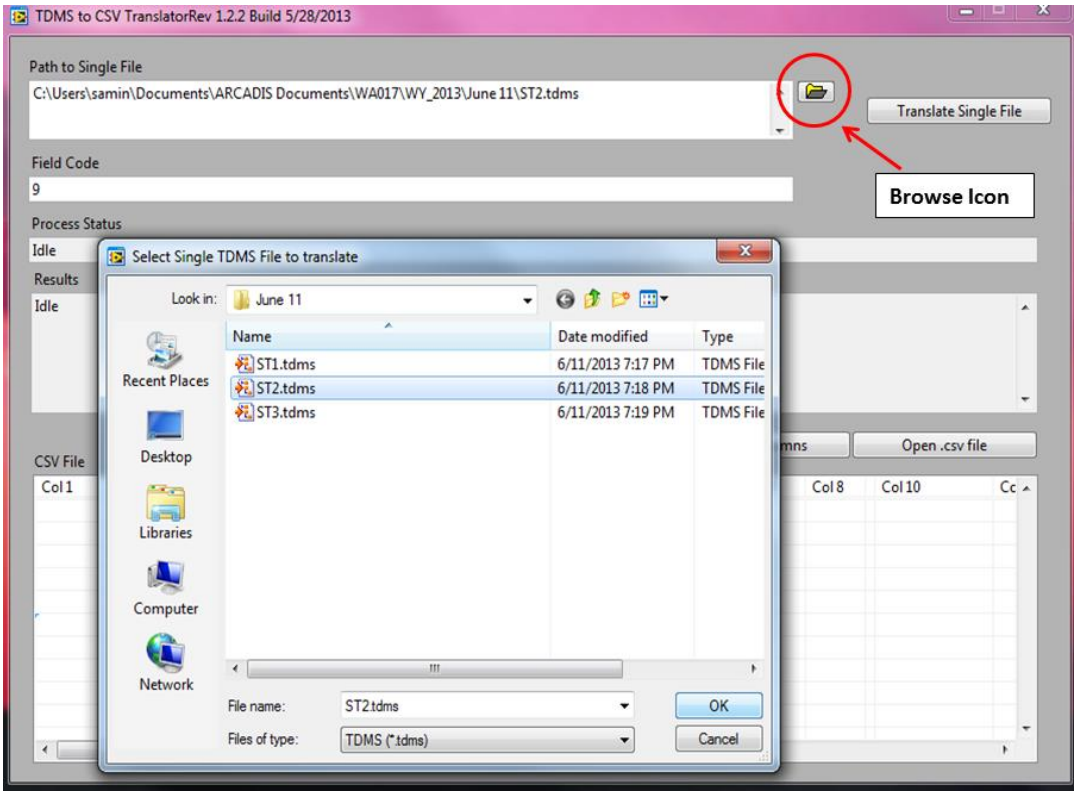

Figure 2. File Translator Program Main Screen Showing Selection of File for Conversion

1. Under the “Field Code”, enter the name of the field study.

2. Click on the “Translate Single File” tab located in the top right corner of the main screen, as shown in Figure 2.

3. The program will begin creating a new STR file in .csv format for each site survey from the particular measurement campaign. After the new file has been created, a confirmation message will appear in the “Results” box on the program main screen, as shown in Figure 3. The new file is created in the same folder as the original .tdms file.

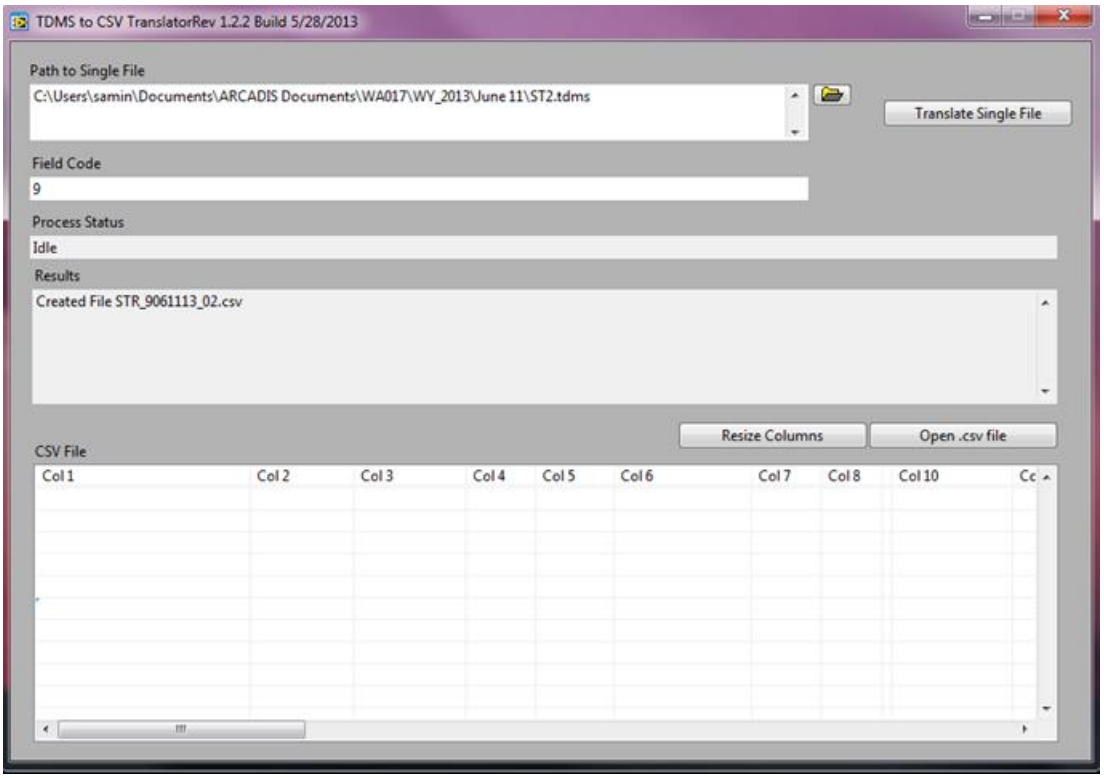

Figure 3. File Translator Program Main Screen Showing Confirmation of File Conversion

An example STR data output file in .csv format is shown in Appendix C of this document.

3.2 Initializing the OTM 33A MATLAB®-based data analysis program

After all raw data output files from a field study have been converted to "STR" files in .csv format (or .xls format for the test files contained here), create a project folder on the C: drive of the computer being used for data analysis. The folder can be labeled using any name convenient to identify the measurement project. Create a subfolder within the project folder named "Analysisfiles". Copy and paste all converted "STR" files into the "Analysisfiles" subfolder. The subfolder should also contain the *MATLAB®-based OTM33_PSG_031913.m (discontinued) or "OTM33_PSG_082715_Histcount.m"* code to analyze the data, and all the associated files necessary to run the program.

Prior to initiating the program to calculate emission rates, it is necessary to input several parameters that the program uses in the emission calculation. These parameters are input to a user input

spreadsheet, *Parameter.xls*, shown in Figure 4 (*OTM33_PSG_031913.mver)*.

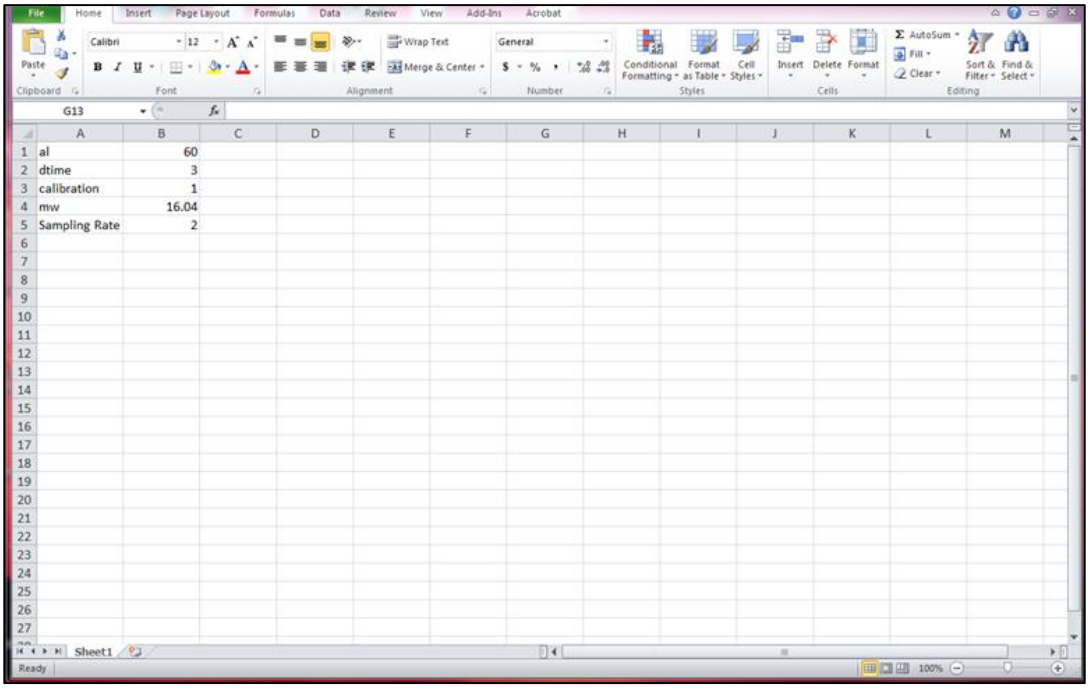

Figure 4. Screenshot of Parameter.xls User Input Spreadsheet

The first parameter, ***al***, is the wind direction angle filter (in degrees). This parameter allows the user to filter data used in the emission calculation based on the prevailing wind direction during the time the data was collected. For example, a wind direction angle filter of 60° would direct the emissions calculation program to use data from time periods only when the prevailing wind direction was ± 60° from the compass heading of the source location, with respect to the location of the measurement vehicle. A wind direction angle filter of 180° would direct the program to use all data collected for the emissions calculation (e.g. no wind filter). In preparation for the preliminary run of a particular dataset, the ***al*** parameter should be set to 180°.

The next parameter, ***dtime***, is the instrument delay time (in seconds), determined from the delay tests conducted during the measurement campaign, and described in Section 2.

The third parameter, ***calibration***, is a calibration factor that can be used to correct concentration data, based on pre-deployment instrument

calibration experiments. The default value for this parameter is 1, but this value may be adjusted depending on the results of the calibration experiments (e.g. if the results of the calibration experiments indicate the analyzer has a 20% bias in concentration measurements, the ***calibration*** value would be entered as 1.20).

The fourth parameter, ***mw***, is the molecular weight of the target compound, in grams/mole. Because this document describes methane emissions calculations, this value is entered as 16.04.

The final parameter, ***Sampling Rate***, is the data sampling rate, in Hz, used during the measurement campaign. The default value for this parameter is 2.

After all parameter values are entered into the spreadsheet, the file is saved in the “Analysisfiles” subfolder.

3.3 Running the *MATLAB®- based* Program for Emissions Calculations

The program is now ready to calculate emission rates from the surveys. Run the program by clicking on the “Debug” icon, and then click on “Run”, as shown in Figure 5.

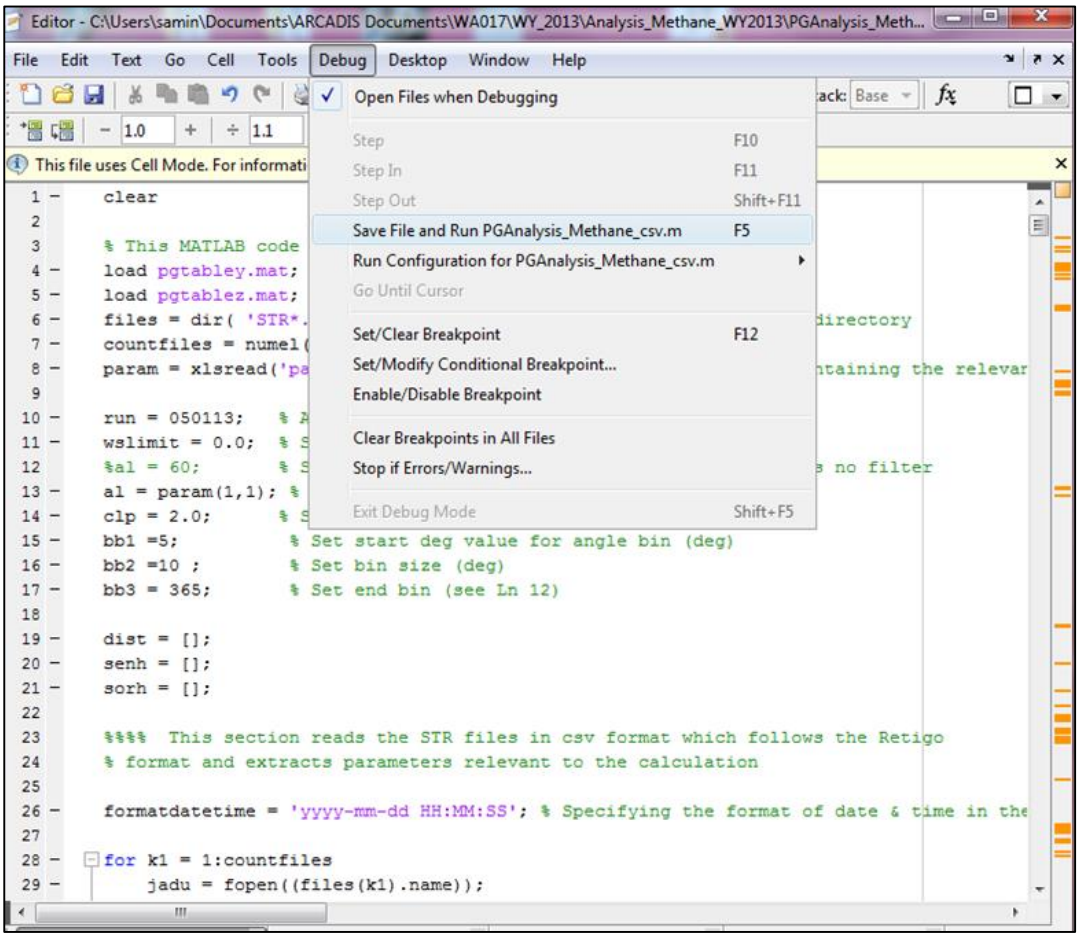

Figure 5. Screenshot from *PG_Analysis* program showing initiation of the program

When the emissions calculations are complete, the program will create a results output file, *Summary_Processed_Methane.xls*, in the "Analysisfiles" subfolder, which includes emission estimate results from all surveys conducted during a particular measurement campaign. An example results output file is shown in Appendix D of this document, including an explanation of all data fields.

In addition to the *Summary_Processed_Methane.xls* file, the program will create a "processed STR" file for each site survey from the measurement campaign. Data in the processed STR file have been time-aligned based on input from the *Parameter.xls* file. Note that the "Analysisfiles" subfolder from a particular measurement campaign will include one *Summary_Processed_Methane.xls* file, and multiple processed STR files (one for each site survey).

Next, open the *Summary_Processed_Methane.xls* file and inspect the values shown in columns **<u> and <w>**. All values in each of these columns should be close to "0", indicating that the orientation of the 3-D Sonic Anemometer instrument in the field was correct. After this has been confirmed, re-open and edit the *Parameter.xls* input file (described in Section 3.2) by changing the value of the **al** parameter to the default value of 60°. In the "Analysisfiles" subfolder, select and delete all "processed STR" files. Then, re-run the *Analysis_*program using the procedure described at the beginning of this section. A new *Summary_Processed_Methane*.xls file will be created (overwriting the original file). This will be the final data output results file.

Note: The Analysis_program is run utilizing a MATLAB® platform, and the user must have MATLAB® installed on their personal computer to use the program for emissions measurement calculations.

4.0 Wind Rose/Concentration Plot Quality Assurance Program

The wind rose/concentration plot quality assurance program creates a plot that displays the measured methane concentrations from each site survey, plotted on a four-quadrant graph, as a function of prevailing wind direction during the measurements. The origin of the graph represents the location of the measurement vehicle during the survey. Methane concentration values are plotted using a color scale, with the highest methane concentrations shown in red and orange colors, the lowest concentrations shown in blue and green, and intermediate concentration values shown in yellow. Concentrations plotted in the northeast quadrant of the graph represent concentrations measured while winds were blowing from the northeast. Concentrations plotted in the southeast quadrant represent measurements with prevailing winds from the southeast, etc. The plot is created to verify the location of the source of emissions detected with the instrumentation (the approximate location of the emissions source is found by starting at the origin of the graph and following the axis of highest methane concentrations, which should terminate in the vicinity of the emissions source. The program also creates a wind direction histogram, showing the frequency of prevailing wind directions.

The plots are created using a custom-designed MatLab® based program called *WindCH4Source.m.* The *WindCH4Source.m* program code is presented in Appendix E of this document. The *WindCH4Source.m* is run using the following procedures:

1. Copy all “processed STR” files from the “Analysisfiles” subfolder (discussed in Section 3.3) into a new folder that also includes the *WindCH4Source.m* MATLAB® code.
2. The program can be executed using either absolute wind data from the AIO Compact Weather Station, or azimuthal wind direction data from the 3D Sonic Anemometer. The wind direction parameter is represented by the variable “**wddir**” in the program code, as shown in line 16 of Figure 6. No changes to the program code are necessary to utilize the AIO Compact Weather Station wind data to create the wind rose/concentration plots. However, in order to use the 3D Sonic Anemometer azimuthal wind direction data to change the plots, the program must be modified by deleting the “%” in line 15, and inserting “%” in line 16, before the variable “wddir”.

The code is executed by clicking on the “Run” icon, and then clicking on “Run WindCH4Source” from the dropdown menu, as shown in Figure 6.

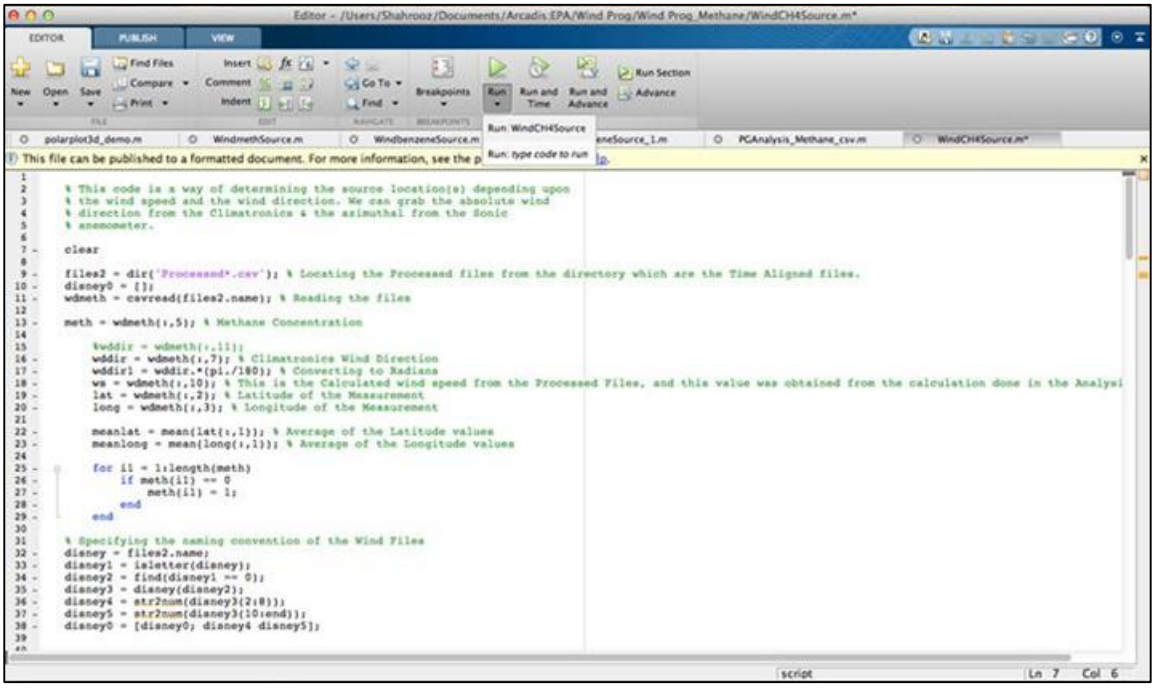

Figure 6. Screenshot of *WindCH4Source.m* Program Code

The *WindCH4Source.m* program creates plots for each processed STR file (each site survey from a particular measurement campaign), a *CH4Source* file (shown in Figure 7), and a *WindHist* file (shown in Figure 8). The *CH4Source* file is used to confirm the location of the emissions source, with respect to the location of the measurement vehicle.

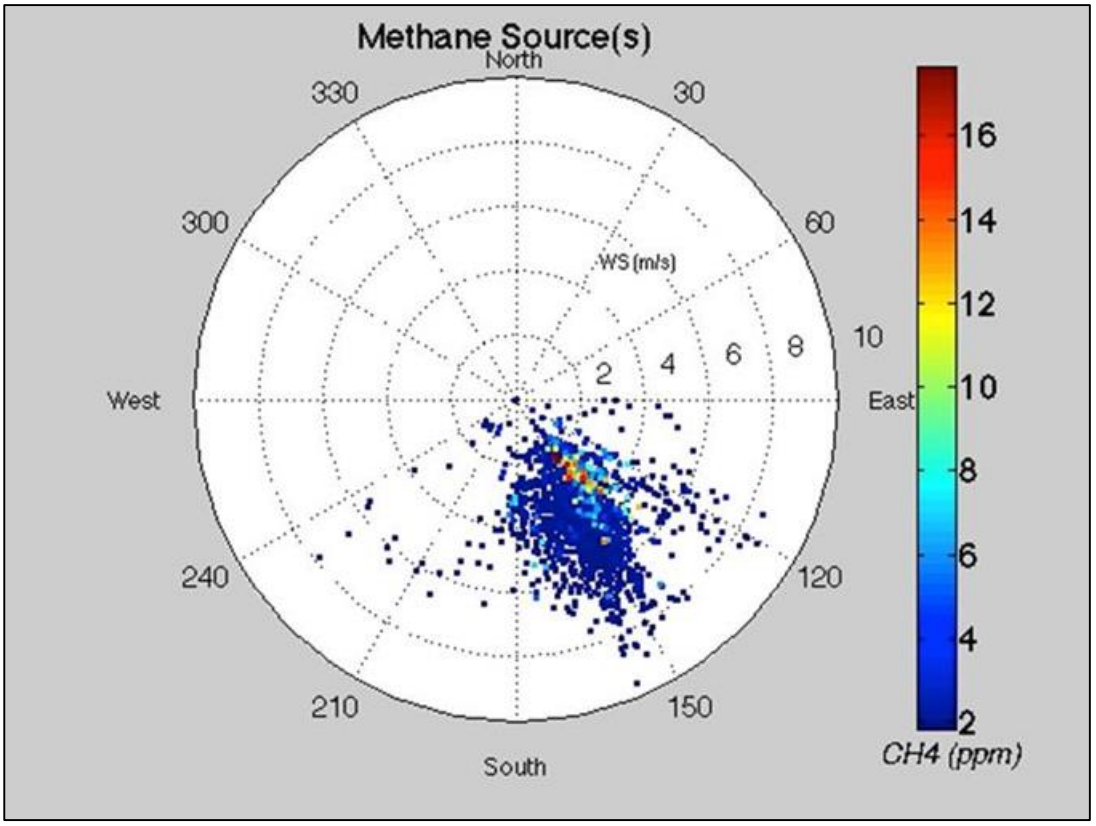

Figure 7. Example CH4Source Plot from the *WindCH4Source.m* Program

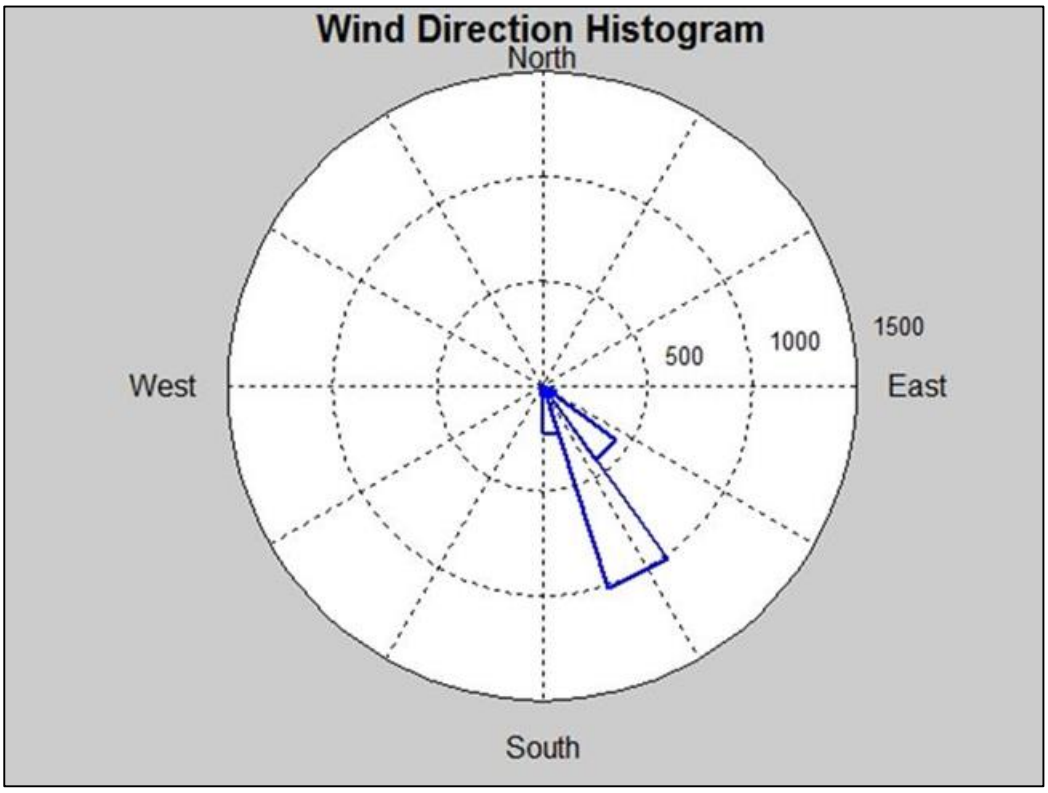

Figure 8. Example WindHist Plot from the *WindCH4Source.m* Program

Appendix A:
Point Source Dispersion Table
for Determining the Values of σ_y and σ_z

Table A-1. Point Source Dispersion Table for Determining Value of Sigma-Y (meters)

Distance from Source (meters)	Stability Class 1	Stability Class 2	Stability Class 3	Stability Class 4	Stability Class 5	Stability Class 6	Stability Class 7
1	2.32	1.89	1.46	1.14	0.83	0.68	0.53
2	2.56	2.10	1.63	1.29	0.94	0.78	0.61
3	2.80	2.31	1.81	1.43	1.06	0.87	0.68
4	3.05	2.52	1.99	1.58	1.18	0.97	0.76
5	3.29	2.73	2.16	1.73	1.29	1.06	0.84
6	3.53	2.94	2.34	1.87	1.41	1.16	0.91
7	3.78	3.15	2.52	2.02	1.52	1.26	0.99
8	4.02	3.36	2.69	2.17	1.64	1.35	1.07
9	4.26	3.57	2.87	2.31	1.75	1.45	1.14
10	4.50	3.78	3.05	2.46	1.87	1.54	1.22
11	4.75	3.99	3.22	2.60	1.99	1.64	1.29
12	4.99	4.20	3.40	2.75	2.10	1.74	1.37
13	5.23	4.41	3.58	2.90	2.22	1.83	1.45
14	5.48	4.62	3.75	3.04	2.33	1.93	1.52
15	5.72	4.83	3.93	3.19	2.45	2.02	1.60
16	5.96	5.04	4.11	3.34	2.56	2.12	1.68
17	6.21	5.25	4.28	3.48	2.68	2.22	1.75
18	6.45	5.46	4.46	3.63	2.80	2.31	1.83
19	6.69	5.66	4.64	3.77	2.91	2.41	1.91
20	6.93	5.87	4.81	3.92	3.03	2.50	1.98
21	7.18	6.08	4.99	4.07	3.14	2.60	2.06
22	7.42	6.29	5.17	4.21	3.26	2.70	2.14
23	7.66	6.50	5.35	4.36	3.37	2.79	2.21
24	7.91	6.71	5.52	4.51	3.49	2.89	2.29
25	8.15	6.92	5.70	4.65	3.61	2.99	2.36
26	8.39	7.13	5.88	4.80	3.72	3.08	2.44
27	8.64	7.34	6.05	4.94	3.84	3.18	2.52
28	8.88	7.55	6.23	5.09	3.95	3.27	2.59
29	9.12	7.76	6.41	5.24	4.07	3.37	2.67
30	9.36	7.97	6.58	5.38	4.18	3.47	2.75
31	9.61	8.18	6.76	5.53	4.30	3.56	2.82
32	9.85	8.39	6.94	5.68	4.42	3.66	2.90
33	10.09	8.60	7.11	5.82	4.53	3.75	2.98

Distance from Source (meters)	Stability Class 1	Stability Class 2	Stability Class 3	Stability Class 4	Stability Class 5	Stability Class 6	Stability Class 7
34	10.34	8.81	7.29	5.97	4.65	3.85	3.05
35	10.58	9.02	7.47	6.11	4.76	3.95	3.13
36	10.82	9.23	7.64	6.26	4.88	4.04	3.20
37	11.07	9.44	7.82	6.41	4.99	4.14	3.28
38	11.31	9.65	8.00	6.55	5.11	4.23	3.36
39	11.55	9.86	8.17	6.70	5.23	4.33	3.43
40	11.79	10.07	8.35	6.85	5.34	4.43	3.51
41	12.04	10.28	8.53	6.99	5.46	4.52	3.59
42	12.28	10.49	8.70	7.14	5.57	4.62	3.66
43	12.52	10.70	8.88	7.28	5.69	4.71	3.74
44	12.77	10.91	9.06	7.43	5.80	4.81	3.82
45	13.01	11.12	9.23	7.58	5.92	4.91	3.89
46	13.25	11.33	9.41	7.72	6.04	5.00	3.97
47	13.50	11.54	9.59	7.87	6.15	5.10	4.05
48	13.74	11.75	9.77	8.02	6.27	5.19	4.12
49	13.98	11.96	9.94	8.16	6.38	5.29	4.20
50	14.22	12.17	10.12	8.31	6.50	5.39	4.27
51	14.47	12.38	10.30	8.45	6.61	5.48	4.35
52	14.71	12.59	10.47	8.60	6.73	5.58	4.43
53	14.95	12.80	10.65	8.75	6.85	5.67	4.50
54	15.20	13.01	10.83	8.89	6.96	5.77	4.58
55	15.44	13.22	11.00	9.04	7.08	5.87	4.66
56	15.68	13.43	11.18	9.19	7.19	5.96	4.73
57	15.93	13.64	11.36	9.33	7.31	6.06	4.81
58	16.17	13.85	11.53	9.48	7.42	6.15	4.89
59	16.41	14.06	11.71	9.62	7.54	6.25	4.96
60	16.65	14.27	11.89	9.77	7.66	6.35	5.04
61	16.90	14.48	12.06	9.92	7.77	6.44	5.11
62	17.14	14.69	12.24	10.06	7.89	6.54	5.19
63	17.38	14.90	12.42	10.21	8.00	6.64	5.27
64	17.63	15.11	12.59	10.36	8.12	6.73	5.34
65	17.87	15.32	12.77	10.50	8.23	6.83	5.42
66	18.11	15.53	12.95	10.65	8.35	6.92	5.50
67	18.36	15.74	13.12	10.79	8.47	7.02	5.57

Distance from Source (meters)	Stability Class 1	Stability Class 2	Stability Class 3	Stability Class 4	Stability Class 5	Stability Class 6	Stability Class 7
68	18.60	15.95	13.30	10.94	8.58	7.12	5.65
69	18.84	16.16	13.48	11.09	8.70	7.21	5.73
70	19.08	16.37	13.65	11.23	8.81	7.31	5.80
71	19.33	16.58	13.83	11.38	8.93	7.40	5.88
72	19.57	16.79	14.01	11.53	9.04	7.50	5.96
73	19.81	17.00	14.19	11.67	9.16	7.60	6.03
74	20.06	17.21	14.36	11.82	9.28	7.69	6.11
75	20.30	17.42	14.54	11.96	9.39	7.79	6.18
76	20.54	17.63	14.72	12.11	9.51	7.88	6.26
77	20.79	17.84	14.89	12.26	9.62	7.98	6.34
78	21.03	18.05	15.07	12.40	9.74	8.08	6.41
79	21.27	18.26	15.25	12.55	9.85	8.17	6.49
80	21.51	18.47	15.42	12.70	9.97	8.27	6.57
81	21.76	18.68	15.60	12.84	10.09	8.36	6.64
82	22.00	18.89	15.78	12.99	10.20	8.46	6.72
83	22.24	19.10	15.95	13.13	10.32	8.56	6.80
84	22.49	19.31	16.13	13.28	10.43	8.65	6.87
85	22.73	19.52	16.31	13.43	10.55	8.75	6.95
86	22.97	19.73	16.48	13.57	10.66	8.84	7.02
87	23.22	19.94	16.66	13.72	10.78	8.94	7.10
88	23.46	20.15	16.84	13.87	10.90	9.04	7.18
89	23.70	20.36	17.01	14.01	11.01	9.13	7.25
90	23.94	20.57	17.19	14.16	11.13	9.23	7.33
91	24.19	20.78	17.37	14.30	11.24	9.32	7.41
92	24.43	20.99	17.54	14.45	11.36	9.42	7.48
93	24.67	21.20	17.72	14.60	11.47	9.52	7.56
94	24.92	21.41	17.90	14.74	11.59	9.61	7.64
95	25.16	21.62	18.07	14.89	11.71	9.71	7.71
96	25.40	21.83	18.25	15.04	11.82	9.80	7.79
97	25.65	22.04	18.43	15.18	11.94	9.90	7.87
98	25.89	22.25	18.61	15.33	12.05	10.00	7.94
99	26.13	22.46	18.78	15.47	12.17	10.09	8.02
100	26.37	22.67	18.96	15.62	12.28	10.19	8.09
101	26.62	22.88	19.14	15.77	12.40	10.28	8.17

Distance from Source (meters)	Stability Class 1	Stability Class 2	Stability Class 3	Stability Class 4	Stability Class 5	Stability Class 6	Stability Class 7
102	26.86	23.09	19.31	15.91	12.51	10.38	8.25
103	27.10	23.30	19.49	16.06	12.63	10.48	8.32
104	27.35	23.51	19.67	16.21	12.75	10.57	8.40
105	27.59	23.72	19.84	16.35	12.86	10.67	8.48
106	27.83	23.93	20.02	16.50	12.98	10.77	8.55
107	28.08	24.14	20.20	16.64	13.09	10.86	8.63
108	28.32	24.35	20.37	16.79	13.21	10.96	8.71
109	28.56	24.56	20.55	16.94	13.32	11.05	8.78
110	28.80	24.77	20.73	17.08	13.44	11.15	8.86
111	29.05	24.98	20.90	17.23	13.56	11.25	8.93
112	29.29	25.19	21.08	17.38	13.67	11.34	9.01
113	29.53	25.40	21.26	17.52	13.79	11.44	9.09
114	29.78	25.61	21.43	17.67	13.90	11.53	9.16
115	30.02	25.82	21.61	17.81	14.02	11.63	9.24
116	30.26	26.03	21.79	17.96	14.13	11.73	9.32
117	30.51	26.24	21.96	18.11	14.25	11.82	9.39
118	30.75	26.45	22.14	18.25	14.37	11.92	9.47
119	30.99	26.65	22.32	18.40	14.48	12.01	9.55
120	31.23	26.86	22.49	18.55	14.60	12.11	9.62
121	31.48	27.07	22.67	18.69	14.71	12.21	9.70
122	31.72	27.28	22.85	18.84	14.83	12.30	9.78
123	31.96	27.49	23.03	18.98	14.94	12.40	9.85
124	32.21	27.70	23.20	19.13	15.06	12.49	9.93
125	32.45	27.91	23.38	19.28	15.18	12.59	10.00
126	32.69	28.12	23.56	19.42	15.29	12.69	10.08
127	32.94	28.33	23.73	19.57	15.41	12.78	10.16
128	33.18	28.54	23.91	19.72	15.52	12.88	10.23
129	33.42	28.75	24.09	19.86	15.64	12.97	10.31
130	33.66	28.96	24.26	20.01	15.75	13.07	10.39
131	33.91	29.17	24.44	20.15	15.87	13.17	10.46
132	34.15	29.38	24.62	20.30	15.99	13.26	10.54
133	34.39	29.59	24.79	20.45	16.10	13.36	10.62
134	34.64	29.80	24.97	20.59	16.22	13.45	10.69
135	34.88	30.01	25.15	20.74	16.33	13.55	10.77

Distance from Source (meters)	Stability Class 1	Stability Class 2	Stability Class 3	Stability Class 4	Stability Class 5	Stability Class 6	Stability Class 7
136	35.12	30.22	25.32	20.89	16.45	13.65	10.84
137	35.37	30.43	25.50	21.03	16.56	13.74	10.92
138	35.61	30.64	25.68	21.18	16.68	13.84	11.00
139	35.85	30.85	25.85	21.32	16.80	13.93	11.07
140	36.09	31.06	26.03	21.47	16.91	14.03	11.15
141	36.34	31.27	26.21	21.62	17.03	14.13	11.23
142	36.58	31.48	26.38	21.76	17.14	14.22	11.30
143	36.82	31.69	26.56	21.91	17.26	14.32	11.38
144	37.07	31.90	26.74	22.06	17.37	14.42	11.46
145	37.31	32.11	26.91	22.20	17.49	14.51	11.53
146	37.55	32.32	27.09	22.35	17.61	14.61	11.61
147	37.80	32.53	27.27	22.49	17.72	14.70	11.69
148	38.04	32.74	27.45	22.64	17.84	14.80	11.76
149	38.28	32.95	27.62	22.79	17.95	14.90	11.84
150	38.52	33.16	27.80	22.93	18.07	14.99	11.91
151	38.77	33.37	27.98	23.08	18.18	15.09	11.99
152	39.01	33.58	28.15	23.23	18.30	15.18	12.07
153	39.25	33.79	28.33	23.37	18.42	15.28	12.14
154	39.50	34.00	28.51	23.52	18.53	15.38	12.22
155	39.74	34.21	28.68	23.66	18.65	15.47	12.30
156	39.98	34.42	28.86	23.81	18.76	15.57	12.37
157	40.23	34.63	29.04	23.96	18.88	15.66	12.45
158	40.47	34.84	29.21	24.10	18.99	15.76	12.53
159	40.71	35.05	29.39	24.25	19.11	15.86	12.60
160	40.95	35.26	29.57	24.40	19.23	15.95	12.68
161	41.20	35.47	29.74	24.54	19.34	16.05	12.75
162	41.44	35.68	29.92	24.69	19.46	16.14	12.83
163	41.68	35.89	30.10	24.83	19.57	16.24	12.91
164	41.93	36.10	30.27	24.98	19.69	16.34	12.98
165	42.17	36.31	30.45	25.13	19.80	16.43	13.06
166	42.41	36.52	30.63	25.27	19.92	16.53	13.14
167	42.66	36.73	30.80	25.42	20.04	16.62	13.21
168	42.90	36.94	30.98	25.57	20.15	16.72	13.29
169	43.14	37.15	31.16	25.71	20.27	16.82	13.37

Distance from Source (meters)	Stability Class 1	Stability Class 2	Stability Class 3	Stability Class 4	Stability Class 5	Stability Class 6	Stability Class 7
170	43.38	37.36	31.33	25.86	20.38	16.91	13.44
171	43.63	37.57	31.51	26.00	20.50	17.01	13.52
172	43.87	37.78	31.69	26.15	20.61	17.10	13.60
173	44.11	37.99	31.87	26.30	20.73	17.20	13.67
174	44.36	38.20	32.04	26.44	20.85	17.30	13.75
175	44.60	38.41	32.22	26.59	20.96	17.39	13.82
176	44.84	38.62	32.40	26.74	21.08	17.49	13.90
177	45.09	38.83	32.57	26.88	21.19	17.58	13.98
178	45.33	39.04	32.75	27.03	21.31	17.68	14.05
179	45.57	39.25	32.93	27.17	21.42	17.78	14.13
180	45.81	39.46	33.10	27.32	21.54	17.87	14.21
181	46.06	39.67	33.28	27.47	21.66	17.97	14.28
182	46.30	39.88	33.46	27.61	21.77	18.07	14.36
183	46.54	40.09	33.63	27.76	21.89	18.16	14.44
184	46.79	40.30	33.81	27.91	22.00	18.26	14.51
185	47.03	40.51	33.99	28.05	22.12	18.35	14.59
186	47.27	40.72	34.16	28.20	22.23	18.45	14.66
187	47.52	40.93	34.34	28.34	22.35	18.55	14.74
188	47.76	41.14	34.52	28.49	22.47	18.64	14.82
189	48.00	41.35	34.69	28.64	22.58	18.74	14.89
190	48.24	41.56	34.87	28.78	22.70	18.83	14.97
191	48.49	41.77	35.05	28.93	22.81	18.93	15.05
192	48.73	41.98	35.22	29.08	22.93	19.03	15.12
193	48.97	42.19	35.40	29.22	23.04	19.12	15.20
194	49.22	42.40	35.58	29.37	23.16	19.22	15.28
195	49.46	42.61	35.75	29.51	23.28	19.31	15.35
196	49.70	42.82	35.93	29.66	23.39	19.41	15.43
197	49.95	43.03	36.11	29.81	23.51	19.51	15.51
198	50.19	43.24	36.29	29.95	23.62	19.60	15.58
199	50.43	43.45	36.46	30.10	23.74	19.70	15.66
200	50.67	43.66	36.64	30.25	23.85	19.79	15.73

Table A-2. Point Source Dispersion Table for Determining Value of Sigma-Z (meters)

Distance from Source (meters)	Stability Class 1	Stability Class 2	Stability Class 3	Stability Class 4	Stability Class 5	Stability Class 6	Stability Class 7
1	0.04	0.35	0.65	0.60	0.55	0.53	0.50
2	0.19	0.47	0.75	0.68	0.62	0.58	0.54
3	0.33	0.59	0.85	0.77	0.69	0.64	0.58
4	0.48	0.71	0.95	0.85	0.76	0.69	0.62
5	0.62	0.83	1.04	0.94	0.83	0.75	0.66
6	0.77	0.96	1.14	1.02	0.89	0.80	0.70
7	0.91	1.08	1.24	1.10	0.96	0.85	0.75
8	1.06	1.20	1.34	1.19	1.03	0.91	0.79
9	1.20	1.32	1.44	1.27	1.10	0.96	0.83
10	1.34	1.44	1.54	1.35	1.17	1.02	0.87
11	1.49	1.56	1.64	1.44	1.24	1.07	0.91
12	1.63	1.69	1.74	1.52	1.31	1.13	0.95
13	1.78	1.81	1.84	1.61	1.37	1.18	0.99
14	1.92	1.93	1.94	1.69	1.44	1.24	1.03
15	2.07	2.05	2.04	1.77	1.51	1.29	1.07
16	2.21	2.17	2.14	1.86	1.58	1.35	1.11
17	2.35	2.30	2.24	1.94	1.65	1.40	1.15
18	2.50	2.42	2.34	2.03	1.72	1.46	1.20
19	2.64	2.54	2.43	2.11	1.78	1.51	1.24
20	2.79	2.66	2.53	2.19	1.85	1.56	1.28
21	2.93	2.78	2.63	2.28	1.92	1.62	1.32
22	3.08	2.90	2.73	2.36	1.99	1.67	1.36
23	3.22	3.03	2.83	2.44	2.06	1.73	1.40
24	3.37	3.15	2.93	2.53	2.13	1.78	1.44
25	3.51	3.27	3.03	2.61	2.19	1.84	1.48
26	3.65	3.39	3.13	2.70	2.26	1.89	1.52
27	3.80	3.51	3.23	2.78	2.33	1.95	1.56
28	3.94	3.64	3.33	2.86	2.40	2.00	1.60
29	4.09	3.76	3.43	2.95	2.47	2.06	1.65
30	4.23	3.88	3.53	3.03	2.54	2.11	1.69
31	4.38	4.00	3.63	3.12	2.60	2.17	1.73
32	4.52	4.12	3.73	3.20	2.67	2.22	1.77
33	4.67	4.24	3.82	3.28	2.74	2.28	1.81

Distance from Source (meters)	Stability Class 1	Stability Class 2	Stability Class 3	Stability Class 4	Stability Class 5	Stability Class 6	Stability Class 7
34	4.81	4.37	3.92	3.37	2.81	2.33	1.85
35	4.95	4.49	4.02	3.45	2.88	2.38	1.89
36	5.10	4.61	4.12	3.53	2.95	2.44	1.93
37	5.24	4.73	4.22	3.62	3.02	2.49	1.97
38	5.39	4.85	4.32	3.70	3.08	2.55	2.01
39	5.53	4.98	4.42	3.79	3.15	2.60	2.05
40	5.68	5.10	4.52	3.87	3.22	2.66	2.10
41	5.82	5.22	4.62	3.95	3.29	2.71	2.14
42	5.96	5.34	4.72	4.04	3.36	2.77	2.18
43	6.11	5.46	4.82	4.12	3.43	2.82	2.22
44	6.25	5.59	4.92	4.21	3.49	2.88	2.26
45	6.40	5.71	5.02	4.29	3.56	2.93	2.30
46	6.54	5.83	5.12	4.37	3.63	2.99	2.34
47	6.69	5.95	5.22	4.46	3.70	3.04	2.38
48	6.83	6.07	5.31	4.54	3.77	3.10	2.42
49	6.98	6.19	5.41	4.62	3.84	3.15	2.46
50	7.12	6.32	5.51	4.71	3.90	3.20	2.50
51	7.26	6.44	5.61	4.79	3.97	3.26	2.55
52	7.41	6.56	5.71	4.88	4.04	3.31	2.59
53	7.55	6.68	5.81	4.96	4.11	3.37	2.63
54	7.70	6.80	5.91	5.04	4.18	3.42	2.67
55	7.84	6.93	6.01	5.13	4.25	3.48	2.71
56	7.99	7.05	6.11	5.21	4.31	3.53	2.75
57	8.13	7.17	6.21	5.30	4.38	3.59	2.79
58	8.28	7.29	6.31	5.38	4.45	3.64	2.83
59	8.42	7.41	6.41	5.46	4.52	3.70	2.87
60	8.56	7.53	6.51	5.55	4.59	3.75	2.91
61	8.71	7.66	6.61	5.63	4.66	3.81	2.95
62	8.85	7.78	6.70	5.71	4.73	3.86	3.00
63	9.00	7.90	6.80	5.80	4.79	3.91	3.04
64	9.14	8.02	6.90	5.88	4.86	3.97	3.08
65	9.29	8.14	7.00	5.97	4.93	4.02	3.12
66	9.43	8.27	7.10	6.05	5.00	4.08	3.16
67	9.57	8.39	7.20	6.13	5.07	4.13	3.20

Distance from Source (meters)	Stability Class 1	Stability Class 2	Stability Class 3	Stability Class 4	Stability Class 5	Stability Class 6	Stability Class 7
68	9.72	8.51	7.30	6.22	5.14	4.19	3.24
69	9.86	8.63	7.40	6.30	5.20	4.24	3.28
70	10.01	8.75	7.50	6.39	5.27	4.30	3.32
71	10.15	8.88	7.60	6.47	5.34	4.35	3.36
72	10.30	9.00	7.70	6.55	5.41	4.41	3.40
73	10.44	9.12	7.80	6.64	5.48	4.46	3.45
74	10.59	9.24	7.90	6.72	5.55	4.52	3.49
75	10.73	9.36	8.00	6.80	5.61	4.57	3.53
76	10.87	9.48	8.09	6.89	5.68	4.63	3.57
77	11.02	9.61	8.19	6.97	5.75	4.68	3.61
78	11.16	9.73	8.29	7.06	5.82	4.73	3.65
79	11.31	9.85	8.39	7.14	5.89	4.79	3.69
80	11.45	9.97	8.49	7.22	5.96	4.84	3.73
81	11.60	10.09	8.59	7.31	6.02	4.90	3.77
82	11.74	10.22	8.69	7.39	6.09	4.95	3.81
83	11.89	10.34	8.79	7.48	6.16	5.01	3.85
84	12.03	10.46	8.89	7.56	6.23	5.06	3.89
85	12.17	10.58	8.99	7.64	6.30	5.12	3.94
86	12.32	10.70	9.09	7.73	6.37	5.17	3.98
87	12.46	10.82	9.19	7.81	6.44	5.23	4.02
88	12.61	10.95	9.29	7.89	6.50	5.28	4.06
89	12.75	11.07	9.39	7.98	6.57	5.34	4.10
90	12.90	11.19	9.48	8.06	6.64	5.39	4.14
91	13.04	11.31	9.58	8.15	6.71	5.45	4.18
92	13.18	11.43	9.68	8.23	6.78	5.50	4.22
93	13.33	11.56	9.78	8.31	6.85	5.55	4.26
94	13.47	11.68	9.88	8.40	6.91	5.61	4.30
95	13.62	11.80	9.98	8.48	6.98	5.66	4.34
96	13.76	11.92	10.08	8.57	7.05	5.72	4.39
97	13.91	12.04	10.18	8.65	7.12	5.77	4.43
98	14.05	12.17	10.28	8.73	7.19	5.83	4.47
99	14.20	12.29	10.38	8.82	7.26	5.88	4.51
100	14.34	12.41	10.48	8.90	7.32	5.94	4.55
101	14.48	12.53	10.58	8.99	7.39	5.99	4.59

Distance from Source (meters)	Stability Class 1	Stability Class 2	Stability Class 3	Stability Class 4	Stability Class 5	Stability Class 6	Stability Class 7
102	14.63	12.65	10.68	9.07	7.46	6.05	4.63
103	14.77	12.77	10.78	9.15	7.53	6.10	4.67
104	14.92	12.90	10.88	9.24	7.60	6.16	4.71
105	15.06	13.02	10.97	9.32	7.67	6.21	4.75
106	15.21	13.14	11.07	9.40	7.73	6.26	4.79
107	15.35	13.26	11.17	9.49	7.80	6.32	4.84
108	15.50	13.38	11.27	9.57	7.87	6.37	4.88
109	15.64	13.51	11.37	9.66	7.94	6.43	4.92
110	15.78	13.63	11.47	9.74	8.01	6.48	4.96
111	15.93	13.75	11.57	9.82	8.08	6.54	5.00
112	16.07	13.87	11.67	9.91	8.15	6.59	5.04
113	16.22	13.99	11.77	9.99	8.21	6.65	5.08
114	16.36	14.11	11.87	10.08	8.28	6.70	5.12
115	16.51	14.24	11.97	10.16	8.35	6.76	5.16
116	16.65	14.36	12.07	10.24	8.42	6.81	5.20
117	16.79	14.48	12.17	10.33	8.49	6.87	5.24
118	16.94	14.60	12.27	10.41	8.56	6.92	5.29
119	17.08	14.72	12.36	10.49	8.62	6.98	5.33
120	17.23	14.85	12.46	10.58	8.69	7.03	5.37
121	17.37	14.97	12.56	10.66	8.76	7.08	5.41
122	17.52	15.09	12.66	10.75	8.83	7.14	5.45
123	17.66	15.21	12.76	10.83	8.90	7.19	5.49
124	17.81	15.33	12.86	10.91	8.97	7.25	5.53
125	17.95	15.46	12.96	11.00	9.03	7.30	5.57
126	18.09	15.58	13.06	11.08	9.10	7.36	5.61
127	18.24	15.70	13.16	11.17	9.17	7.41	5.65
128	18.38	15.82	13.26	11.25	9.24	7.47	5.69
129	18.53	15.94	13.36	11.33	9.31	7.52	5.74
130	18.67	16.06	13.46	11.42	9.38	7.58	5.78
131	18.82	16.19	13.56	11.50	9.44	7.63	5.82
132	18.96	16.31	13.66	11.58	9.51	7.69	5.86
133	19.11	16.43	13.75	11.67	9.58	7.74	5.90
134	19.25	16.55	13.85	11.75	9.65	7.79	5.94
135	19.39	16.67	13.95	11.84	9.72	7.85	5.98

Distance from Source (meters)	Stability Class 1	Stability Class 2	Stability Class 3	Stability Class 4	Stability Class 5	Stability Class 6	Stability Class 7
136	19.54	16.80	14.05	11.92	9.79	7.90	6.02
137	19.68	16.92	14.15	12.00	9.86	7.96	6.06
138	19.83	17.04	14.25	12.09	9.92	8.01	6.10
139	19.97	17.16	14.35	12.17	9.99	8.07	6.14
140	20.12	17.28	14.45	12.26	10.06	8.12	6.19
141	20.26	17.40	14.55	12.34	10.13	8.18	6.23
142	20.40	17.53	14.65	12.42	10.20	8.23	6.27
143	20.55	17.65	14.75	12.51	10.27	8.29	6.31
144	20.69	17.77	14.85	12.59	10.33	8.34	6.35
145	20.84	17.89	14.95	12.67	10.40	8.40	6.39
146	20.98	18.01	15.05	12.76	10.47	8.45	6.43
147	21.13	18.14	15.15	12.84	10.54	8.51	6.47
148	21.27	18.26	15.24	12.93	10.61	8.56	6.51
149	21.42	18.38	15.34	13.01	10.68	8.61	6.55
150	21.56	18.50	15.44	13.09	10.74	8.67	6.59
151	21.70	18.62	15.54	13.18	10.81	8.72	6.64
152	21.85	18.75	15.64	13.26	10.88	8.78	6.68
153	21.99	18.87	15.74	13.35	10.95	8.83	6.72
154	22.14	18.99	15.84	13.43	11.02	8.89	6.76
155	22.28	19.11	15.94	13.51	11.09	8.94	6.80
156	22.43	19.23	16.04	13.60	11.15	9.00	6.84
157	22.57	19.35	16.14	13.68	11.22	9.05	6.88
158	22.72	19.48	16.24	13.76	11.29	9.11	6.92
159	22.86	19.60	16.34	13.85	11.36	9.16	6.96
160	23.00	19.72	16.44	13.93	11.43	9.22	7.00
161	23.15	19.84	16.54	14.02	11.50	9.27	7.04
162	23.29	19.96	16.63	14.10	11.57	9.33	7.09
163	23.44	20.09	16.73	14.18	11.63	9.38	7.13
164	23.58	20.21	16.83	14.27	11.70	9.43	7.17
165	23.73	20.33	16.93	14.35	11.77	9.49	7.21
166	23.87	20.45	17.03	14.44	11.84	9.54	7.25
167	24.01	20.57	17.13	14.52	11.91	9.60	7.29
168	24.16	20.69	17.23	14.60	11.98	9.65	7.33
169	24.30	20.82	17.33	14.69	12.04	9.71	7.37

Distance from Source (meters)	Stability Class 1	Stability Class 2	Stability Class 3	Stability Class 4	Stability Class 5	Stability Class 6	Stability Class 7
170	24.45	20.94	17.43	14.77	12.11	9.76	7.41
171	24.59	21.06	17.53	14.85	12.18	9.82	7.45
172	24.74	21.18	17.63	14.94	12.25	9.87	7.49
173	24.88	21.30	17.73	15.02	12.32	9.93	7.54
174	25.03	21.43	17.83	15.11	12.39	9.98	7.58
175	25.17	21.55	17.93	15.19	12.45	10.04	7.62
176	25.31	21.67	18.02	15.27	12.52	10.09	7.66
177	25.46	21.79	18.12	15.36	12.59	10.14	7.70
178	25.60	21.91	18.22	15.44	12.66	10.20	7.74
179	25.75	22.04	18.32	15.53	12.73	10.25	7.78
180	25.89	22.16	18.42	15.61	12.80	10.31	7.82
181	26.04	22.28	18.52	15.69	12.86	10.36	7.86
182	26.18	22.40	18.62	15.78	12.93	10.42	7.90
183	26.33	22.52	18.72	15.86	13.00	10.47	7.94
184	26.47	22.64	18.82	15.94	13.07	10.53	7.98
185	26.61	22.77	18.92	16.03	13.14	10.58	8.03
186	26.76	22.89	19.02	16.11	13.21	10.64	8.07
187	26.90	23.01	19.12	16.20	13.28	10.69	8.11
188	27.05	23.13	19.22	16.28	13.34	10.75	8.15
189	27.19	23.25	19.32	16.36	13.41	10.80	8.19
190	27.34	23.38	19.41	16.45	13.48	10.86	8.23
191	27.48	23.50	19.51	16.53	13.55	10.91	8.27
192	27.62	23.62	19.61	16.62	13.62	10.96	8.31
193	27.77	23.74	19.71	16.70	13.69	11.02	8.35
194	27.91	23.86	19.81	16.78	13.75	11.07	8.39
195	28.06	23.98	19.91	16.87	13.82	11.13	8.43
196	28.20	24.11	20.01	16.95	13.89	11.18	8.48
197	28.35	24.23	20.11	17.03	13.96	11.24	8.52
198	28.49	24.35	20.21	17.12	14.03	11.29	8.56
199	28.64	24.47	20.31	17.20	14.10	11.35	8.60
200	28.78	24.59	20.41	17.29	14.16	11.40	8.64

MATLAB®-based OTM33_PSG_031913.m – Discontinued

```
clear

% This MATLAB code is used to read CSV raw data files and analyze the data to
yield methane emission rate estimates
load pgtabley.mat; % Load the Sigma y table
load pgtablez.mat; % Load the Sigma z table
files = dir( 'STR*.csv'); % Pointing to all the STR files in the directory
countfiles = numel(files);
param = xlsread('parameter.xls'); % Reading the Parameter file containing the
relevant parameters to perform the calculation

run = 050113;   % Analysis run MMDDYY(see notbook)
wslimit = 0.0;  % Set wind speed cut limit (m/s)
%al = 60;      % Set wind angle cut limit (+/- deg, 180 indicates no filter
al = param(1,1); % This is the wind cut
clp = 2.0;     % Set bin density cut limit (%)
bb1 =5;         % Set start deg value for angle bin (deg)
bb2 =10 ;       % Set bin size (deg)
bb3 = 365;      % Set end bin (see Ln 12)

dist = [];
senh = [];
sorh = [];

%%%%  This section reads the STR files in csv format which follows the Retigo
% format and extracts parameters relevant to the calculation

formatdatetime = 'yyyy-mm-dd HH:MM:SS'; % Specifying the format of date &
time in the STR file

for k1 = 1:countfiles
    jadu = fopen((files(k1).name));

ach = textscan(jadu,'%s %f %f %n %f %f %f %f %n %f %f %f %f %f %f %f %n %f %f
%f %f %f %f %f %f %f %f %n %f %f %n %f %n %f %f %n %n %n %n %f %f ',...
    'Delimiter',',','CollectOutput',1,'HeaderLines',9); % Specifiying the
format type of the given columns
fclose(jadu)
format short g
newdat = (ach{1}(1:end,:));
newdat1 = char(newdat);

% Since the format of date and time is very different from what MATLAB
% recognizes, this following code enables MATLAB to read the date and time
% properly from the STR file
for si = 1:length(newdat1)

jaguar (si) = strfind(newdat1(si,:),'T');
```

```
end

 for si = 1:length(newdat1)
 jaguar1(si) = strfind(newdat1(si,:),'+');
 end

 newdat1(:,[jaguar,jaguar1:end]) = ' ';
 newdat2 = datenum(newdat1,formatdatetime);


jadu1 = [(newdat2) ach{2}];


% This part of the code is extracting distance values, sensor height, and
% source height from the STR file
jaduguy = fopen((files(k1).name));
dis = textscan(jaduguy,'%s %f',...
    'Delimiter',',','CollectOutput',1);
fclose(jaduguy)
jaduguy1 = (dis{2}(1:end,:));
dista = jaduguy1(1,1);
dist = [dist;dista]; % Tabulating distance values of all the measurements
senht = jaduguy1(101,1);
senh = [senh;senht]; % Tabulating sensor height values of all the
measurements if provided
sorht = jaduguy1(121,1);
sorh = [sorh;sorht]; % Tabulating source height values of all the
measurements if provided

% Taking care of when the weather measuring instrument does not log in
% data which may happen in the field
[r2 c2] = size (jadu1);
  for y1 = 1:r2


      nanchk = isnan (jadu1((1:end),22));
      [y1 n2] = find( nanchk == 1);


     end

  jadu1(y1(1:end),:) = [];

delay = param(2,1); % Delay time
samprate = param(5,1); % Sampling Rate
n1 = delay.*samprate;
```

```
time = jadu1((1:end),1); % Date & Time


lat = jadu1((1:end),2);    % GPS latitude
long = jadu1((1:end),3);   % GPS longitude
ws2 = jadu1((1:end),11);    % 2D wind speed (m/s)
wd2 = jadu1((1:end),12);    % 2D wind direction (deg, auto aligned north)
temp = jadu1((1:end),13);  % Met station Temperature (deg C)
pres = jadu1((1:end),15);  % Met Station Pressure (mbar)
ws3y = jadu1((1:end),18);  % 3D u  (m/s)
ws3x = jadu1((1:end),19);  % 3D v  (m/s)
ws3z = jadu1((1:end),20);  % 3D w  (m/s)
ws3 = jadu1((1:end),22);    % 3D sonic 3D wind speed (m/s)
wd3 = jadu1((1:end),23);    % 3D wind direction (deg, arbitrary)
ws3t = jadu1((1:end),26);  % 3D t  (deg C)
co2 = jadu1((1:end),29);    % CO2 concentration (ppm)
ch4 = jadu1((1:end),30);    % Ch4 concentration (ppm)

 co2(1:n1) = [];  % Accomodating all other variables for the delay time
  ch4(1:n1) = [];
  c1 = length(co2);
  time(c1+1:end) = [];
  ws3(c1+1:end) = [];
  wd3(c1+1:end) = [];
  ws2(c1+1:end) = [];
  wd2(c1+1:end) = [];
  temp(c1+1:end) = [];
  pres(c1+1:end) = [];
  ws3z(c1+1:end) = [];
  ws3y(c1+1:end) = [];
  ws3x(c1+1:end) = [];
  ws3t(c1+1:end) = [];
  lat(c1+1:end) = [];
  long(c1+1:end) = [];



  calib = param(3,1);
  ch4=ch4./calib; % calibration correction

  % Rotate 3D sonic coordinate system to streamlined set (-180 deg)
   % Field data should be near -180 deg by manual sonic rotation set
   % If mean direction is in N hemispehre, rotation sets to 0 deg (warning)
   % Output <x>, <z>, = 0 or check file
    mws3x = mean(ws3x);
    mws3y = mean(ws3y);
```

```
    mws3z = mean(ws3z);

    mdir = 180 + (atan2(-1*(mws3y),-1*(mws3x))*57.29578);

    RA = atan(mws3y/mws3x); % First rotation (set <x> = 0)

    CRA = cos(RA);
    SRA = sin(RA);

    nws3x = ws3x*CRA + ws3y*SRA;
    nws3y = (-1)*ws3x*SRA + ws3y*CRA;

    mnws3x = mean(nws3x);
    mnws3y = mean(nws3y);
    mndir = 180 + (atan2(-1*(mnws3y),-1*(mnws3x))*57.29578);

    RB = atan(mws3z/mnws3x); %Second rotation (set <z> = 0)

    CRB = cos(RB);
    SRB = sin(RB);

    nnws3x = nws3x*CRB + ws3z*SRB;
    nws3z = (-1)*nws3x*SRB + ws3z*CRB;

    mnnws3x = mean(nnws3x);
    mnws3z = mean(nws3z);
    mmndir = 180 + (atan2(-1*(mnws3z),-1*(mnnws3x))*57.29578);

   % assigned streamlined coordates
    ws3x = nnws3x;
    ws3y = nws3y;
    ws3z = nws3z;
   % calculate and asign new 3D sonic 2D wind direction
    ndir = 180 + (atan2(-1*(ws3y),-1*(ws3x))*57.29578);
    wd3 = ndir;
   % calculate and asign new 3D sonic 2D wind speed
    sp = (ws3x.*ws3x + ws3y.*ws3y);
    sp1 = sp.^0.5;
    ws3 =sp1;

jadu2 = [time lat long co2 ch4 ws2 wd2 temp pres ws3 wd3 ws3y ws3x ws3z
ws3t]; % write the key arrays onto a new file



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Wind speed filter
   [r c] = size(jadu2);
   a2 = find(ws3<wslimit);
   jadu2(a2,:) = [];
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Wind angle filter
wd3j = jadu2((1:end),11); % 3D sonic wind direction
cl=clp/100;
cutM = cl* length(wd3j); % Standard is 1% cut

degM = 1:10:361;
i2 = 1:length(degM);
i1 = 1:length(wd3j);
resM(i2) = 0;
countM(i2) = 0;
 for i2 = 1:length(degM)
    for i1 = 1:length(wd3j)
        if wd3j(i1) >= degM(i2) && wd3j(i1) < degM(i2+1)
            countM(i2)= countM(i2) +1 ;
            resM(i2)= resM(i2)+ ch4(i1) ;
        end
    end
 end

 for i2 = 1:length(degM)
    if countM(i2) >= cutM
          resMN(i2) = resM(i2)./countM(i2);
  elseif countM(i2) < cutM
        resMN(i2) = min(resM);
     end
  end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% resNM = resM./countM;
% [C,binx] = max(resNM);
 [C,binx] = max(resMN);

binxx = 10*binx - 5;

degl = binxx - al;
degh = binxx + al;
    [r c] = size(jadu2);
    a3 = find(wd3j<degl);
    a4 = find(wd3j>degh);
    a5 = cat (1,a3,a4);
    a6 = sort (a5);
    jadu2(a6,:) = [];

    csvwrite(['Processed',files(k1).name], jadu2); % Save Time Aligned files
as Processed.csv in the same directory
end

files2 = dir('Processed*.csv');

% Variables which keep recording new values as the code goes through each
% files
```

```
countfiles2 = numel(files2);
impnum = [];
emflux2 = [];
disney0 = [];
coeffi = [];
coeffi1 = [];
rsq = [];
cut0 = [];
numabcknd = [];
pnum = [];
rt2rp2 = [];
methanemean = [];
binqa = [];
cangl1 = [];
countqa = [];
tempqa = [];
stdws3tqa = [];
minws3tqa = [];
maxws3tqa = [];
a1bkg1 = [];




aaa = dist./dist;
wsl = wslimit*aaa;
all = al*aaa;
clpl =clp*aaa;
bin = bb2*aaa;

rt1rp1 = 298/1013.25;
gasconst = 8.314510;
mw = param(4,1); % Molecular weight
grt1rp1 = rt1rp1*gasconst;
rt0rp0 = (gasconst*298)/101.325;
opt = (1e-06 * mw*1000)/rt0rp0; % Converting the concentration to grams per
meter cube

for k2 = 1:countfiles2

a1 = csvread(files2(k2).name);
ws = a1(:,10);            % 3D soinic 3D wind speed (m/s)
ws2 = a1(:,6);           % Met. Station 2D wind wpeed (m/s)
temp = a1(:,8)+273.15;     % Met. Station temp (deg K)
pres = a1(:,9);          % Met. Station atmospheric pressure (mbar)
a1a = a1(:,11);           % 3D wind direction
a1b = a1(:,5);           % Methane concentration
a1c = a1(:,4);           % CO2 concentration
ws3z = a1(:,14);          % 3D w
ws3y = a1(:,12);          % 3D u
```

```
ws3x = a1(:,13);          % 3D v
ws3t = a1(:,15);          % 3D t
maxmet = max(a1b);         % Maximum methane concentration
minmet = min(a1b);         % Minimum methane concentration
mmet = mean(a1b);          % Mean methane concentration
mws = mean(ws);           % Mean 3D sonic wind speed (m/s)
mws2 = mean(ws2);          % Mean met. Station 2D wind speed (m/s)
mtemp = mean(temp);        % Mean met station temp. (K)
mpres = mean(pres);        % Mean met. station atmospheric pressure (mbar)
m3Dt = mean(ws3t)+273.15;   % Mean 3D sonic temp. (K)
mco2 = mean(a1c);          % Mean CO2 concentration (ppm)
stdco2 = std(a1c);         % Std dev in CO2 (ppm)
stdws3t = std(ws3t);       % Std dev in 3D sonic t (C)
minws3t = min(ws3t);       % Minimum in 3D sonic t (C)
maxws3t = max(ws3t);       % Maximum in 3D sonic t (C)
lat = a1(:,2);            % Latitude
long = a1(:,3);           % Longitude
lat1 = mean(lat(:,1));     % Mean Latitude
long1 = mean(long(:,1));    % Mean Longitude
latstd = std(lat);         % Standard Deviation of the Latitude
longstd = std(long);       % Standeard Deviation of the Longitude

% This section specifies how to name the figure files containing the
% Gaussian fit with the relevant parameters
disney = files2(k2).name;
disney1 = isletter(disney);
disney2 = find(disney1 == 0);
disney3 = disney(disney2);
disney4 = str2num(disney3(2:8));
disney5 = str2num(disney3(10:end));
disney0 = [disney0; disney4 disney5];

% Taking the average of the lowest methane concentration values
og = length(a1b);
oga = og*0.05;
lowval = sort(a1b);
 lowmet = mean(lowval(1:oga)); %lowest 5% values
%lowmet = mean(lowval(1:100)); %lowest 100 values
    methanemean = [methanemean;lowmet];
    mmmet = mmet - lowmet;
  lowmetstd = std(lowval(1:oga));
 hival = sort(a1b,'descend');
  himet = mean(hival(1:oga));
  himetstd = std(hival(1:oga));

  if lowmetstd >= 0.005;
       lflag = 1;
   else
       lflag = 0;
  end
  if maxmet >= 2.5*himet;
```

```
        hflag = 1;
    else
        hflag = 0;
  end

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%calculating raw mean products for use in Windtrax
       my = mean(ws3y);
       mx = mean(ws3x);
       mz = mean(ws3z);
       mt = mean(ws3t);
       y2 = ws3y.*ws3y;
       my2 = mean(y2);
       x2 = ws3x.*ws3x;
       mx2 = mean(x2);
       z2 = ws3z.*ws3z;
       mz2 = mean(z2);
       t2 = ws3t.*ws3t;
       mt2 = mean(t2);
       yx = ws3y.*ws3x;
       myx =  mean(yx);
       yz = ws3y.*ws3z;
       myz =  mean(yz);
       yt = ws3y.*ws3t;
       myt =  mean(yt);
       xz = ws3x.*ws3z;
       mxz =  mean(xz);
       xt = ws3x.*ws3t;
       mxt =  mean(xt);
       zt = ws3z.*ws3t;
       mzt =  mean(zt);

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%calculating wind parameters

%turbulent intensity
    sws3z = std(ws3z);
    turbuint = sws3z/mws;

% Average and the standard deviation of wind 3D sonic azmuth wind direction

wd3mean = 180 + (atan2(-1*(my),-1*(mx))*57.29578);
stddev3 = std(a1a);



% Average and the standard deviation of wind direction for 2D met station
wd2rad = a1(:,7) * pi/180;
wd2y = sin(wd2rad);
wd2x = cos(wd2rad);
wd2num = numel(wd2rad);
wd2xmean = mean(wd2x);
```

```
wd2ymean = mean(wd2y);

if wd2xmean > 0 && wd2ymean > 0
wd2mean = atan(wd2ymean/wd2xmean);
wd2mean = wd2mean * (180/pi);

ep1 = wd2xmean.^2 + wd2ymean.^2;
ep1 = sqrt(1-ep1);
stddev1 = asin(ep1);
stddev2 = 1+(1.1547 -1)*ep1.^3;
stddev = stddev1 * stddev2;
stddev = stddev * (180/pi);

elseif wd2xmean > 0 && wd2ymean < 0
   wd2mean = atan(wd2ymean/wd2xmean);
wd2mean = wd2mean * (180/pi);
wd2mean = wd2mean + 360;

ep1 = wd2xmean.^2 + wd2ymean.^2;
ep1 = sqrt(1-ep1);
stddev1 = asin(ep1);
stddev2 = 1+(1.1547 -1)*ep1.^3;
stddev = stddev1 * stddev2;
stddev = stddev * (180/pi);

elseif wd2xmean < 0
   wd2mean = atan(wd2ymean/wd2xmean);
wd2mean = wd2mean * (180/pi);
wd2mean = wd2mean + 180;

ep1 = wd2xmean.^2 + wd2ymean.^2;
ep1 = sqrt(1-ep1);
stddev1 = asin(ep1);
stddev2 = 1+(1.1547 -1)*ep1.^3;
stddev = stddev1 * stddev2;
stddev = stddev * (180/pi);

end

%%%%%%%%%%%%%%%%%% Section estimates sigmay and sigmaz estimate based on
stability class estimate

%%% Wind Speed PG Indicator (3D)
if mws > 0 && mws < 1.700;
   pgwsi = 1;
elseif mws >= 1.700 && mws < 2.400;
   pgwsi = 2;
elseif mws >= 2.400 && mws < 3.200;
   pgwsi = 3;
elseif mws >= 3.200 && mws < 4.100;
```

```
    pgwsi = 4;
elseif mws >= 4.100 && mws < 5.100;
    pgwsi = 5;
elseif mws >= 5.100 && mws < 6.200;
    pgwsi = 6;
elseif mws >= 6.200;
    pgwsi = 7;
end

%%% Wind Speed PG Indicator (2D)
if mws2 > 0 && mws2 < 1.700;
    pgws2i = 1;
elseif mws2 >= 1.700 && mws2 < 2.400;
    pgws2i = 2;
elseif mws2 >= 2.400 && mws2 < 3.200;
    pgws2i = 3;
elseif mws2 >= 3.200 && mws2 < 4.100;
    pgws2i = 4;
elseif mws2 >= 4.100 && mws2 < 5.100;
    pgws2i = 5;
elseif mws2 >= 5.100 && mws2 < 6.200;
    pgws2i = 6;
elseif mws2 >= 6.200;
    pgws2i = 7;
end

%%% Stdev Wind Direction PG Indicator (3D)
if stddev3 > 27.50;
    pgstddev3i = 1.00;
elseif stddev3 <= 27.50 && stddev3 > 23.50;
    pgstddev3i = 2.00;
elseif stddev3 <= 23.50 && stddev3 > 19.50;
    pgstddev3i = 3.00;
elseif stddev3 <= 19.50 && stddev3 > 15.50;
    pgstddev3i = 4.00;
elseif stddev3 <= 15.50 & stddev3 > 11.50;
    pgstddev3i = 5.00;
elseif stddev3 <= 11.50 & stddev3 > 7.50;
    pgstddev3i = 6.00;
elseif stddev3 <= 7.50
    pgstddev3i = 7.00;
end

%%% Stdev Wind Direction PG Indicator (2D)
if stddev > 27.50;
    pgstddevi = 1.00;
elseif stddev <= 27.50 && stddev > 23.50;
    pgstddevi = 2.00;
elseif stddev <= 23.50 && stddev > 19.50;
    pgstddevi = 3.00;
elseif stddev <= 19.50 && stddev > 15.50;
```

```
    pgstddevi = 4.00;
elseif stddev <= 15.50 && stddev > 11.50;
    pgstddevi = 5.00;
elseif stddev <= 11.50 && stddev > 7.50;
    pgstddevi = 6.00;
elseif stddev <= 7.50
    pgstddevi = 7.00;
end

%%% Turbulent Intensity PG Indicator (3D)
if turbuint > 0.205;
    pgturbi = 1.00;
elseif turbuint <= 0.205 && turbuint > 0.180;
    pgturbi = 2.00;
elseif turbuint <= 0.180 && turbuint > 0.155;
    pgturbi = 3.00;
elseif turbuint <= 0.155 && turbuint > 0.130;
    pgturbi = 4.00;
elseif turbuint <= 0.130 && turbuint > 0.105;
    pgturbi = 5.00;
elseif turbuint <= 0.105 && turbuint > 0.080;
    pgturbi = 6.00;
elseif turbuint <= 0.080;
    pgturbi = 7.00;
end

%%%%this line calculates average pg indicator

pgi = [pgstddevi + pgturbi]/2;


%%% Selection of the Stability Class
if pgi < 1.500;
    pgt = 1.00;
elseif pgi >= 1.500 && pgi < 2.500;
    pgt = 2.00;
elseif pgi >= 2.500 && pgi < 3.500;
    pgt = 3.00;
elseif pgi >= 3.500 && pgi < 4.500;
    pgt = 4.00;
elseif pgi >= 4.500 && pgi < 5.500;
    pgt = 5.00;
elseif pgi >= 5.500 && pgi < 6.500;
    pgt = 6.00;
elseif pgi >= 6.500
    pgt = 7.00;
end


d = dist(k2);
```

```
%d = dist;
dd = round(d);
pgty = pgtabley(dd,pgt); % Specifying the plume width in the y direction
pgtz = pgtablez(dd,pgt); % Plume width in the z direction

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end of pg indicator section
% Saving values
impnum1= [mpres my mx mz mt my2 mx2 mz2 mt2 myx myz myt mxz mxt mzt lat1
latstd long1 longstd mco2 stdco2 oga maxmet himet himetstd minmet lowmet
lowmetstd mmet mmmet mws mtemp wd2mean wd3mean stddev stddev3 turbuint pgws2i
pgwsi pgstddevi pgstddev3i pgturbi pgi pgt pgty pgtz];
impnum = [impnum; impnum1];
csvwrite(['ImpValues_',files2(k2).name], impnum1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Binning measured concetration in in given degree increments

deg = bb1:bb2:bb3;  %bin configuration
i2 = 1:length(deg);
i1 = 1:length(a1a);
Mini = min(a1a);
Maxi = max(a1a);

% Set cut filter to remove low denity bins
% cl=clp/100;
% cut = cl* length(a1a); % Standard is 1% cut
cut = cutM;
    cut2 = cut/10;
    cut0 = [cut0;cut2];

% Subtracting off the background which is the avg of the lowest 5% values
for i1 = 1:length(a1b)
   if a1b(i1) ~= 0
       a1b(i1) = a1b(i1)-lowmet;
   else
      a1b(i1) = a1b(i1);
   end
end

% Bin concetration values
res(i2) = 0;
count(i2) = 0;
 for i2 = 1:length(deg)
    for i1 = 1:length(a1a)
       if a1a(i1) >= deg(i2) && a1a(i1) < deg(i2+1)
       count(i2)= count(i2) +1 ;
       res(i2)= res(i2)+ a1b(i1) ;
         end
     end
 end
```

```
% Accounts for case when the count is equal to zero or less
% the number of points given in the cut filter before taking the average
  for i2 = 1:length(deg)
    if count(i2) >= cut
          resN(i2) = res(i2)./count(i2);
  elseif count(i2) < cut
        resN(i2) = min(res);
     end
  end
 [C,bqa] = max(resN);
 cangl = 10*bqa - 5;
 cqa = max(count);
 tqa = mtemp/m3Dt;
    binqa =[binqa;bqa];
    cangl1 =[cangl1;cangl];
    countqa =[countqa;cqa];
    tempqa =[tempqa;tqa];
    stdws3tqa = [stdws3tqa;stdws3t];
    minws3tqa = [minws3tqa;minws3t];
    maxws3tqa = [maxws3tqa;maxws3t];

% Centers data so it is located at 180 degree
 degcent = 180;     % center of the binning
 degmax = 360;      % maximum of the binning
 degmin = 0;        % minimum of the binning
 resNmax = find(resN == max(resN));
 resNbtwn = find(resN >= min(resN) & resN <= max(resN));
 x = (degcent - deg(resNmax));
 deg = deg + x;
 for i1 = 1:length(deg)
     if deg(i1) > degmax
         deg(i1) = deg(i1)-degmax;
  elseif deg(i1) < degmax && deg(i1) >= degmin
        deg(i1) = deg(i1);
  elseif deg(i1) < degmin
       deg(i1) = deg(i1) + degmax;
     end
 end

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%fitting routine
 resNT = resN';
 degT = deg';
%%%%%%%%%%%%%temp plot of methane vs.wind direction
 figure(900)
 plot(deg,res,'k.','linewidth',3)
 title('Composite Methane vs.Wind Direction Histogram');
 ylabel('Methane Concentration (AU)');
 xlabel ('Wind Direction (deg.)');
 grid on
```

```
box on
%%%%%%%%%%%%%%%temp plot of count vs.wind direction
figure(910)
plot(deg,count,'k.','linewidth',3)
title('Measurment Count vs. Wind Direction');
ylabel('Count (#)');
xlabel ('Wind Direction(deg.)');
grid on
box on

%figure(k2)
afig = figure('visible','off')
plot(deg,resN,'ro','linewidth',1)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%weighted fit setting

NN = ones(size(resN));

[C,I] = max(resN);

% NN(I)= NN(I)*1.0;
% NN(I+1)= NN(I+1)*3.0;
% NN(I+2)= NN(I+2)*5.0;
% NN(I-1)= NN(I-1)*3.0;
% NN(I-2)= NN(I-2)*5.0;
% NN(I-3)= NN(I-3)*5.0;

resNNI = 1./NN;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x = degT;
y = resNT;
dy = resNNI;
dyT = dy';
gg = horzcat(resNT,dyT);
fw = ezfit(degT,gg ,'gauss');
showfit(fw,'fitcolor','black','maxlengtheq',0);

grid on
box on
set(gca,'fontsize',12);
ylabel('Concentration (ppm)');
xlabel ('Wind Direction (deg)');
xlim([0 360]);
legend('Data','Fit');
title('Methane Concentration vs Wind Direction');
%text(250,(max(resN)+ min(resN))/2,strcat(num2str(disney4),'_ _
_',num2str(disney5)), 'FontSize',10);
```

```
text(250,(max(resN)+
min(resN))/2,strcat(num2str(disney4),'#',num2str(disney5)), 'FontSize',10);
text(250,(max(resN)+ min(resN))/2.3,strcat('pgi_', num2str(pgt),'__pgy_',
num2str(pgty)), 'FontSize',10);
text(250,(max(resN)+ min(resN))/2.6,strcat('ti_',
num2str(turbuint),'__maxc_', num2str(maxmet)), 'FontSize',10);
text(250,(max(resN)+ min(resN))/2.9,strcat('wsf_',
num2str(wslimit),'__angf_', num2str(al)), 'FontSize',10);
text(250,(max(resN)+ min(resN))/3.2,strcat('count_', num2str(cqa),'__bin_',
num2str(bqa)), 'FontSize',10);

%Saving Figures in the current directory using the convention specified

print('-djpeg','-r300',strcat(num2str(disney4),'_',num2str(disney5)));

% Saving parameters to Summary processed files
    rt3rp3 = mpres/mtemp;
    %rt3rp3 = mpres/m3Dt;
    rt2rp2 = [rt2rp2;rt3rp3];
    coeffi = [coeffi;fw];
    disp(coeffi(k2).m(1:2))
    coeffi1 = [coeffi1; coeffi(k2).m(1:2)];
    rsq = [rsq; coeffi(k2).r];
    h1 = coeffi1(k2,1);
    a1bkg = lowmet + h1;
    a1bkg1 = [a1bkg1,a1bkg];

        emflux3 = mws.*2.*pi.*pgty.*pgtz;
     emflux2 = [emflux2;emflux3];
end
close all

% Converting to distance-based coordinates from angle space
sigmadeg = coeffi1(:,2);
sigmar = sigmadeg*(pi/180);
sigmadist = 0;
sigmadist = [];
for k3 = 1:length(dist)
    sigmadist1 = tan(sigmar(k3)/2)*2*dist(k3);
    sigmadist = [sigmadist;sigmadist1];
end

format short
mconct = coeffi1(:,1); % normalized methane concentration
mconctg = [];

% Calculating the peak methane concentration in g/m^3
for k4 = 1:length(mconct)
    mconct1 = mconct(k4) *opt* rt1rp1*rt2rp2(k4);
    mconctg = [mconctg;mconct1];
```

```
end

emflux = [];
 for k5 = 1:length(emflux2)
    emflux1 = emflux2(k5) .*mconctg(k5);
    emflux = [emflux;emflux1];
 end
a1bkg2 = a1bkg1.';
impval = [disney0 dist senh tor1 a1bkg2 methanemean impnum cangl1 bin wsl all
clpl coeffi1 mconctg sigmadist rsq binqa countqa tempqa stdws3tqa minws3tqa
maxws3tqa emflux];

percentcut = [disney0 cut0];


% Saving the calculated parameters as the main output file in csv format

xlswrite('Summary_Processed_Methane', impval);
```

Appendix C:
Example STR Data Output File in .csv Format

Distance	71
Mast Heading	180
FLIR File	
Photo File	
Notes	
Sensor Ht	0
Source Ht	0
Delay Time	0

Time	GPS Latitude	GPS Longitude	ID (-)	GPS Track	GPS Ground Speed	GPS Time	GPS DoP
2013-06-17T15:52:08+00:00	42.31482	-110.312	9	301.63	0.043496	71526.5	1
2013-06-17T15:52:08+00:00	42.31482	-110.312	9	75.84	0.093206	71527	1
2013-06-17T15:52:09+00:00	42.31482	-110.312	9	176.62	0.118061	71527.5	1
2013-06-17T15:52:09+00:00	42.31482	-110.312	9	160.04	0.16777	71528	1
2013-06-17T15:52:10+00:00	42.31482	-110.312	9	240.64	0.074565	71528.5	1
2013-06-17T15:52:10+00:00	42.31482	-110.312	9	255.07	0.093206	71529	1
2013-06-17T15:52:11+00:00	42.31482	-110.312	9	237.19	0.111847	71529.5	1
2013-06-17T15:52:11+00:00	42.31482	-110.312	9	315.72	0.155343	71530	1
2013-06-17T15:52:12+00:00	42.31482	-110.312	9	275.33	0.043496	71530.5	1
2013-06-17T15:52:12+00:00	42.31482	-110.312	9	88.75	0.086992	71531	1

SUMMA Pressure	WS Wind Speed	WS Wind Direction	WS Temperature	WS Rel Humidity	WS Bar. Pressure	WS Compass Heading
-0.1545	4.2	155	21.3	15	759.7	74
-0.1545	4.2	155	21.3	15	759.7	74
-0.16973	4.4	153	21.3	15	759.7	75
-0.17735	4.4	153	21.3	15	759.7	75
-0.19257	4.7	150	21.3	15	759.7	75
-0.19257	4.7	150	21.3	15	759.7	75
-0.1545	4.7	150	21.3	15	759.7	75
-0.22303	5	148	21.2	15	759.7	75
-0.2078	5.2	146	21.2	15	759.7	75
-0.20019	5.2	146	21.2	15	759.7	75

3DS U	3DS V	3DS W	3DS 2D Speed	3DS 3D Speed	3DS Azimuth	3DS Elevation	3DS SoS	3DS Sonic Temp	3DS Error Code
0.47	-4.35	-0.31	4.38	4.4	173.8	-4.1	343.82	20.21	0
0.71	-4.81	0.5	4.87	4.9	171.5	6	344.04	20.56	0
0.82	-4.17	0.36	4.26	4.28	168.9	4.9	344	20.51	0
1.32	-5.61	-0.03	5.77	5.78	166.8	-0.4	343.24	19.2	0
1.02	-5.63	-0.03	5.73	5.74	169.7	-0.4	343.56	19.74	0
1.07	-5.22	-0.2	5.34	5.35	168.4	-2.2	343.24	19.22	0
1.21	-4.94	0	5.09	5.1	166.2	0.1	344.04	20.59	0
0.73	-4.99	0.13	5.05	5.06	171.6	1.5	344.14	20.74	0
0.58	-5.14	0.11	5.18	5.19	173.5	1.2	343.4	19.48	0
0.16	-5.12	0.12	5.13	5.14	178.2	1.4	343.72	20.03	0

CO_2	CH_4	H_2O	H_2S	C_2H_2	BEN	TOL	ETB	M-Xylene	O-Xylene	P-Xylene	Cell Temp	Cell Pressure
394.07	2.13155				2.06691	-56.5593	32.5754	-115.096	-846.045	-143.712	28.7818	-48.5183
393.694	2.01101				2.06691	-56.5593	32.5754	-115.096	-846.045	-143.712	28.7818	-48.5183
393.869	2.06994				9.34052	-81.1741	56.8057	-105.084	-828.639	-155.119	28.7818	-48.6556
393.569	2.16082				9.34052	-81.1741	56.8057	-105.084	-828.639	-155.119	28.7818	-48.6556
393.363	2.16187				14.8855	-107.484	30.8417	-110.068	-943.659	-155.469	28.7818	-49.2048
393.898	1.99006				14.8855	-107.484	30.8417	-110.068	-943.659	-155.469	28.7818	-49.2048
394.691	1.92135				14.8855	-107.484	30.8417	-110.068	-943.659	-155.469	28.7818	-49.2048
395.464	1.89396				15.7865	-59.1672	45.1765	-126.08	-967.059	-150.171	28.7818	-49.0675
394.324	1.90753				15.7865	-59.1672	45.1765	-126.08	-967.059	-150.171	28.7818	-49.0675
394.007	1.95888				12.6283	-20.992	63.5617	-190.095	-1039.56	-149.438	28.7818	-48.7929

Appendix D:
Example Emissions Calculation Output File

This data output file consists of 70 columns, which contain both raw data and analyzed data output parameters. All column headings are shown below with three rows of data for illustrative purposes only.
An explanation of each column heading is presented at the end of this appendix.

Filename	Event	Distance (m)	Sensor Height (m)	Source Height (m)	a1 + bkgCH4 (ppm)	bkgCH4 (ppm)	Pressure (mbar)
9061113	1	41	0	0	3.216979363	1.916691736	775.8775728
9061113	2	41	0	0	2.924262341	1.956654918	775.7017908
9061113	3	41	0	0	2.041596769	1.880253061	775.5687247

<u>	<v>	<w>	<t>	<u2>	<v2>	<w2>	<t2>	<uv>	<uw>	<ut>	<vw>	<vt>	<wt>
0.004433997	-2.651242025	0.000544673	16.26443624	0.328409627	7.469092121	0.100007865	264.9929505	-0.012550073	-0.021820057	0.017940362	0.033188212	-43.04048895	0.092227583
0.00555057	-2.624626821	0.000912211	18.229593	0.453969726	7.31676515	0.110246536	332.9397548	-0.065346253	-0.004662591	0.271734862	0.0399352	-47.75575457	0.105465044
1.590637146	-2.050840628	0.15639557	20.03214575	3.289652273	5.492288689	0.188747247	402.1072589	-2.778545388	0.337928351	31.58096966	-0.141606196	-41.37586989	3.23636251

GPS Latitude	GPS STDev Latitude	GPS Longitude	GPS STDev Longitude	Mean CO_2 (ppm)	STDev CO_2 (ppm)	5% of the Data	Highest Methane (ppm)	Mean of the Highest 5% of Methane Conc (ppm)	STDev of the Highest 5% of Methane Conc (ppm)	Lowest Methane (ppm)	Background Methane (ppm)	STDev of the Background Methane (ppm)	Mean Methane (ppm)	Mean Methane - bkg (ppm)
42.824	2.24E-12	-109.89	6.88E-12	406.5931	1.744056	121.95	5.328	4.023955	0.321303	1.8997	1.916692	0.007905	2.339454	0.422763
42.824	2.27E-12	-109.89	6.94E-12	402.0346	1.463576	122.85	5.3117	3.988144	0.335346	1.9409	1.956655	0.005409	2.309635	0.35298
42.824	3.84E-13	-109.89	5.26E-13	399.2293	0.809249	49.4	2.4732	2.260284	0.090979	1.8552	1.880253	0.007914	2.004422	0.124169

3DS 3D Wind Speed (m/s)	Temperature (K)	Mean 2DWind (deg)	Mean 3DWind (deg)	Std WD 2D (deg)	Std WD 3D (deg)	Turbulent Intensity
2.716495	289.8942	98.23194	179.9042	14.02502	13.23402	0.116439
2.713028	291.9245	91.33561	179.8788	16.9099	15.209	0.122409
2.828824	293.8543	143.632	142.2028	30.63098	22.72537	0.143356

WS 2D PGI	WS 3D PGI	StdDev. WD 2D PGI	StdDev. WD 3D PGI	Turbulent Intensity PGI	Weighted Avg. PGI	Estimated PGI	PGσ_y (m)	PGσ_z (m)	Centroid Dir (deg)	Angle BinSize (deg)	Wind Speed Cut (m/s)	Accept Angle Limit (+/- deg)	Bin Density Cut (%)
3	3	5	5	5	5	5	5.46	3.29	155	10	0	60	2
3	3	4	5	5	4.5	5	5.46	3.29	155	10	0	60	2
3	3	1	3	4	2.5	3	8.53	4.62	95	10	0	60	2

a1 (ppm)	σ (deg)	CH4 a1 (g/m3)	σ_y (m)	rsq	BinQA	Count QA	Temp QA	3Dtemp stdev QA	3Dtemp min QA	3Dtemp max QA	PSG Emission Estimate (g/s)
1.300288	9.907165	0.000671	7.107135	0.973213	16	770	1.001658	0.679156	14.95	18.88	0.205844
0.967607	11.11894	0.000496	7.98161	0.982744	16	575	1.00187	0.788636	16.41	21.03	0.151885
0.161344	27.31359	8.22E-05	19.92395	0.930456	10	196	1.002292	0.906216	18.6	23.16	0.057542

Complete list of headings (defined on following pages):

1. Filename
2. Event
3. Distance (m)
4. Sensor Height (m)
5. Source Height (m)
6. a1 + bkgCH4 (ppm)
7. bkgCH4 (ppm)
8. Pressure (mbar)
9. <u>
10. <v>
11. <w>
12. <t>
13. <u2>
14. <v2>
15. <w2>
16. <t2>
17. <uv>
18. <uw>
19. <ut>
20. <vw>
21. <vt>
22. <wt>
23. GPS Latitude
24. GPS STDev Latitude
25. GPS Longitude
26. GPS STDev Longitude
27. Mean CO_2 (ppm)
28. STDev CO_2 (ppm)
29. 5% of the Data
30. Highest Methane (ppm)
31. Mean of the Highest 5% of Methane Conc (ppm)
32. STDev of the Highest 5% of Methane Conc (ppm)
33. Lowest Methane (ppm)
34. Background Methane (ppm)
35. STDev of the Background Methane (ppm)
36. Mean Methane (ppm)
37. Mean Methane - bkg (ppm)
38. 3DS 3D Wind Speed (m/s)
39. Temp. (K)
40. Mean 2DWind (deg)
41. Mean 3DWind (deg)
42. Std WD 2D (deg)
43. Std WD 3D (deg)
44. Turbulent Intensity
45. WS 2D PGI
46. WS 3D PGI
47. StdDev. WD 2D PGI
48. StdDev. WD 3D PGI
49. Turbulent Intensity PGI
50. Weighted Avg. PGI
51. Estimated PGI
52. PGσ_y (m)
53. PGσ_z (m)
54. Centroid Dir (deg)
55. Angle Bin Size (deg)
56. wind Speed Cut (m/s)
57. Accept Angle Limit (+/- deg)
58. Bin Density Cut (%)
59. a1 (ppm)
60. σ (deg)
61. CH4 a1 (g/m3)
62. σ_y (m)
63. rsq
64. BinQA
65. Count QA
66. Temp QA
67. 3Dtemp stdev QA
68. 3Dtemp min QA
69. 3Dtemp max QA
70. PSG Emission Estimate (g/s)

Heading Definitions

Filename: Study code and date of survey, denoted by SSMMDDYY where SS=study code, MM=month, DD=day, and YY=year.

Event: Event is the experiment number of the given day.

Distance (m): The distance from the measurement location to the methane emission source. This value is originally entered by the user in the field, and the MATLAB Analysis code pulls it from the STR file created.

Sensor Height (m): The height of the sensor. This value is entered by the user in the field. If no value is entered, the default value is zero.

Source Height (m): The height of the source, and often the user cannot measure it. In this case, the default value entered is zero.

A1 + bkgCH4: Peak methane value obtained from the Gaussian fit added to the background methane value

bkgCH4 (ppm): The background methane concentration which is calculated to be the mean of the lowest 5 % of the methane concentration values.

Pressure (mbar): The mean ambient pressure value measured in mbar.

<u>: The mean value of the velocity in the y direction

<v>: The mean value of the velocity in the x direction

<w>: The mean value of the velocity in the z direction

<t>: The mean temperature value in degrees Celsius

<u2>: The mean value of the dot product of u.u

<v2>: The mean value of the dot product of v.v

<w2>: The mean value of the product of w.w

<t2>: The mean value of the product of t.t, where t is measured temperature in degrees Celsius.

<uv>: The mean value of the dot product of u and v

<uw>: The mean value of the dot product of u and w

<ut>: The mean value of the dot product of u and t

<vw>: The mean value of the dot product of v and w

<vt>: The mean value of the dot product of v and t

<wt>: The mean value of the dot product of w and t

GPS Latitude: The average GPS latitude of the location where the measurements were collected

GPS STDev Latitude: Computed standard deviation of the GPS Latitude values

GPS Longitude: The average GPS longitude of the location where the measurements were collected

GPS STDev Longitude: Computed standard deviation of the GPS Longitude values

Mean CO2 (ppm): The average carbon dioxide values measured in ppm

STDEV CO2: Computed standard deviation of the carbon dioxide values

5% of the Data: Number of data values in 5% of the data

Highest Methane: Highest methane concentration value recorded during the survey

Mean of the Highest 5% of Methane Conc (ppm): Average of the highest 5 % of the methane concentration values measured in ppm

STDev of the Highest 5% of Methane Conc: Standard deviation of the highest 5 % of methane concentration values

Lowest Methane: The lowest methane concentration value measured during the survey

Background Methane: The average of the lowest 5 % of methane concentration values

STDev of the Background Methane: Standard deviation of the lowest 5 % of methane concentration values

Mean Methane: The average of all the methane concentration values recorded, after the time alignment, during the survey

Mean Methane-bkg: Value obtained after subtracting the average methane concentration value from the computed background methane concentration value

3DS 3D Wind Speed: The computed wind speed performed in the MATLAB Analysis code using data from the 3D Sonic Anemometer

Temp.: The average ambient temperature value during the survey, in degrees Kelvin

Mean 2DWind: The average value of the absolute wind direction computed by the MATLAB code using data from the AIO Compact Weather Station

Mean 3DWind: The average value of the azimuthal rotated wind direction calculated by the MATLAB code using data from the3D Sonic Anemometer

Std WD 2D: Standard deviation of the absolute wind direction measured with the 3D Sonic Anemometer

Std WD 3D: Standard deviation of the azimuthal rotated wind direction measured with the 3D Sonic Anemometer

Turbulent Intensity: Ratio of the standard deviation of the wind speed in the z direction to the calculated mean wind speed

WS 2D PGI: Wind Stability Class based on the AIO Compact Weather Station Wind Speed value

WS 3D PGI: Wind Stability Class based on the 3D Wind Speed value

StdDev. WD 2D PGI: Wind Stability Class based on the standard deviation of the wind direction measured by the AIO Compact Weather Station

StdDev. WD 3D PGI: Wind Stability Class based on the standard deviation of the wind direction measured by the 3D Sonic Anemometer

Turbulent Intensity PGI: Wind Stability class based on the turbulent intensity

Weighted Avg. PGI: This is the average of the Wind Stability Class

Estimated PGI: Since the weighted average PGI may not be an integer, the MATLAB analysis code converts the weighted average PGI value to an integer

PGσy: Plume width in the y-direction in meters retrieved from wind stability class tables presented in Appendix A

PGσz: Plume height in the z-direction in meters retrieved from wind stability class tables presented in Appendix A

Centroid Dir: Centroid angle in degrees

Angle Bin Size: Size of angle bins for Gaussian fit (default value of 10 degrees)

Wind Speed Cut: Wind speed filter value designating that wind speed values must be above a certain value to be included in data analysis

Accept Angle Limit: Value chosen for wind direction filter (default value is 60 degrees)

Bin Density Cut: Value designating that every wind angle bin must have this percentage of data to be included in the data analysis

a1: Peak methane value retrieved from Gaussian fit of methane concentration values

σ: Plume width in the y-direction in degrees retrieved from Gaussian fit of methane concentration values

CH4 a1: a1 value converted to units of g/m^3

σy: Plume width in degrees converted into units of distance (meters)

rsq: R square value of the Gaussian fit of methane concentration data, indicating how well the fit matches the data points

CountQA: Maximum number of values in the bin

TempQA: Ratio of the mean temperature measured by the AIO Weather Station to the mean temperature measured by the 3D Sonic Anemometer

3Dtemp stdev QA: Standard deviation of the 3D Sonic Anemometer temperature in degrees Celsius

3Dtemp min QA: Minimum value of the 3D Sonic Anemometer temperature in degrees Celsius

3Dtemp max QA: Maximum value of the 3D Sonic Anemometer temperature in degrees Celsius

PSG Emission Estimate: Methane emission estimate in grams per second

Appendix E:
WindCH4 Source.m Program Code

```
% This code is used to produce a plot of methane concentrations measured
during a survey as a function of wind direction.  The plot is utilized to
determine the source location(s)based upon
% the wind speed and the wind direction. The programs uses the absolute wind
% direction from the Climatronics wind monitor& the azimuthal from the Sonic
% anemometer.

clear

files2 = dir('Processed*.csv'); % Locating the Processed files from the
directory which are the Time Aligned files.
disney0 = [];
wdmeth = csvread(files2.name); % Reading the files

meth = wdmeth(:,5); % Methane Concentration

   %wddir = wdmeth(:,11); % Sonic Wind Direction
   wddir = wdmeth(:,7); % Climatronics Wind Direction
   wddir1 = wddir.*(pi./180); % Converting to Radians
   ws = wdmeth(:,10); % This is the Calculated wind speed from the Processed
Files, and this value was obtained from the calculation done in the Analysis
file
   lat = wdmeth(:,2); % Latitude of the Measurement
   long = wdmeth(:,3); % Longitude of the Measurement

   meanlat = mean(lat(:,1)); % Average of the Latitude values
   meanlong = mean(long(:,1)); % Average of the Longitude values

   for i1 = 1:length(meth)
      if meth(i1) == 0
         meth(i1) = 1;
      end
   end

% Specifying the naming convention of the Wind Files
disney = files2.name;
disney1 = isletter(disney);
disney2 = find(disney1 == 0);
disney3 = disney(disney2);
disney4 = str2num(disney3(2:8));
disney5 = str2num(disney3(10:end));
disney0 = [disney0; disney4 disney5];



[x,y,z] = pol2cart(wddir1,ws,meth); % Converting from the Polar to Cartesian
Coordinates

m = length(y);
```

```
n = length(x);
[m n] = size(z);



% Here, we are plotting the wind histogram; specifying the coordinate
% system in a clockwise polar coordinate; and then saving the files in the
directory
 rose(wddir1)
 hline = findobj(gca,'Type','line');
 set(hline,'LineWidth',2)
 hHiddenText = findall(gca,'type','text');

 view([90,-90]);

angles = 0:30:330;

hObjToDelete = zeros( length(angles)-2, 1 );
k = 0;
for ang = angles
   hObj = findall(hHiddenText,'string',num2str(ang));
   switch ang

      case 0
      set(hObj,'string','North','VerticalAlignment','Top','FontSize',12);
   case 90
      set(hObj,'string','East','HorizontalAlignment','Left','FontSize',12);
   case 180
      set(hObj,'string','South','VerticalAlignment','Top','FontSize',12);
   case 270
      set(hObj,'string','West','HorizontalAlignment','Right','FontSize',12);

   otherwise
      k = k + 1;
      hObjToDelete(k) = hObj;
   end
end
delete( hObjToDelete(hObjToDelete~=0) );

set(gca,'fontsize',15);
title('Wind Direction Histogram','FontWeight','Bold');

imagewdhist = getframe(gcf);
WindHist =
sprintf('WindHist#%s.jpg',strcat(num2str(disney4),'_',num2str(disney5)));
imwrite(imagewdhist.cdata, WindHist,'Quality',100);
```

```
%%%% This part of the code plots the methane concentration in variation
%%%% with wind speed and direction in a clockwise polar coordinate. This plot
does show the relative
%%%% location of the source(s).

source =  polar(wddir1,ws);

hold on
 set(source,'Visible','Off');


scatter(x,y,meth/5,meth,'fill','o','Linewidth',2);
text(mean(ws),mean(wddir1),'WS (m/s)','FontSize',10)
h1 = colorbar;

set(get(h1,'title'),'string','CH4 (ppm)','FontSize',15,'FontAngle','italic');


situate = get(h1,'title');
posit = get(situate,'position');

posit(1,:) = min(meth) - 1; % Specifying position of the colorbar title
posit(1,1) = posit(1,1) +1;
set(situate,'posit',posit);


set(gca,'fontsize',18);
title ('Methane Source(s)','FontWeight','bold');

 hHiddenText = findall(gca,'type','text');

  view([90,-90]);

angles = 0:30:330;

hObjToDelete = zeros( length(angles)-2, 1 );

%k = 0;
for ang = angles
   hObj = findall(hHiddenText,'string',num2str(ang));
   switch ang
      case 0
      set(hObj,'string','North','VerticalAlignment','Top','FontSize',12);

   case 90
      set(hObj,'string','East','HorizontalAlignment','Left','FontSize',12);
   case 180
      set(hObj,'string','South','VerticalAlignment','Top','FontSize',12);
```

```
   case 270
      set(hObj,'string','West','HorizontalAlignment','Right','FontSize',12);

   otherwise
%      k = k + 1;
%      hObjToDelete(k) = hObj;
   end
end
delete( hObjToDelete(hObjToDelete~=0) );


imagewd = getframe(gcf);
SourceIdentify =
sprintf('CH4Source#%s.jpg',strcat(num2str(disney4),'_',num2str(disney5)));
imwrite(imagewd.cdata, SourceIdentify,'Quality',100);









% Mapping
% This is optional part of the code; This code draws a US map and depending
% upon the given latitude and longitude limit it can direcly name the given
% state(s) and point the measurement location

latlim1 = [41    45];
longlim1 = [-111.05 -104.05];
 figure(1000)
ax1 = usamap(latlim1,longlim1);
axis off

states = shaperead('usastatehi',...
    'UseGeoCoords', true, 'BoundingBox', [longlim1', latlim1']);

faceColors = makesymbolspec('Polygon',...
    {'INDEX', [1 numel(states)], ...
    'FaceColor', polcmap(numel(states))});

geoshow(ax1, states, 'SymbolSpec', faceColors)

for k = 1:numel(states)
  labelPointIsWithinLimits =...
    latlim1(1) < states(k).LabelLat &&...
    latlim1(2) > states(k).LabelLat &&...
```

```
    longlim1(1) < states(k).LabelLon &&...
    longlim1(2) > states(k).LabelLon;
  if labelPointIsWithinLimits
    textm(states(k).LabelLat,...
    states(k).LabelLon, states(k).Name, ...
       'HorizontalAlignment',
'center','FontSize',15,'FontWeight','bold','FontAngle','italic')

  end
end

textm(meanlat,meanlong,'***',...
    'FontSize',20,'fontweight','bold','Rotation',0,'BackgroundColor',[0 1 0] )
title('Measurement Location','FontWeight','bold','FontSize',18);



imagemap = getframe(gcf);
imwrite(imagemap.cdata, 'Map.jpg','Quality',100);

a2 = imread(SourceIdentify, 'jpg');
ra2 = imref2d(size(a2));
a3 = imread('Map.jpg');
ra3 = imref2d(size(a3));
ra2.XWorldLimits = ra3.XWorldLimits;
ra2.YWorldLimits = ra3.YWorldLimits;



[a4,rc2] = imfuse(a2,ra2,a3,ra3,'montage','Scaling','none');
SourceMapIdentify =
sprintf('CH4S&MeasLoc#%s.jpg',strcat(num2str(disney4),'_',num2str(disney5)));

imwrite(a4,SourceMapIdentify,'Quality',100);




close all
```

MATLAB®-based OTM33_PSG_082715_Histcount.m PSG Analysis Program

Notes:

(1) Same files structure as previous version
(2) Updates to MATLAB® may be required for all included functions
(3) A version called "…Histc" is the same as "…Histcount" has a workaround for the "Histcount" function if not included in the MATLAB®Version

```
clear;
%version 092215 Halley's updated code using Histc
% Delete processed files from directory before each run
load pgtabley.mat; % Load lookup table for sigma y
load pgtablez.mat; % Load lookup table for sigma z
files = dir( 'STR*.xls');
countfiles = numel(files);

run = 09231502;   % Analysis run MMDDYY(see notbook)
wslimit = 0.0;  % Set wind speed cut limit (m/s)
al = 60;      % Set wind angle cut limit (+/- deg, 180 indicates no filter
clp = 2.0;     % Set bin density cut limit (%)
bb1 = 0         % Set start deg value for angle bin (deg)
bb2 =10        % Set bin size (deg)
bb3 = 360      % Set end bin (see Ln 12)
degB = bb1:bb2:bb3; % Wd bins
n1 = 40;    % Remove sampling time deplay

% Additional information for files
% Distances from observation to source
dist =
[40.45;59.83;35.18;97.44;56.99;87.76;87.76;98.4;98.4;102.96;102.96;102.96;82;5
7.00;57.00;81.00;97.80;65.40;41.00;92.90];
% Sensor heights
senh =
[2.69;2.69;2.69;2.69;2.69;2.69;2.69;2.69;2.69;2.69;2.69;2.69;2.69;2.69;2.69;2.
69;2.69;2.69;2.69;2.69];
% Source heights
sorh =
[3.0;3.0;3.0;3.0;3.0;3.0;3.0;3.0;3.0;3.0;3.0;3.0;3.0;3.0;2.0;3.0;3.0;3.0;3
.0];

% Constants used
rt1rp1 = 298/1013.25;
gasconst = 8.314510;
mw = 16.04;
grt1rp1 = rt1rp1*gasconst;
rt0rp0 = (gasconst*298)/101.325;
```

```
opt = (1e-06 * mw*1000)/rt0rp0;

% Listing of variables used in code
impnum = [];
disney0 = [];
coeffi = [];
coeffi1 = [];
rsq = [];
cut0 = [];
numabcknd = [];
pnum = [];
rt2rp2 = [];
methanemean = [];
binqa = [];
cangl1 = [];
countqa = [];
tempqa = [];
stdws3tqa = [];
minws3tqa = [];
maxws3tqa = [];
a1bkg1 = [];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Loop through files
%k1 = 6
for k1 = 1:countfiles
    % Import File
    jadu1 = xlsread(files(k1).name);
    time = double(jadu1(:,1));
    time = time*24;
    co2 = jadu1((1:end),2);     % c02 concentration (ppm)
    ch4 = jadu1((1:end),3);     % Ch4 concentration (ppm)
    lat = jadu1((1:end),4);     % GPS latitude
    long = jadu1((1:end),5);    % GPS longitude
    ws3 = jadu1((1:end),6);     % 3D sonic 3D wind speed (m/s)
    wd3 = jadu1((1:end),7);     % 3D wind direction (deg, arbitrary)
    ws2 = jadu1((1:end),8);     % 2D wind speed (m/s)
    wd2 = jadu1((1:end),9);     % 2D wind direction (deg, auto aligned north)
    temp = jadu1((1:end),10);   % Met station Temperature (deg C)
    pres = jadu1((1:end),12);   % Met Station Pressure (mbar)
    ws3z = jadu1((1:end),13);   % 3D w  (m/s)
    ws3x = jadu1((1:end),14);   % 3D u  (m/s)
    ws3y = jadu1((1:end),15);   % 3D v  (m/s)
    ws3t = jadu1((1:end),16);   % 3D t  (deg C)

    % Rewrite all arrays to remove time delay n1
    co2(1:n1) = [];
    ch4(1:n1) = [];
    c1 = length(co2);
    time(c1+1:end) = [];
    ws3(c1+1:end) = [];
    wd3(c1+1:end) = [];
    ws2(c1+1:end) = [];
    wd2(c1+1:end) = [];
```

```
temp(c1+1:end) = [];
pres(c1+1:end) = [];
ws3z(c1+1:end) = [];
ws3x(c1+1:end) = [];
ws3y(c1+1:end) = [];
ws3t(c1+1:end) = [];
lat(c1+1:end) = [];
long(c1+1:end) = [];

% Rotate 3D sonic coordinate system to streamlined set (-180 deg)
% First rotation (set <x> = 0)
RA = pi + (atan2(-1*(mean(ws3x)),-1*(mean(ws3y))));
nws3y = ws3y*cos(RA) + ws3x*sin(RA);
nws3x = (-1)*ws3y*sin(RA) + ws3x*cos(RA);
% Second rotation (set <z> = 0)
RB = pi + (atan2(-1*(mean(ws3z)),-1*(mean(nws3y))));
nnws3y = nws3y*cos(RB) + ws3z*sin(RB);
nws3z = (-1)*nws3y*sin(RB) + ws3z*cos(RB);
% Assigned streamlined coordates
ws3y = nnws3y;
ws3x = nws3x;
ws3z = nws3z;
% Calculate and asign new 3D sonic 2D wind direction
wd3 = 180 + (atan2(-1*(ws3x),-1*(ws3y))*57.29578);
% Calculate and asign new 3D sonic 2D wind speed
ws3 = (ws3y.*ws3y + ws3x.*ws3x).^0.5;
wd3(wd3<0) = 0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Wind angle filter
% Calculate mean ch4 conc by wd bin;
% Threshold for number of measurements per bin (2%)
cut = clp/100 * length(wd3);
%[countM, edges, wdbin] = histcounts(wd3, degB);
[countM2, wdbin] = histc(wd3, degB);
countM = countM2(1:(size(degB,2)-1)).';
resM = accumarray(wdbin, ch4, [], @(x) mean(x));
% Set bin-values below threshold to 0;
resM(find(countM < cut)) = 0;
% Find maximum;
[C,binx] = max(resM);
binxx = (bb2-bb1)*(binx) - bb2/2;
% If maximum near 0 or 360 edges, rotate wd3 by 180
if (binxx < 90 | binxx > 270);
    wd3(find(wd3 > 180)) = wd3(find(wd3 > 180)) - 360;
    wd3 = wd3 + 180;
    if (binxx > 270);
        binxx = binxx - 180;
    else;
        binxx = binxx+180;
    end;
end;

% Filter file based on wind criteria
```

```
    jadu2 = [time co2 ch4 ws3 wd3 ws2 wd2 temp pres ws3z lat long ws3x ws3y
ws3t];
    % Windspeed filter
    jadu2(find(ws3 < wslimit),:) = [];
    jadu2(find(wd3 < (binxx - al) | wd3 > (binxx + al)),:) = [];
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    a1 = jadu2;
    ws = a1(:,4);             % 3D soinic 2D wind speed (m/s)
    ws2 = a1(:,6);            % Met. Station 2D wind wpeed (m/s)
    temp = a1(:,8)+273.15;     % Met. Station temp (deg K)
    pres = a1(:,9);           % Met. Station atmospheric pressure (mbar)
    a1a = a1(:,5);            % 3D wind direction
    a1b = a1(:,3);            % Methane concentration
    a1c = a1(:,2);            % CO2 concentration
    ws3z = a1(:,10);           % 3D w
    ws3x = a1(:,13);           % 3D u
    ws3y = a1(:,14);           % 3D v
    ws3t = a1(:,15);           % 3D t
    maxmet = max(a1b);         % Maximum methane concentration
    minmet = min(a1b);         % Minimum methane concentration
    mmet = mean(a1b);          % Mean methane concentration
    mws = mean(ws);           % Mean 3D sonic wind speed (m/s)
    mws2 = mean(ws2);          % Mean met. Station 2D wind speed (m/s)
    mtemp = mean(temp);        % Mean met station temp. (K)
    mpres = mean(pres);        % Mean met. station atmospheric pressure (mbar)
    m3Dt = mean(ws3t)+273.15;   % Mean 3D sonic temp. (K)
    mco2 = mean(a1c);          % Mean CO2 concentration (ppm)
    stdco2 = std(a1c);         % Std dev in CO2 (ppm)
    stdws3t = std(ws3t);       % Std dev in 3D sonic t (C)
    minws3t = min(ws3t);       % Minimum in 3D sonic t (C)
    maxws3t = max(ws3t);       % Maximum in 3D sonic t (C)
    lat = a1(:,11);
    long = a1(:,12);
    lat1 = mean(lat(:,1));
    long1 = mean(long(:,1));
    latstd = std(lat);
    longstd = std(long);

    disney = files(k1).name;
    disney1 = isletter(disney);
    disney2 = find(disney1 == 0);
    disney3 = disney(disney2);
    disney4 = str2num(disney3(2:8));
    disney5 = str2num(disney3(10:end));
    disney0 = [disney0; disney4 disney5];

    % Taking the average of the lowest methane concentration values
    og = length(a1b);
    oga = round(og*0.05);
    lowval = sort(a1b);
    lowmet = mean(lowval(1:oga)); %lowest 5% values
    methanemean = [methanemean;lowmet];
    mmmet = mmet - lowmet;
```

```
    lowmetstd = std(lowval(1:oga));
    hival = sort(a1b,'descend');
    himet = mean(hival(1:oga));
    himetstd = std(hival(1:oga));

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %calculating raw mean products for use in Windtrax
    my = mean(ws3x);
    mx = mean(ws3y);
    mz = mean(ws3z);
    mt = mean(ws3t);
    y2 = ws3x.*ws3x;
    my2 = mean(y2);
    x2 = ws3y.*ws3y;
    mx2 = mean(x2);
    z2 = ws3z.*ws3z;
    mz2 = mean(z2);
    t2 = ws3t.*ws3t;
    mt2 = mean(t2);
    yx = ws3x.*ws3y;
    myx =  mean(yx);
    yz = ws3x.*ws3z;
    myz =  mean(yz);
    yt = ws3x.*ws3t;
    myt =  mean(yt);
    xz = ws3y.*ws3z;
    mxz =  mean(xz);
    xt = ws3y.*ws3t;
    mxt =  mean(xt);
    zt = ws3z.*ws3t;
    mzt =  mean(zt);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%calculating wind parameters

    %turbulent intensity
    sws3z = std(ws3z);
    turbuint = sws3z/mws;

    % Average and the standard deviation of wind 3D sonic azmuth wind
direction

    wd3mean = 180 + (atan2(-1*(my),-1*(mx))*57.29578);
    stddev3 = std(a1a);

    % Average and the standard deviation of wind direction for 2D met station
    wd2rad = a1(:,7) * pi/180;
    wd2y = sin(wd2rad);
    wd2x = cos(wd2rad);
    wd2num = numel(wd2rad);
    wd2xmean = mean(wd2x);
    wd2ymean = mean(wd2y);

    if wd2xmean > 0 && wd2ymean > 0
        wd2mean = atan(wd2ymean/wd2xmean);
```

```
        wd2mean = wd2mean * (180/pi);

        ep1 = wd2xmean.^2 + wd2ymean.^2;
        ep1 = sqrt(1-ep1);
        stddev1 = asin(ep1);
        stddev2 = 1+(1.1547 -1)*ep1.^3;
        stddev = stddev1 * stddev2;
        stddev = stddev * (180/pi);

    elseif wd2xmean > 0 && wd2ymean < 0
        wd2mean = atan(wd2ymean/wd2xmean);
        wd2mean = wd2mean * (180/pi);
        wd2mean = wd2mean + 360;

        ep1 = wd2xmean.^2 + wd2ymean.^2;
        ep1 = sqrt(1-ep1);
        stddev1 = asin(ep1);
        stddev2 = 1+(1.1547 -1)*ep1.^3;
        stddev = stddev1 * stddev2;
        stddev = stddev * (180/pi);

    elseif wd2xmean < 0
        wd2mean = atan(wd2ymean/wd2xmean);
        wd2mean = wd2mean * (180/pi);
        wd2mean = wd2mean + 180;

        ep1 = wd2xmean.^2 + wd2ymean.^2;
        ep1 = sqrt(1-ep1);
        stddev1 = asin(ep1);
        stddev2 = 1+(1.1547 -1)*ep1.^3;
        stddev = stddev1 * stddev2;
        stddev = stddev * (180/pi);

    end

    %%%%%%%%%%%%%%%%%% Seciton estimates sigmay and sigmaz estimate based on
stability class estimate

    %%% Wind Speed PG Indicator (3D)
    if mws > 0 && mws < 1.700;
        pgwsi = 1;
    elseif mws >= 1.700 && mws < 2.400;
        pgwsi = 2;
    elseif mws >= 2.400 && mws < 3.200;
        pgwsi = 3;
    elseif mws >= 3.200 && mws < 4.100;
        pgwsi = 4;
    elseif mws >= 4.100 && mws < 5.100;
        pgwsi = 5;
    elseif mws >= 5.100 && mws < 6.200;
        pgwsi = 6;
    elseif mws >= 6.200;
        pgwsi = 7;
```

```
end

%%% Wind Speed PG Indicator (2D)
if mws > 0 && mws < 1.700;
    pgws2i = 1;
elseif mws >= 1.700 && mws < 2.400;
    pgws2i = 2;
elseif mws >= 2.400 && mws < 3.200;
    pgws2i = 3;
elseif mws >= 3.200 && mws < 4.100;
    pgws2i = 4;
elseif mws >= 4.100 && mws < 5.100;
    pgws2i = 5;
elseif mws >= 5.100 && mws < 6.200;
    pgws2i = 6;
elseif mws >= 6.200;
    pgws2i = 7;
end

%%% Stdev Wind Direction PG Indicator (3D)
if stddev3 > 27.50;
    pgstddev3i = 1.00;
elseif stddev3 <= 27.50 && stddev3 > 23.50;
    pgstddev3i = 2.00;
elseif stddev3 <= 23.50 && stddev3 > 19.50;
    pgstddev3i = 3.00;
elseif stddev3 <= 19.50 && stddev3 > 15.50;
    pgstddev3i = 4.00;
elseif stddev3 <= 15.50 & stddev3 > 11.50;
    pgstddev3i = 5.00;
elseif stddev3 <= 11.50 & stddev3 > 7.50;
    pgstddev3i = 6.00;
elseif stddev3 <= 7.50
    pgstddev3i = 7.00;
end

%%% Stdev Wind Direction PG Indicator (2D)
if stddev > 27.50;
    pgstddevi = 1.00;
elseif stddev <= 27.50 && stddev > 23.50;
    pgstddevi = 2.00;
elseif stddev <= 23.50 && stddev > 19.50;
    pgstddevi = 3.00;
elseif stddev <= 19.50 && stddev > 15.50;
    pgstddevi = 4.00;
elseif stddev <= 15.50 && stddev > 11.50;
    pgstddevi = 5.00;
elseif stddev <= 11.50 && stddev > 7.50;
    pgstddevi = 6.00;
elseif stddev <= 7.50
    pgstddevi = 7.00;
end

%%% Turbulent Intensity PG Indicator (3D)
```

```
    if turbuint > 0.205;
       pgturbi = 1.00;
    elseif turbuint <= 0.205 && turbuint > 0.180;
       pgturbi = 2.00;
    elseif turbuint <= 0.180 && turbuint > 0.155;
       pgturbi = 3.00;
    elseif turbuint <= 0.155 && turbuint > 0.130;
       pgturbi = 4.00;
    elseif turbuint <= 0.130 && turbuint > 0.105;
       pgturbi = 5.00;
    elseif turbuint <= 0.105 && turbuint > 0.080;
       pgturbi = 6.00;
    elseif turbuint <= 0.080;
       pgturbi = 7.00;
    end

    %%%%this line calculates average pg indicator

    pgi = [pgstddevi + pgturbi]/2;

    %%% PGtable select
    if pgi < 1.500;
       pgt = 1.00;
    elseif pgi >= 1.500 && pgi < 2.500;
       pgt = 2.00;
    elseif pgi >= 2.500 && pgi < 3.500;
       pgt = 3.00;
    elseif pgi >= 3.500 && pgi < 4.500;
       pgt = 4.00;
    elseif pgi >= 4.500 && pgi < 5.500;
       pgt = 5.00;
    elseif pgi >= 5.500 && pgi < 6.500;
       pgt = 6.00;
    elseif pgi >= 6.500
       pgt = 7.00;
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    d = dist(k1);
    dd = round(d);
    pgty = pgtabley(dd,pgt);
    pgtz = pgtablez(dd,pgt);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%end of pg indicator section
    % Saving values
    impnum1= [mpres my mx mz mt my2 mx2 mz2 mt2 myx myz myt mxz mxt mzt lat1
latstd long1 longstd mco2 stdco2 oga maxmet himet himetstd minmet lowmet
lowmetstd mmet mmmet mws mtemp wd2mean wd3mean stddev stddev3 turbuint pgws2i
pgwsi pgstddevi pgstddev3i pgturbi pgi pgt pgty pgtz];
    impnum = [impnum; impnum1];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Binning measured concetration in in given degree increments
% Subtracting off the background which is the avg of the lowest 5% values
a1b = a1b - lowmet;

% Bin background subtracted CH4 by wd
%[count, edges, wdbin] = histcounts(a1a, degB);
[count2, wdbin] = histc(a1a, degB);
count = count2(1:(size(degB,2)-1)).';
resN = accumarray(wdbin, a1b, [], @(x) mean(x));
% Set bin-values below threshold to 0;
resN(find(count < cut)) = 0;
% Find maximum;
[C,bqa] = max(resN);
cangl = (bb2-bb1)*(bqa) - bb2/2;

% QA indicators
cqa = max(count);
tqa = mtemp/m3Dt;
binqa =[binqa;bqa];
cangl1 =[cangl1;cangl];
countqa =[countqa;cqa];
tempqa =[tempqa;tqa];
stdws3tqa = [stdws3tqa;stdws3t];
minws3tqa = [minws3tqa;minws3t];
maxws3tqa = [maxws3tqa;maxws3t];

% Centers data so it is located at 180 degree
degcent = 180;    % center of the binning
degmax = 360;     % maximum of the binning
degmin = 0;       % minimum of the binning
resNmax = find(resN == max(resN));
resNbtwn = find(resN >= min(resN) & resN <= max(resN));
x = (degcent - degB(resNmax));
degc = degB + x;
for i1 = 1:length(degc)
   if degc(i1) > degmax
      degc(i1) = degc(i1)-degmax;
   elseif degc(i1) < degmax && degc(i1) >= degmin
      degc(i1) = degc(i1);
   elseif degc(i1) < degmin
      degc(i1) = degc(i1) + degmax;
   end
end

degc = degc(2:end)-bb2/2;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Fitting routine
resNT = resN';
degT = degc';
fw = ezfit(degT, resNT ,'gauss');
```

```
    % Plot methane vs wind direction
    figure()
    plot(degc,resN,'b.','linewidth',0.5)
    set(gca,'fontsize',13);
    title('Average Methane Concentration vs. Wind Direction');
    ylabel('Methane Concetration (ppm)');
    xlabel ('Wind Direction (deg.)');
    grid on
    box on
    showfit(fw,'fitcolor','black','maxlengtheq',0);
    print('-djpeg','-r100',strcat(num2str(disney4),'_',num2str(disney5)));


    % Saving parameters to Summary processed files
    rt3rp3 = mpres/mtemp;
    %rt3rp3 = mpres/m3Dt;
    rt2rp2 = [rt2rp2;rt3rp3];
    coeffi = [coeffi;fw];
    disp(coeffi(k1).m(1:2))
    coeffi1 = [coeffi1; coeffi(k1).m(1:2)];
    rsq = [rsq; coeffi(k1).r];
    a1 = coeffi1(k1,1);
    a1bkg = lowmet + a1;
    a1bkg1 = [a1bkg1,a1bkg];

end

close all

% Converting to distance-based coordinates
sigmadeg = coeffi1(:,2);
sigmar = sigmadeg*(pi/180);
sigmadist = 0;
sigmadist = [];
for k3 = 1:length(dist)
    sigmadist1 = tan(sigmar(k3)/2)*2*dist(k3);
    sigmadist = [sigmadist;sigmadist1];
end

format short
mconct = coeffi1(:,1); % normalized methane concentration
mconctg = [];

% Calculating the peak methane concentration in g/m^3
for k4 = 1:length(mconct)
    mconct1 = mconct(k4) *opt* rt1rp1*rt2rp2(k4);
    mconctg = [mconctg;mconct1];
end

a1bkg2 = a1bkg1.';
impval = [disney0 dist senh sorh a1bkg2 methanemean impnum cangl1 ...
    bb2*ones(countfiles,1) wslimit*ones(countfiles,1)  ...
    al*ones(countfiles,1) clp*ones(countfiles,1) coeffi1 ...
```

```
    mconctg sigmadist rsq binqa countqa tempqa stdws3tqa ...
    minws3tqa maxws3tqa];

% Saving the calculated parameters in the excel sheets
dlmwrite('Summary_ProcessedWT2.csv', impval, 'precision', 10);
```

Appendix G:
Updated OTM 33A "R" Code

Notes:

(4) The primary R analysis files reproduced is named "calcPSG.R"
(5) This code is contained in the Zip file "PSG_Rcode .zip"
(6) All contained files are needed to run the code along with "R" and "RStudio" (recommended)
(7) The test data files are in the Data/STR subfolder

```
##############################################################################

# Estimate Rate Function

# Halley Brantley

# 10/01/15

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

library(plyr)

library(pracma)

library(ggplot2)

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


calcPSG <- function(STR, binwidth, wdfilt, datafolder, plot = TRUE) {

   ## STR: STR file loaded using import STR, includes STR$dat which contains

   ## concentration time series, and STR$runinfo that contains the

   ## distance from the soure, height of the source, height of the sensor
```

```
## and true release rate

## binwidth: width of wind direction bins (in degrees)

## wdfilt: wind direction cut-off (degrees), e.g. 60 indicates only use

## data when the wind is coming from the direction of peak concentration

## +/- 60

## datafolder: contains point source gaussian table


load(file.path(datafolder, "pgsigma.RData"))

bins <- seq(-360, 360, binwidth)

# Create data.frame with wind direction bin labels and centers

bins.df <- subset(data.frame(wdbin=cut(bins, bins, right=FALSE),

                                    thetabin=cut(bins, bins, right=FALSE),

                                    center=bins+binwidth/2), !is.na(wdbin))

dat <- STR$dat

distance <- STR$runinfo[, "distance"]

sensor_ht <- STR$runinfo[, "sensor_ht"]

source_ht <- STR$runinfo[, "source_ht"]

release_rate <- STR$runinfo[, "release_rate"]

dat$time_int <- seq(1, nrow(dat), 1)


# Aggregate data by wd bins

dat$wdbin <- cut(dat$wd3, bins, right=FALSE)

ch4.wdbin1 <- ddply(subset(dat, !is.na(wdbin)), .(wdbin), summarize,

                    ch4 = mean(ch4L),
```

```
                    n = length(ch4L))

# Add bin center to aggregated values

ch4.wdbin <- merge(subset(ch4.wdbin1, n> 0.01*nrow(dat)), bins.df)


# Set initial values for gaussian fit estimation

mu0 <- ch4.wdbin[which(ch4.wdbin$ch4==max(ch4.wdbin$ch4)), "center"]

k0 <- max(ch4.wdbin$ch4)


# Fit Gaussian curve to wd bins

wdfit <- summary(nls(ch4 ~ k*exp(-1/2*(center-mu)^2/(sigma^2)),

                    start=c(mu = mu0, sigma=10, k = k0),

                    weights = n, dat=ch4.wdbin, algorithm = "port"))


# Center wind direction plume peak

dat$theta <- dat$wd3 - wdfit$coeff[1]

dat$theta <- ifelse(dat$theta > 180, dat$theta-360, dat$theta)

dat$theta <- ifelse(dat$theta < -180, dat$theta+360, dat$theta)


# Plot Gaussian fit to wind direction

if (plot==TRUE) {

   mu.wd <- wdfit$coeff[1, 1]

   sigma2.wd <- (wdfit$coeff[2, 1])^2

   a.wd <- wdfit$coeff[3, 1]
```

```
  # Plot Gaussian Fit
  ch4.wdbin$preds <- a.wd*
    exp(-0.5 * ((ch4.wdbin$center - mu.wd)^2/sigma2.wd))

 fig1 <- ggplot(ch4.wdbin, aes(x=center, y=ch4), col="black") +
    geom_point(data=dat, aes(x=wd3, y=ch4L), alpha=0.2)+
    geom_point() +
    geom_point(aes(y=preds), col="red") +
    geom_line(aes(y=preds), col="red") +
    theme_bw(base_size=16) +
    xlab("Wind Direction") +
    ylab("CH4 above background")
print(fig1)
}
##############################################################################
# Subset file to only include wind from the source +/- wind filter
dat.filt <- subset(dat, theta > -wdfilt & theta < wdfilt)
dat.filt$thetabin <- cut(dat.filt$theta, bins, right=FALSE)

ch4.thetabin1 <- ddply(subset(dat.filt, !is.na(thetabin)), .(thetabin), summarize,
                       ch4 = mean(ch4L),
                       n = length(ch4L)
                       )
```

```
# Add bin center to aggregated values

ch4.thetabin <- merge(subset(ch4.thetabin1, n > 1), bins.df)


# Calculate proxy distance (based on wd) along axis perpendicular to

# wind direction (Ly)

ch4.thetabin$Ly <- distance*sin(ch4.thetabin$center*pi/180)

dat.filt$Ly <- distance*sin(dat.filt$theta*pi/180)


# Fit Gaussian curve to proxy distance

# set initial values

sigma0 <- (max(ch4.thetabin$Ly)-min(ch4.thetabin$Ly))/5

mu0 <- ch4.thetabin[which(ch4.thetabin$ch4==max(ch4.thetabin$ch4)), "Ly"]

a0 <- max(ch4.thetabin$ch4)

param.fit <- NA

try(param.fit <- summary(nls(ch4 ~ a*exp(-1/2*(Ly-mu)^2/(sigma^2)),

                             weights = n,

                             start=c(mu = mu0, sigma=sigma0, a=a0),

                             dat=ch4.thetabin, algorithm = "port")))


mu <- param.fit$coeff[1, 1]

sigma2 <- (param.fit$coeff[2, 1])^2

a <- param.fit$coeff[3, 1]

ch4.thetabin$Dy <- 1/sqrt(2*pi*sigma2)*

   exp(-0.5 * ((ch4.thetabin$Ly - mu)^2/sigma2))
```

```
ch4.thetabin$predDy <- ch4.thetabin$Dy*a*sqrt(2*pi*sigma2)


if (plot == TRUE) {
# Plot Gaussian Fit
   fig2 <- ggplot(ch4.thetabin, aes(x=Ly, y=ch4), col="black") +
      geom_point() +
      #geom_point(aes(y=Dy*a*sqrt(2*pi*sigma2), col="red")) +
      geom_line(aes(y=predDy), col="red") +
      theme_bw(base_size=16) +
      xlab("Ly") +
      ylab("CH4 above background")
   print(fig2)
}
################################################################################
# Constants from Gryning et al 1983
IDvar <- which(names(dat.filt) == "ID")
temp <- dat.filt$temp
pres <- dat.filt$pres
ws3 <- dat.filt$ws3
ws3w <- dat.filt$ws3w
wd2_raw <- dat.filt$wd2_raw


n <- nrow(dat.filt)
Tbar <- mean(temp)
```

```
Pbar <- mean(pres)

Ubar <- mean(ws3)


ws_sd <- sd(ws3)

ep <- sqrt(1-(mean(sin(wd2_raw*pi/180))^2+mean(cos(wd2_raw*pi/180))^2))

wd_sd <- asin(ep)*(1 + (2/sqrt(3) - 1)*ep^3) * 180/pi # Yamartino Method

turbint <- sd(ws3w)/Ubar


# PGI from turbulence

PGturbi <- as.numeric(as.character((cut(turbint, turbint.breaks, labels=rev(seq(1,7,1))))))

# PGI from wd sd

PGstddevi <- as.numeric(as.character(cut(wd_sd, wdsd.breaks, labels=rev(seq(1,7,1)))))

# Calculate average PGI, round up if 0.5

PGI <- round((PGstddevi + PGturbi)/2 + 0.0001)

# Merge calculated values with lag file and PSG summary file

dist.int <- round(distance)

pgsigmay <- as.numeric(pgsigma[which(pgsigma$dist.int == dist.int &

                                     pgsigma$PGI == PGI), "sigmay"])

pgsigmaz <- as.numeric(pgsigma[which(pgsigma$dist.int == dist.int &

                                     pgsigma$PGI == PGI), "sigmaz"])


# Unit conversion constants

rt1rp1 <- 298/1013.25; gasconst <- 8.314510; mw <- 16.04;

rt0rp0 <- (gasconst*298)/101.325; opt <- (1e-06 * mw*1000)/rt0rp0;
```

```
  # Convert a1 to g/m3
  a1_gperm3 <- a*opt*rt1rp1*Pbar/Tbar
  # Calculate PSG
  PSG <- 2*pi*a1_gperm3*Ubar*pgsigmay*pgsigmaz
  ##############################################################################
   # Check for non-representative plumes
  ch4.thetabin <- ch4.thetabin[order(ch4.thetabin$center), ]
  tails1 <- ch4.thetabin$Dy[1]*a*sqrt(2*pi*sigma2)
  tails2 <- ch4.thetabin$Dy[nrow(ch4.thetabin)]*a*sqrt(2*pi*sigma2)
  NonGauss <- FALSE
  if (tails1 > 0.1 | tails2 > 0.1) {
     NonGauss <- TRUE
  }
  return(list(ID=IDvar, PSG=PSG, a_ppm=a, Ubar = Ubar, PGI = PGI, NonGauss = NonGauss))
}
```