UNIVERSITY OF OSLO
Department of Informatics

Statistical Analysis of Concentration Fluctuations as Detected by LIDAR backscatter

Master's Thesis

Lei Yang

December 2014

On the front is the ink painting 'Shrike on a Dead Branch', by Miyamoto Musashi. It can be taken as a representation of a natural logarithm, and is for instance similar to the shape of the boundary layer velocity profile [1].

Abstract

Experimentally obtained data of LIDAR backscatter experiments performed by DTU Risø during the MADONA campaign was analyzed with emphasis on concentration fluctuations of atmospherically diffused contaminants in the context of toxic contaminants. The collected data contained concentrations measured along the LIDAR line of sight downstream and perpendicular to released tracer aerosols.

Probability density functions and joint probability density functions for concentration and time-derivatives of concentrations are generated and visualized through Gaussian smoothing. The reduced information in the lowest order moments is investigated, in particular, to test for possible analytical relations between these quantities. Detailed results for excess statistics are obtained, such as average level crossing frequencies for varying positions in the plumes, and average excess durations. The results are placed in a framework of theoretical results for turbulent diffusion. Several conditions of atmospheric LIDAR measurements were considered both in fixed and moving center of mass frames of the plumes. Possible universal properties of these results will have value for safety precautions in relation to accidents where toxic material is released, as well as cases of heavy local pollution released as contaminant plumes.

[1] Drawing from www.flowillustrator.com/images/MaterialDerivative.jpg.

Acknowledgements

This thesis exists because Prof. Hans E. Jørgensen and Prof. Torben Mikkelsen from DTU Risø allowed me to work on their LIDAR data, painstakingly collected during the MADONA experiments. I am grateful both for the opportunity to work on the data, as well as for dispensation to attend their PhD summer school. The stay at Risø was the kick I needed to engage more deeply with the data at a time when I had trouble even formulating the analytical basis for an analysis.

I am especially indebted to my supervisor, Prof. Hans L. Pécseli, who allowed me a great deal of freedom in the way I pursued the project, offering his support throughout, while suggesting physics readings as I needed them. Since I was the only person in the group with a project related to this topic, I am very grateful to Hans for always keeping an open door whenever I wanted to discuss a new result, as well as for reading through my drafts innumerable times. It's hard to forget that you came over to the office on a Saturday evening near the deadline to check up on my progress. (Especially since it was yesterday.)

I am grateful to Bjørn Lybekk, one of our group's senior engineers, for his initial translation of the encrypted MADONA data to a form where I could store them into plain text files. He also made a very nice plot for showing estimation of excess statistics for me, fig.6.9, and has been welcoming and friendly from the first time I became part of the group. I am also grateful to Dr. Wojciech Jacek Miloch, our teacher in a course on Plasma Physics, who ended up closer to personal tutor as people dropped out over the semester. I can no longer remember how many times he had to listen to me derive the Debye length. Wojciech unwittingly taught me aspects of linear fluid theory throughout the course.

It is not at all clear in what precise form the physics department want their Master's theses. I therefore want to thank Anne S. Bergsaker for sending me the LateX template she used in her own thesis. I would also like to thank Dr. Vegard Lundby Rekaa for the sum of many encouraging words, and in particular for pointing me towards Hans as thesis supervisor, and my "officemates" Klaus Normann and Vigdis Holta for discussions about physics, and some necessary procrastination. Thanks are also in order for Atta-Ul-Monem Ayaz, who gave me some nice last minute tips, and Helén Vikdal Thorbjørnsrud for reading through my draft. I also want to thank Senior Consultant Grete Stavik-Døvle, who has always been the person in the Physics Department Administration that I went to with the most inane questions and requests.

Finally, and in particular, I would like to thank my parents, Yang Jiansheng and Cui Wei, for their support throughout the years.

Contents

Chapter 1

Introduction

> The real justifications of these definitions, however, will reside in their implications. (On the use of entropy as a measure of information.)
>
> *C. E. Shannon*
> A Mathematical Theory of Communications

This thesis analyses the often violent concentration fluctuations of tracer *plumes* released into the near-ground atmospheric boundary layer. The plumes considered here contain continuous and steady releases of non-reactive atmospheric aerosols. We analyze their structure at a position downstream with emphasis on their use in understanding the diffusion of accidentally released toxic contaminants or pollution within the *atmospheric boundary layer flow*.

The motivation for this analysis is its potential application in the understanding, and possible prediction, of toxic plume concentration distributions. Toxic contamination or pollution can be separated into the categories of *long-term* or *immediate* hazards. The problems caused by stable species, such as greenhouse gases, are difficult to quantify, but for a long time scale, accurate measurements may be taken of the amount of contaminants in the atmosphere. The form of the initial diffusion across the atmosphere is less important than the overall quantity, so a statistical model of particle distributions immediately after release is correspondingly less interesting in this case. The locations of these contaminants are also distributed throughout the atmosphere, and not limited to the surface layer of the Earth. These large scale problems are outside the scope of our project.

This thesis is instead concerned with the category of local hazards. The hazards of immediate contaminants, such as unintentionally released toxic contaminants,

are primarily local. Their harmful consequences are heavily dependent on the spread of the concentration distribution as it diffuses according to atmospheric flow shortly after release. This is also true for environments near factory pipe releases, where local populations are chronically subjected to medium to low concentration toxicity over large amounts of time. The distribution of these plumes is driven by the near-surface wind conditions, and the shape of the terrain. Surface wind conditions are typically turbulent, and therefore generally unpredictable. Because the greatest harm occurs where population density is greatest, urban environments should ideally be considered. Buildings in an urban environment will manifest as surface roughness from the point of view of the atmospheric fluid flow. Large scale gas dispersion experiments in a city are unfeasible. Complex terrain experiments like MADONA can therefore act as substitutes, because they to a certain extent take into account the *complex* surface roughness, as compared to a controlled lab environment. Analysis of data along these lines has been reported in relation to industrial accidents (Nielsen, Chatwin, Jørgensen, Mole, Munro and Ott, 2002). Similar studies have been conducted in relation to pollution due to smell from industry or farming (Mikkelsen and Jørgensen, 2002). A careful literature search have provided very few studies like these. The closest are the two previously mentioned reports.

The experiments were done during the *MADONA* campaign in September and October 1992 (see Cionco et al., 1999) using an aerosol backscatter *LIDAR* (Jørgensen, Mikkelsen, Streicher, Herrmann, Werner and Lyck, 1997) with periodic measurements spaced at $T = 3s$, held at ground level and measuring in a horizontal line perpendicular to the plume movement. The measured concentrations are averaged cross sections distributed along up to 800 meters in the LIDAR line of sight. Noise was subsequently removed by the DTU Risø group, and the data was packaged with programs made for display. Due to the age of the programs, we were unable to directly access the data. The tools were originally written in FORTRAN, and one of our senior engineers, Bjørn Lybekk, wrote a script for color density visualization of the data. Figs.1.5a and 1.5b were created using this program, and showcase the variable spatial spread of the data after noise reduction by the original MADONA team. I subsequently added lines to output the data in ASCII without additional encryption. The analysis performed in this thesis is based on these data sets, which are concentration measurements over time in a line parallel to the ground and perpendicular to the flow. We received MADONA LIDAR backscatter measurement data sets, as well as permission to work on them, from *DTU Risø*, and have therefore constrained this analysis accordingly.

Our plan was to create an analogy between the LIDAR measured gas release from MADONA and case study of pollution, with existing data on the state of the population after exposure. We then wanted to generalize potentially universal

aspects of the problem through an estimate of the true probability density functions the data are sampled from. Since this combination, for LIDAR data, has to our knowledge never been done before, we made an effort to formulate the basic problem through a "brute force" estimate prior to more detailed analyses. The entirety is divided into three parts:

(i) Comparing MADONA data sets with realistic pollution measurements. Pollution measurements are typically time averaged single-positional concentration data. In many cases, only a single averaged data point corresponds to the same plume, due to for instance changing wind conditions and the often worse time resolution compared to LIDARs. We therefore scaled the center of mass (CM) concentration of the MADONA data set we considered best, *mad21K*, to correspond with the local maximum of a single-positional data set gathered from an inhabited area with an active volcano (section 1.3) and compared estimates based on concentration thresholds (c_t) and their associated exposure time limits using environmental air quality standards. Based on qualitative agreement with medical records, we concluded that the analogy seemed viable. We identified three quantities that are of interest: percentage time ($\%t$) above c_t, frequency of fluctuations, measured as one-way crossings, (f_{cross}), over c_t, and the expected time ($\langle T \rangle$) above c_t for each crossing. The MADONA data sets are capable of giving these for a continuous concentration threshold and over one dimensional space, in both CM frame and fixed frame (FF).

(ii) Analytical and data processing tools and methods. The three quantities mentioned above were found for continuous thresholds and positions. We found the data to be rich enough to support analysis on this level, and three dimensional visualization of crossings in particular yielded qualitative understanding of the dangers of low-concentration fluctuations some distance from the CM. We attempted model fits corresponding to physical interpretations with falsifiable basis on $\langle T \rangle$ and $\%t$.

Concepts in fluid dynamics, turbulent similarity theory, and diffusion were discussed from a "first principles" point of view. These allowed us to understand the underlying physics of the problem, and in particular limitations in using LIDAR to perform statistical sampling of atmospheric diffusion of aerosols. It is explicitly assumed, as in all related studies (Pope, 2000), that the contaminants are passively following the flow.

(iii) Generalization. What we mean by generalization is to estimate the underlying *true* probability density function (PDF) $p(c)$, as well as the joint probability density function (JPDF) $p(c, dc/dt)$, by binning measured occurrences of concentration ranges. Due to the amount of data samples for each position (449 for *mad21K* after removing blank data), we have reason to believe that the data set constitutes a good estimate of the true distribution. Much longer measurement

times are not necessarily better, since wind conditions could change. If any feature of the $\langle T \rangle$, f_{cross} or $\%t$ is universal, it can be represented through the PDF and JPDF (sec.6.3), which in principle contain all the information of the statistical distribution. In practice, the PDFs we obtain will be limited by the resolution of the binning, which again depends on the number of samples. In order to facilitate estimation of histogram shapes for the human eye, we introduce Gaussian smoothing (sec.6.4).

We did not have a template for this work, since we found no previous analysis of LIDAR data for statistical inferences of contaminant concentration fluctuations in existing literature. Since we were free to formulate the problem as we worked, both analytical methods and data processing tools were typically used on the data right after they were developed. For this reason, the application of tools on the data, and the results found, are often discussed in the same section where they are presented, rather than in a large discussion towards the end of the thesis. I have tried to summarize the most important results in the conclusion.

1.1 MADONA campaign

Figure 1.1: Picture taken of a plume release during the MADONA experiments

(a) Gas containing sulfur dioxide leaks from Miyake-jima during the volcanic eruption in 2000. The gas plumes from the volcano after resettlement are smaller.[1]

(b) Plumes from the volcano on Miyake island September 2011.[2]

Figure 1.2

The purpose of the MADONA experiments was to collect high resolution, complex terrain, meteorological data from *"diffusion experiments using smoke, sulphurhexafluoride (SF_6), and propylene gas during unstable, neutral, and stable atmospheric conditions"* (Cionco et al., 1999). The MADONA scientists performed concentration fluctuation measurements on both short bursts of tracer gas (*puffs*) and long continuous releases of tracer gas (*plumes*). In this paper we consider plume releases, because they best simulate the emission of common toxic pollutants, such as factory releases. An additional reason is that continuous releases are needed to estimate the underlying probability distributions. This is necessary because we, in the future, want to compare possibly universal features of the distribution to similar experiments, as well as extend results to analogous situations.

The plumes in the MADONA experiments were created by mixing liquid $SiCl_4$ with a 25% solution of NH_4OH, and blown out through a strong air jet created by two constant flow rate pumps. SF_6 gas was mixed with the plumes and puffs. 18 LIDAR-measured plume data sets were collected over 7 days. Out of these, we selected 5 sets, namely *mad21K*, *mad21G*, *mad21F*, *mad21H* and *mad15J*, as high resolution data sets to work with. We mainly work with the data set *mad21K*.

Fig.1.1 shows one of the plumes released during the MADONA experiments. A LIDAR station can be seen measuring the concentration across the plume at a fixed position in a line normal to the flow centerline. In this application, it is important that data from the low concentration edges of the plume are intact, and the 5 data sets were therefore partially selected on these grounds.

1.2 Initial Considerations

We first wish to ascertain whether experimental measurements of released plumes over complex terrain in Salisbury, U.K. can realistically be generalized to account for similar situations in different geographical locations, under different wind conditions and for different atmospheric contaminants. One requirement for such an analogy is that the data must be scaleable with respect to incomplete parameter data from the situation where it is to be applied. The moments of the PDF contain all the information of the distribution (Ch.3), but if we had the PDF of the actual problem there would be no need for external data sets. In many situations, only rough averages of concentrations are available, which are sufficient to estimate large-scale concentration movements, but fail in describing the finer details of heavily fluctuating concentrations guided by the turbulent atmosphere. We could then extend the superior estimates of true plume PDFs from LIDAR measurements like MADONA to point concentration measurements in *similar* locations. We can test the feasibility of the analogy *qualitatively* by noticing that for people standing at a distance away from the center of the plume, the nature of the fluctuation may decide whether they are safe or in danger. Thus the fruitfulness of fitting detailed data to similar situations is dependent on its predictive ability for low level concentration fluctuations.

Table 1.1: Sulfur Dioxide Exposure Limits

time [min]	concentration [ppm]
15	5
60	0.075

A pertinent case study is *sulfur dioxide* SO_2 poisoning. SO_2 is a common factory-produced air pollutant. According to the U.S. Environmental Protection Agency, inhalation leads to adverse respiratory effects including bronchoconstriction and increased asthma symptoms. SO_2 at high concentrations leads to formation of other SO_X, which react with atmospheric compounds to form small aerosols. These aerosols penetrate into the lungs, and can cause or worsen respiratory disease, such as bronchitis (NAAQS, 2014). In this thesis, the SO_2 is used as a label to refer to contaminant gas consisting mainly of sulfur dioxide, with elements of SO_X and associated aerosols. The short-term exposure limit, at 15 minutes, is 5 p.p.m.,

[2]Picture to the left above: http://cais.gsi.go.jp/Virtual_GSI/Volcanology/2000_Miyake/miyake-index.html

[2]Picture to the right above: http://static.panoramio.com/photos/large/58953262.jpg

while the 1 hour standard is 0.075 p.p.m., per 2010 (NAAQS, 2014). See table (1.1) for convenience.

To study the short-term exposure distribution of such an atmospheric contaminant, one needs a statistically significant amount of temporal concentration data points obtained at least along one dimension. Concentration measurements sampled over time can be found for various atmospheric measuring stations around the world, especially for geologically active sites, such as volcanoes. However, the measured data sets do not cover spatial dimensions, and the publicly available data is often in the form of concentration averages over several minutes. Armed with only this average number, it is hard to say anything about the statistical concentration distributions of actual contaminants. Therefore, a suitable compromise seems to be the use of detailed LIDAR measurements taken from the MADONA sets, *scaled* onto the given, averaged values of the concentration data from a suitable measuring station exposed to the contaminant.

We needed measurements of a controlled population living in polluted, complex terrain. The area could not be urbanized, if we wanted to compare with the MADONA data sets. The best fit was Miyake island, south of Tokyo. It had a population of 2884 by the 2006 census, living near an active volcano (figs.1.2a, 1.2b). Due to the sulphur dioxide which periodically leaks from the volcano, the residents of Miyake village must at all times have a gas mask ready, and often wear it. A big volcano eruption in July 2000 forced the inhabitants from the island, but most of the residents decided to return in 2005, when sulphuric gas release levels had not yet normalized to pre-2000 levels. Toxic contamination on the island can serve as a case study, because average and maximum measurements of SO_2 levels have been measured and published for several "zones" around the island, alongside medical data concerning the health of longterm residents (Iwasawa et al., 2009). The MADONA data set *mad21K* was chosen as an appropriate high-resolution specimen for comparison (fig.1.4a).

Our "brute force" analysis is limited to measurements of the two thresholds in table 1.1 with their associated tolerance times. We consider the CM frame in order to postpone discussion of parameters. This allows us to introduce the specifics of translating the data to CM, scaling, threshold measurements, and suggests topics to be investigated in detail.

Three *quantifiable* central factors we can surmise are:

- Percentage of time spent in concentrations above the concentration thresholds.

- Average time the concentration spends above each of the concentration

[2]From: http://cais.gsi.go.jp/Virtual_GSI/Volcanology/2000_Miyake/miyake-index.html

[3]Picture from: http://all-that-is-interesting.com/the-town-where-everyone-wears-a-gas-mask

Figure 1.3: Picture taken at Miyake-jima, year unknown.[3]

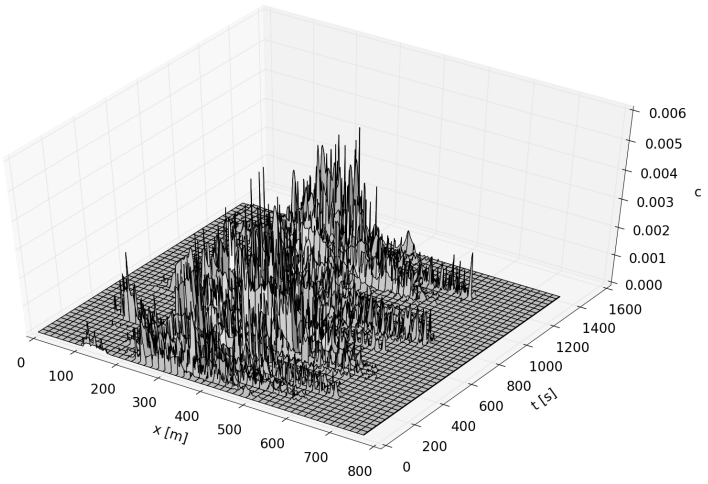
thresholds.

- Frequency of higher-than-threshold exposure, as measured by one-way crossings over a concentration threshold.

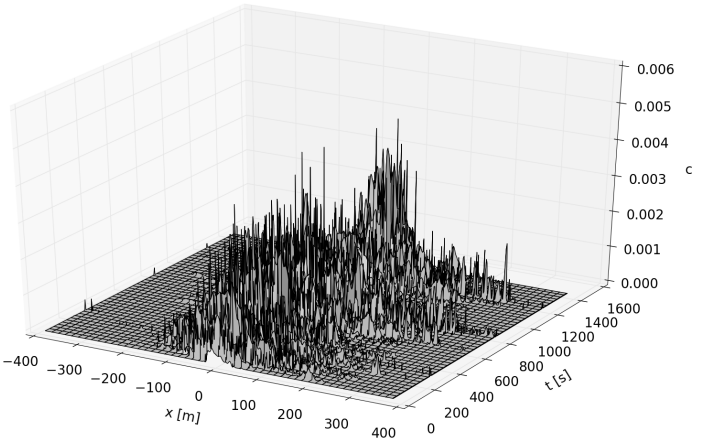
We chose these factors because they can be sampled directly from the data by setting a concentration threshold, as well as computed from the estimated true PDFs. This means that serve both as a measure of the "wellness" of PDF estimates, as well as the "wellness" of using LIDAR data to estimate the distribution near point measurement concentrations.

The average and maximal concentrations of SO_2 in Miyake reported in (Iwasawa et al., 2009) were taken from fixed stations on the island in the period between February 2005 and November 2006, in specified geographical zones. The largest maximal 5-min average of all the zones was 17.25 [ppm], while the lowest maximal 5-min average of all the zones was 4.27 [ppm]. Because these are 5-min averages, they will be assumed to represent average concentration rates of centers of plumes drifting from the volcano. As a smallest estimate, the 4.27 [ppm]. average will be considered.

Parameters like wind direction and speed are not considered in this brute force application. The data set *mad21K* was therefore translated to its center of mass

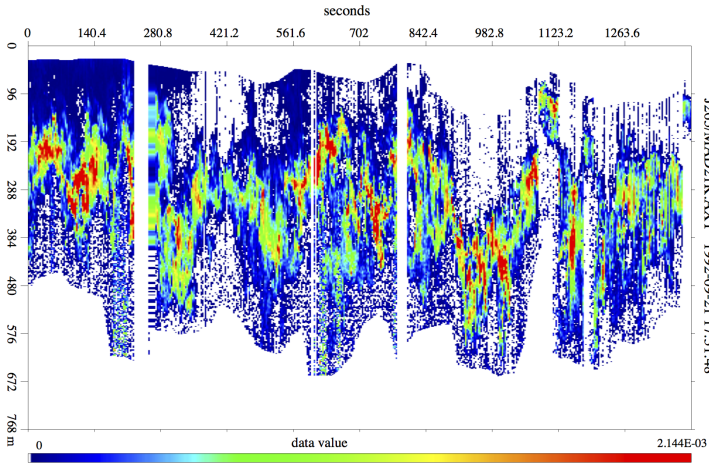


(a) Madona data set mad21K. Concentration c is in unnormalized LIDAR signal extinction units. Lab frame.

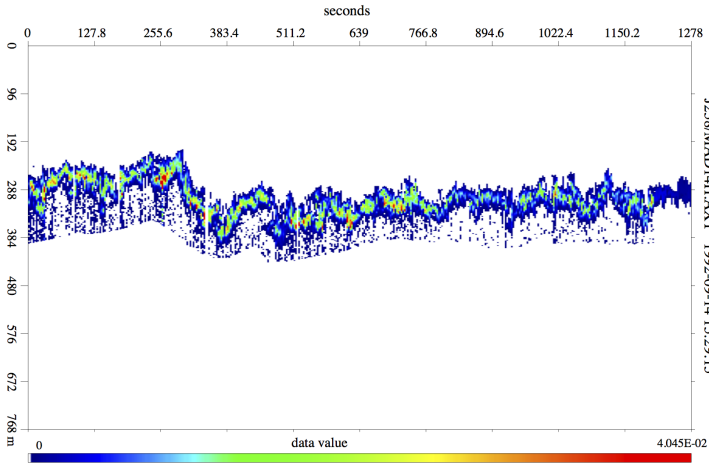


(b) Madona data set mad21K. Concentration c is in unnormalized LIDAR signal extinction units. Center of mass frame

Figure 1.4



(a) *mad21K*. Gaps in the data, seen as white vertical "stripes" are avoided in the analysis. These gaps appear only in some of the data sets. The entire sub-array from analysis if all positions at the sample time have $c == 0$.

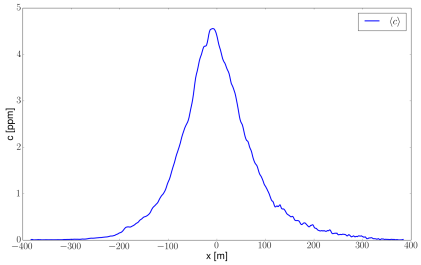


(b) *mad14H*

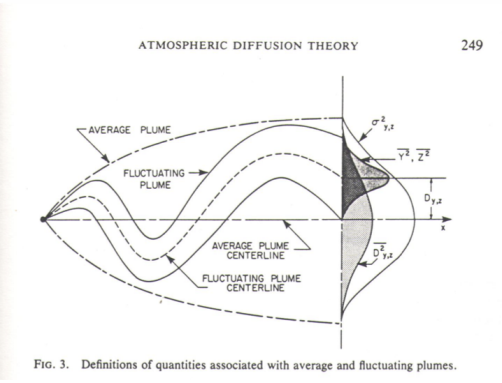
Figure 1.5: Colour coded density plot of the data. Made by Bjørn Lybekk.

frame (fig.1.4b). The definition of the center of mass is (§8 Landau and Lifshitz, 1976)

$$R = \frac{\sum_i m_i r_i}{\sum_i m_i},$$



(a) Madona data set mad21K. Temporal average of concentration in CM frame [R]. Note the slight skewness and subsequent trailing tail to the right.



(b) Illustration of definitions of average and fluctuating plumes, as given in (Seinfeld, 1983).

Figure 1.6

where we take the concentration as directly proportional to mass. It represents the reference frame of the motion of the toxic cloud as a whole. This "well-behaved" plume is the best case scenario for someone who is exposed to a toxic contaminant plume, since he can move away from high concentrations. In FF, the effect of the wind serves to make the high concentration positions less predictable.

Note that the data in the CM frame (fig.1.4b) is contained in arrays of the same size as those in the FF. This means some of the data on the edges for high wind fluctuations are cut off from the CM representation. This was done consciously in order not to artificially "skew" the data by introducing positions that are not sampled for the entire data set.

$\%t$, $\langle T \rangle$ and f_{cross} above the two exposure thresholds were considered for the CM of the plume, and about one Gaussian standard deviation away from it on both sides. This distance was estimated by considering that if the plume is distributed normally, 68.2% of the concentration will be distributed within one standard deviation away from the center of mass on both sides. The Gaussian distribution is a solution for the case of a purely classical diffusion of contaminants, sec.4.1, and can be treated theoretically. However, it does not fit atmospherically diffused concentration distributions well (Munro, Chatwin and Mole, 2003). In particular, the FF frame data often exhibit strong skewness. However, the CM frame *average* concentration, plotted over the 1D cross-section in fig.1.6a, is more symmetric. In this section, we call this estimated Gaussian standard deviation σ, and use it as a measure of distance.

The averaging process in fig.1.6a serves to obscure several important details. The time-averaged distribution seems to be nowhere near the 15 minute limit from table 1.1, yet post hoc results from table 1.2 reveal that the center of mass in fact spends 33% of the time above this threshold. Concentration fluctuations are glossed over in this form. By average exposure time above thresholds is meant the

time spent above the given concentration threshold at a time, averaged over all occurrences in the data. It is unclear what values this takes in the distribution tail on the right side of the CM. This is significant for the 1 hour threshold. An initial brute force analysis can therefore start by counting concentration occurrences above the given thresholds over time, for selected positions.

1.3 Brute Force Approach

1.3.1 Methods

In order to quantitatively predict the toxicity levels of a contaminant plume, we need the time spent above a given concentration, averaged over all occurrences of fluctuations above the threshold. In theory, we want

$$\langle T \rangle = \lim_{\tau \to \infty} \frac{1}{\tau} \int_0^\tau f(t|c_t) dt,$$

where $f(t)$ is the unknown, underlying function that determines the time spent above a threshold, and c_t is the given threshold.

Because we have discrete measurements, the estimated time differs from the real expectation value $\langle T \rangle$. Our approach is to divide the *number of data points above* c_t by the *number of crossings over* c_t. Since data points rarely end up directly on the threshold, the pertinent question is how to pinpoint the exact spot where the crossing occurs. In the brute force analysis, we make the arbitrary rule that the flag for the timer that starts incrementing above c_t starts counting once a data point with concentration $c > c_t$ is found, and stops counting once it spots a data point with concentration $c < c_t$. We assume that some of the error by this method are offset on average, although we can see that a linear interpolation between two data points on either side of the threshold would yield more accurate results. The $\%t$ excess durations, total crossings, and the average excess duration, $\langle T \rangle$, were found for the two thresholds in 1.1 using these rules.

The concentrations in the data supplied by the MADONA data sets were scaled to the toxic levels of SO_2 plumes on Miyake by averaging 10 data points around the center of mass of the plume average shown in fig.1.6a. All concentrations were linearly scaled according to the ratio found by comparing this average maximum level with the *smallest* of the 5-minute maximum levels given by (Iwasawa et al., 2009), which was 4.27 [ppm]. Our assumption was that a 5-minute maximum corresponds to a sensor which for considerable portions of that time was located in the center of the plume. By taking 10 data points around the center of mass, we can account for the fact that the sensor was not *exactly* in the center of that plume during those 5 minutes. Finally, the lowest maximum was used as a conservative

reference point. This is because a distribution analysis can pinpoint situations where low excess concentrations occur over long periods of time, and simulate the fact that the inhabitants are used to living in hazardous conditions, and use their gas masks properly.

Note that the CM frame arrays are of the same size as the FF arrays. This means that some information on the edges is lost if the CM at any time is significantly different from the CM of the rest of the data. We did this consciously, since these fluctuations would otherwise skew the data artificially if the position they represent in CM space moved outside of the LIDAR sampling space.

1.3.2 Results

Table 1.2: Preliminary Results

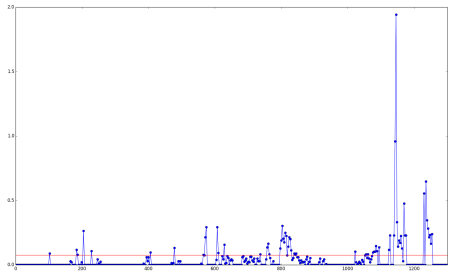
Measured	R - 2σ	R - σ	R	R + σ	R + 2σ
Crossings for $c_t = 5$ [ppm]	0	7	59	2	2
Crossings for $c_t = 0.075$ [ppm]	25	43	6	65	45
$\langle T \rangle$ for $c_t = 5$ [ppm]	0s	3.86s	7.29s	4.50s	3.00
$\langle T \rangle$ for $c_t = 0.075$ [ppm]	7.44s	20.4s	148s	12.6s	5.53s
$\%t$ for $c_t = 5$ [ppm]	0%	2.00%	33.9%	0.67%	0.45%
$\%t$ for $c_t = 0.075$ [ppm]	13.8%	65.3 %	98.4%	60.6%	18.5%

The results from data set mad21K were scaled against the 4.27 [ppm] average by taking 10 points across from the center of mass over the entire $1407s$ measurement. This is the method we use throughout the thesis for analyzing the MADONA data sets with realistic concentration thresholds. This scaling is performed separately for each data set.

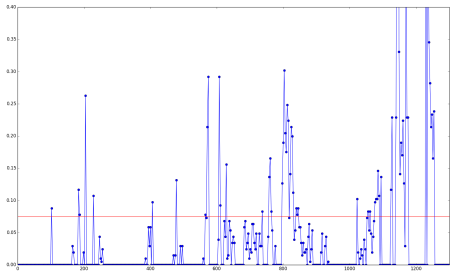
The two cutoff concentrations considered were the 15 [min] limit at 5 [ppm] and the 1 [hour] limit at 0.075 [ppm]. The time-averaged data set (fig.1.6a) exhibits a slightly larger tail towards the right. The fluctuations also seem stronger here than on the left side, and the concentration itself does not die off as fast. We are interested in how this manifests through the SO_2 thresholds. Snapshots of the concentrations can be seen in figs.1.7a, 1.7c, 1.7d, 1.7e and and 1.7f.

1.4 Discussion

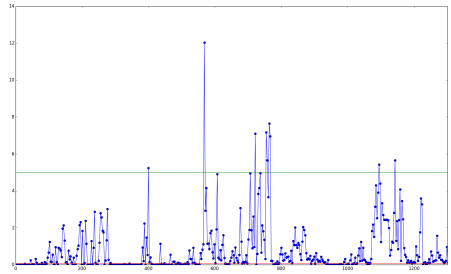
The resulting data calculated was based on the bullet points above can be seen in Table 1.2. As expected, the number of crossings of the higher concentration threshold is only significant near the center of the plume. Due to the skewness, there are more crossings at σ left of the CM than to the right. Since the inhabitants



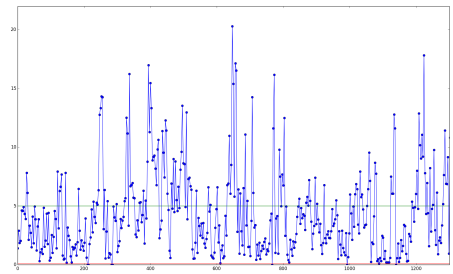
(a) Data set *mad21K*: 2σ towards the LIDAR from CM.



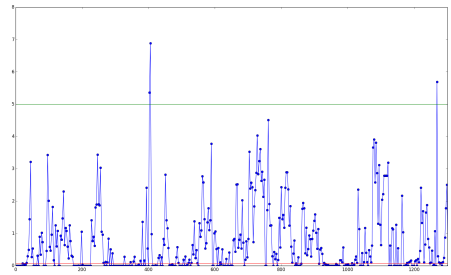
(b) Zoomed in version of fig.1.7c.



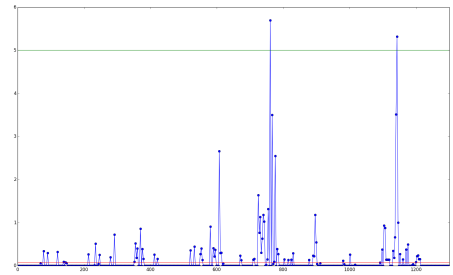
(c) Data set *mad21K*: σ towards the LIDAR from CM.



(d) *mad21K*: sampled at CM.



(e) Data set *mad21K*: σ in the direction opposite to the LIDAR, from the CM.



(f) Data set *mad21K*: 2σ in the direction opposite to the LIDAR, from the CM.

Figure 1.7: Red line represents 60 min. c_t, green line represents 15 min. c_t.

of Miyake deal with toxic fumes daily, we expect them to react appropriately close to the CM. However, one "standard deviation" corresponds to 127.5 meters. At this range, it might be harder to estimate the danger levels. Hence, the 1% of time spent above the 15 [min] threshold is of some concern. The biggest problem is at 2σ (255 [m]) away from the center of mass. Here an actual 28% of the time is spent above the 1 [hour] threshold, which means that 4 [hours] without a gas mask in seemingly safe surroundings will expose a person to potentially dangerous levels of SO_2. Considering a random exposure over several months, this *least harm* "brute force" analysis finds dangers of long-term respiratory problems

developing. The conclusions of the medical science team that studied the patients between autumn 2004 and November 2006 were that 'the study subjects showed no deterioration in lung function, but that prevalence of cough and phlegm among all participants were significantly higher in 2006 than in 2004. SO_2 exposure-related respiratory symptoms were observed in adult Miyakejima residents after returning to the island' (Iwasawa et al., 2009). This concurs with the "brute force" analysis, but would not necessarily agree with a blind estimate using only point measurements averaged over time.

1.5 Framework for a detailed analysis

In order to quantify the results measured in the first part of this chapter, we illustrate some basic characteristics with reference to fig.1.6b.

No violent up-down motions of the plume means stable stratification, as discussed in section 2.2. This implies that there are few or no "jumps" in the data. An example of such a "jump" is around 1120 [s] in fig.1.5a. Note that we do not mean the entirely blank "gaps" in the same figure, which are due to lack of LIDAR data at those sample times. The motion of a plume released by a consistent source in a mean wind with additional horizontal random motions give rise to a 'meandering' of the plume as illustrated in fig.1.6b.

A horizontally fluctuating plume can be characterized by its local average width and its local average position. Analytical studies address the mean square values of both these quantities. A summary of basic results is presented in fig.1.6b.

The basic elements needed to give analytical predictions of excess statistics (i.e. average crossing frequencies and average excess durations) are discussed in section 6.4.

Chapter 2

Analytical Concepts

> What I cannot create, I do not understand.
>
> *R. P. Feynman*
> on his blackboard at the time of his death

In this chapter we briefly discuss relevant concepts related to linear classical mechanics. This allows for a discussion of the mathematical description of the fundamental ideas in fluid flow, which in the interest of clarity is discussed from first principles. Limiting factors in atmospheric diffusion experiments include the parameters of *reference frame, flow speed, viscosity*, and *stability*. These are experimental variables that a responsible data analyst must take into consideration in order to avoid the errors of blind data processing.

2.1 Fluid Dynamics

Transport of a toxic, or otherwise unwanted dissolved material in the atmosphere is dependent on whether the contaminant is reactive with the surrounding atmosphere. In so far as it is reactive, this is a job for the *chemist*, but the transport properties themselves are dependent on fluid *flow*, which is studied in the domain of *fluid dynamics*.

The frame of reference of the observer is here important because the smallest unit considered is still a collection of molecules, which are in general *not* vibrating around a fixed position in a 'grid'. This collection of molecules, called a fluid *element*, is itself a frame of reference wherein molecules fluctuate in and out. It is known as the *Lagrangian frame*, and is the *rest frame* of the fluid element.

Variables, e.g. the velocity field $\mathbf{u}$, are here specified by

$$\mathbf{u}(\mathbf{x}(t, \mathbf{y}(t_0)), t),$$

where $\mathbf{y}$ is the position at reference time t_0 (Pope, 2000). On the other hand, the lab frame is called *Eulerian*, and variables are specified in the standard form

$$\mathbf{u}(\mathbf{x}, t),$$

where the positional vector $\mathbf{x}$ is an independent variable measured by a stationary observer outside the flow.

Fluid dynamics is a *continuous approximation*, and its proper language is therefore calculus. The fundamental rules are the same as for classical mechanics, namely *conservation of mass* and *conservation of momentum/force balance*. The mass density of fluid elements is denoted ρ. An expression of conservation of mass in the absence of sources and sinks is that the *mass density flux* $\rho\mathbf{u}$ out of the surface $\hat{\mathbf{n}}dA$, with $\hat{\mathbf{n}}$ pointing outwards from the body, is the same as the *loss rate* $\frac{\partial \rho}{\partial t}$ of the density throughout the volume (§1, Landau and Lifshitz, 1989). In the Lagrangian frame,

$$-\int_V \frac{\partial \rho}{\partial t} dV = \oint_S (\rho \mathbf{u}) \cdot \hat{\mathbf{n}} dA,$$

and using the *divergence theorem*,

$$\int_V \left[\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) \right] dV = 0,$$

which must be true for an arbitrary volume, so that

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0, \tag{2.1}$$

called the *continuity equation*. This can be expressed differently:

$$\frac{D\rho}{Dt} + \rho \nabla \cdot \mathbf{u} = 0,$$

where $\frac{D}{Dt} = \frac{\partial}{\partial t} + \mathbf{u} \cdot \nabla$ is called the *convective* or *material* derivative. It should be noted that the convective derivative is *not* simply the total derivative with respect to time, but rather emerges due to the frame of reference. This can be seen because the velocity field $\mathbf{u}(\mathbf{x}, t)$ is defined for an *independent* variable $\mathbf{x}$, rather than $\mathbf{x}(t)$, as it would have been if the convective derivative was a total derivative (Hazeltine and Waelbrock, 1998). In deriving the continuity equation (2.1), a fluid element of arbitrary size was considered implicitly in its *rest frame*, and the mass flux was related to this element. On the other hand, position is an independent variable only

in an Eulerian frame. The convective derivative therefore represents the derivative in the *Lagrangian* frame - for an observer following the fluid element - expressed in an *Eulerian* field variable $\mathbf{u}$.

If the fluid element conserves its density in the Lagrangian frame it is *incompressible*. This happens when the fluid element is unaffected by pressure variations, and even in gases when local pressure variations are small enough (Batchelor, 2000).

Thus for incompressible fluids,

$$\frac{D\rho}{Dt} = 0,$$

and consequently

$$\nabla \cdot \mathbf{u} = 0. \tag{2.2}$$

Momentum conservation can be expressed similarly to the above derivation of the continuity equation (2.1) (Pécseli, 2013). The momentum flux $\rho\mathbf{uu}$ through the surface $\hat{\mathbf{n}}dA$ is balanced by the volume rate of change of momentum $\frac{d(\rho\mathbf{u})}{dt}$. Unlike mass conservation, however, momentum sources and sinks are common in the form of both *long range* and *local* forces, here represented by the sum of all force densities $\mathbf{F}$. For any given volume,

$$\int_V \frac{\partial(\rho\mathbf{u})}{\partial t} dV = -\oint_S \rho\mathbf{uu} \cdot \hat{\mathbf{n}} dA + \int_V \mathbf{F} dV,$$

and again using the divergence theorem,

$$\int_V \left[\frac{\partial(\rho\mathbf{u})}{\partial t} + \nabla \cdot (\rho\mathbf{uu}) - \mathbf{F} \right] dV = 0. \tag{2.3}$$

Using the vector relation $\nabla \cdot (\mathbf{AB}) = (\nabla \cdot \mathbf{A})\mathbf{B} + (\mathbf{A} \cdot \nabla)\mathbf{B}$,

$$\int_V \left[\rho \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \frac{\partial \rho}{\partial t} + (\nabla \cdot (\rho\mathbf{u}))\mathbf{u} + \rho\mathbf{u} \cdot \nabla\mathbf{u} - \mathbf{F} \right] dV = 0,$$

where the 2^{nd} and 3^{rd} terms sum to 0 by the continuity equation (2.1). Again, the integrand must be 0 for an arbitrary volume, whence

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla\mathbf{u} \right) = \mathbf{F},$$

which is the general form of the force density balance needed. It is a representation of Newton's second law, and may be more explicitly written as

$$\rho \frac{D\mathbf{u}}{Dt} = \mathbf{F}.$$

This is another reminder that the convective derivative is the time variation in the fluid element's rest frame.

Applicable long range force fields may be superposed on the description as needed. These fields must be applied on a case-by-case basis, and are not predicted by a local theory. The common field in an electro-neutral fluid is the gravitational field, although in other manifestations of fluid theory, such as *magneto-hydrodynamic* (MHD) descriptions of plasma, other fields such as that for the force due to an imposed magnetic field should also be included (Pécseli, 2013).

The local force field acting on an arbitrary fluid volume may be expressed in terms of the pressure gradient (§2, Landau and Lifshitz, 1989)

$$-\oint_S p\hat{\mathbf{n}}dA = -\int_V \nabla p dV.$$

If the gravitational acceleration is added,

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\frac{\nabla p}{\rho} + \mathbf{g}.$$

This is called *Euler's equation*. It is sufficient for many purposes, for instance usually in hydrodynamic wave theory, but in other cases internal friction in the form of *viscous* forces must be accounted for.

A general way of looking at the variation of momentum in a fluid element is to consider the tensor formulation of the integrand in eq.(2.3) in the absence of long range fields. Let the local force be expressed by the gradient of the pressure, as in Euler's equation:

$$\frac{\partial(\rho u_i)}{\partial t} = -\frac{\partial}{\partial x_j}(p\delta_{ij} + \rho u_i u_j).$$

Evidently, the momentum is time-propagated by the *spatial* variation of a tensor quantity. It is named the *momentum flux tensor* Π, and describes here the thermodynamically reversible transfer of momentum in the fluid volume due to mechanical transport and internal, *adiabatic* pressure differences. The governing equation for momentum transport in the absence of long range fields is therefore:

$$\frac{\partial(\rho u_i)}{\partial t} = -\frac{\partial \Pi_{ij}}{\partial x_j}. \tag{2.4}$$

To incorporate the effects of internal friction, an additional, thermodynamically irreversible term called the *viscous stress tensor* σ can be added:

$$\Pi_{ij} = p\delta_{ij} + \rho u_i u_j - \sigma_{ij}.$$

The viscous stress tensor together with the pressure constitute the *stress tensor* T, the mathematical representation of the internal forces that manifest when fluid

elements interact with each other. Its components T_{ij} represent the i^{th} component of stress on a surface with normal vector pointing in the j direction (Acheson, 1990). Atmospheric gases are in general *Newtonian fluids*, and for these the viscous stress tensor is a linear function of the first derivatives of velocity. It can be shown (§15 Landau and Lifshitz, 1989) that for Newtonian fluids, the momentum equation can then be written as

$$\rho\left[\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}\right] = -\nabla p + \mu\nabla^2\mathbf{u} + (\zeta + \frac{1}{3}\mu)\nabla(\nabla\cdot\mathbf{u}),$$

where μ is the *dynamic* viscosity coefficient. ζ is sometimes called the *second viscosity*. Together with eq.(2.1) this constitutes the full Navier-Stokes equations, and can be solved to model a fluid flow given appropriate boundary and initial conditions. They can be simplified substantially by assuming incompressible flow, and become

$$\begin{aligned} \nabla\cdot\mathbf{u} &= 0 \\ \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} &= -\frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{u}. \end{aligned} \tag{2.5}$$

$\frac{\mu}{\rho} = \nu$ is the *kinematic* viscosity coefficient. Long range fields such as gravity may be superposed on the solution when needed. In theory, given boundary and initial conditions, the solution of the Navier-Stokes equation set (2.5) will give a complete description of the fluid flow in a given system.

Two issues then arise. (**i**) Under what circumstances can incompressibility be assumed in the context of atmospheric physics? (**ii**) Is the set of Navier-Stokes equations generally solvable under atmospheric conditions?

(**i**) The condition for incompressibility can be illustrated by assuming compressible perturbations $p = p_0 + \tilde{p}$ and $\rho = \rho_0 + \tilde{\rho}$ in an *irrotational, ideal gas*. The velocity field can then be characterized by a scalar potential through $\nabla\phi = \mathbf{u}$. It can be shown (§64 Landau and Lifshitz, 1989) that this potential satisfies the wave equation

$$\frac{\partial^2\phi}{\partial t^2} - C^2\nabla^2\phi = 0,$$

where the velocity of sound $C = \sqrt{\partial p/\partial\rho}$ under constant entropy. In the case of a plane wave, this can be solved together with the adiabatic property $\tilde{p} = (\partial p/\partial\rho_0)_s\tilde{\rho}$ (Landau and Lifshitz, 1980) to give the 1 dimensional condition

$$\frac{u}{C} = \frac{\tilde{\rho}}{\rho_0}.$$

Since the right hand side of the equation represents the *compressibility* of the gas, a rough estimate for incompressibility is $u << C$, or in terms of the Mach number $M = \frac{u}{C}$,

$$M << 1.$$

This is satisfied under normal wind conditions, considering that a moderate breeze on the *Beaufort scale* (MET, 2014) is only $\sim 6-8$ m/s, while the speed of sound through air is ~ 343 m/s. On the other hand, the presently highest measured wind speed we could find is 113 m/s (WMO, 2014), and for these sorts of anomalous speeds the incompressibility assumption is obviously very poor. Rapidly traveling objects will compress the air, but the aerosols considered in this thesis are simply released into the wind and considered as passive scalars. This thesis therefore assumes incompressible flow.

(**ii**) Whether the Navier-Stokes set is solvable is dependent on the flow conditions. Consider a decomposition of the field velocity into a mean and a fluctuating part:

$$\mathbf{u} = \mathbf{u}_0 + \tilde{\mathbf{u}}.$$

If the flow can be linearized, i.e. fluctuations to the 2^{nd} order in the non-linear term $(\mathbf{u} \cdot \nabla)\mathbf{u}$ may be neglected, then the momentum flux tensor Π will not depend on *shear* stress terms $\tilde{u_i}\tilde{u}_{j\neq i}$, and the set can then be closed. However, the atmosphere is not mainly driven by linear processes, and the momentum transfer cannot be accounted for purely by normal stresses, since these contribute little to momentum transport (Tennekes and Lumley, 1972). The *shear* stresses cannot be found within the Navier-Stokes equations without generating additional unknowns. This is the closure problem of *turbulent* flow, which will be discussed in chapter 3.

2.2 Instability

Regarding the conditions under which LIDAR measurements of concentration fluctuations are taken, a distinction is made between *stable* and *unstable* atmospheric flow. Given that the LIDAR measures a contaminant plume from a fixed position, good statistical data is dependent on the plume being relatively stationary with respect to the LIDAR line of sight, so that the concentration fluctuations measured over time are always related to the same horizontal axis in the plume's reference frame. The meandering of the plume along the LIDAR line of sight is analyzed in the fixed frame statistical analysis, but vertical meandering is suggested by the sudden lowering of concentration throughout the data. This can be difficult to spot for a data analyst who was not present during the measurements, considering that large fluctuations occur naturally in a turbulent fluid. Even though atmospheric instability onset can be described as a deterministic phenomenon which can be solved through classical physics, the unstable phenomenon itself cannot be fully described deterministically. Deliberate measurements of unstable plumes may be fruitful, but even if the LIDAR could follow the center line of the plume as it moves vertically, the change of angle necessitates an assumption of isotropy if the

data is to be analyzed statistically. Therefore, one dimensional LIDAR scattering sensors may not be the best experimental tool for this sort of measurement.

What remains to be shown is that atmospheric instability onset can be derived mathematically through classical physics. One point of interest is what happens to a fluid element that is transported in a local gravitational field in Earth's boundary layer. Since a fluid element is not a single molecule, but a collection, the mass of the element must be found from the mass density. Assume that there are $n(z)$ molecules in a given fluid element. Say that the average mass of one molecule is $\langle m \rangle$. Then,

$$\rho = n(z)\langle m \rangle,$$

where z is the height-coordinate.

Consider an element with density marked ρ', in a medium of density ρ. In the rest frame of the surrounding gas, the fluid element experiences a gravitational pull relative to the surroundings due to its density difference from the media. Since the densities of the elements are dependent on z, we can use the well-known equation

$$\mathbf{F} = m\mathbf{g} = -\frac{dU}{d\mathbf{x}}, \tag{2.6}$$

to construct the potential energy, U, of a fluid element, due to the local gravitational potential. The relative potential that a fluid element experiences in the reference frame of the surrounding media can be expressed as

$$U = -gV \int_{z_0}^{z} (\rho - \rho')dz',$$

where V is the volume of the fluid element. The element is assumed small enough to be uniform, and the variation of ρ with respect to height is assumed to apply equally to the entire element. We can set $z_0 = 0$ as the point of origin of the fluid element. Then $\rho'_0 = \rho_0$ due to thermodynamic equilibrium. If the fluid element is moved slightly from its natural habitat to a position z, as long as the height increment of the element, Δz, is small, the new density can be expressed as a linearized extension of the previous one:

$$\frac{\partial \rho}{\partial z}|_{surroundings} = \frac{\rho - \rho_0}{\Delta z}, \qquad \frac{\partial \rho}{\partial z}|_{element} = \frac{\rho' - \rho'_0}{\Delta z}.$$

Now let $\Delta z \to z$. In the context of linearization, $\frac{\partial \rho}{\partial z}$ is considered a constant. The relative gravitational potential between the two reference frames is then

$$U = \frac{1}{2} V g z^2 \left(\frac{\partial \rho}{\partial z}|_{element} - \frac{\partial \rho}{\partial z}|_{surroundings}\right). \tag{2.7}$$

Consider a fluid element that performs oscillations of this type in the atmosphere, so that an average potential energy $\langle U \rangle$ can be found. If the oscillations are constrained, the time average of kinetic and potential energy, $\langle T \rangle$ and $\langle U \rangle$ respectively, may be taken as

$$\langle T \rangle = \lim_{\tau \to \infty} \frac{1}{\tau} \int_0^\tau T(t) dt, \qquad \langle U \rangle = \lim_{\tau \to \infty} \frac{1}{\tau} \int_0^\tau U(t) dt,$$

and U is a homogeneous function of the co-ordinates, i.e.

$$U(\alpha \mathbf{r}_1, ... \alpha \mathbf{r}_n) = \alpha^k U(\mathbf{r}_1, ... \mathbf{r}_n).$$

The *virial* theorem then applies (§10 Landau and Lifshitz, 1976), and

$$2\langle T \rangle = k \langle U \rangle, \tag{2.8}$$

where k is called the *degree of homogeneity* of U.

The degree of homogeneity of the potential energy found in eq. (2.7) is $k = 2$. This case is special. Using $k = 2$ in the virial theorem, eq.(2.8), we apply eq.(2.6) to get

$$-\frac{dU(z^2)}{dz} = m\ddot{z} \propto z.$$

The result describes a harmonic oscillator, and is interesting because the period of small oscillations is independent of the amplitude of oscillation. This means that an *angular frequency* of fluid element oscillations may be ascribed to general, thermodynamic conditions, without knowing the exact distance each fluid element moves.

This property of a harmonic oscillator may be shown using *mechanical similarity* principles. In classical mechanics, *Lagrange's equations*

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = 0,$$

with appropriate boundary conditions, completely determine the equations of motion of the classical system. L is the Lagrangian of the system, which in an energy potential that is not explicitly dependent on time gives

$$L = T - U,$$

while q_i are the generalized coordinates of the system. In this case, there is only one coordinate, z, and because it is a *Cartesian* coordinate, the equations of motion are expressed in terms of *mass*,

$$m\ddot{x} = f(x, \dot{x}).$$

Multiplication of the Lagrangian of the system by a constant does not change the equations of motion (§10 Landau and Lifshitz, 1976). One can multiply the space coordinate z and the time coordinate t with constant factors such that

$$z_b = \alpha z_a \qquad t_b = \beta t_a.$$

Since

$$U(z_b) = U(\alpha z_a) = \alpha^k U(z_a),$$

and

$$T(z_b, t_b) = (\frac{\alpha}{\beta})^2 T(z_a, t_a),$$

there is only one way to express

$$L(z_b, t_b) = cL(z_a, t_a)$$

for constant c, the condition for conserving the equation of motion, and this is by setting

$$\beta = \alpha^{1-\frac{k}{2}}.$$

This means that there is a classically deterministic relation between the size of path of the particle, and the corresponding time of motion for the particles to follow the paths. However, setting $k = 2$ yields a time of motion, in this case the period of oscillation, that is independent of the size of the path taken, in this case the amplitude of oscillation.

Note that it is now known that $\langle T \rangle = \langle U \rangle$. The virial theorem in eq.(2.8) only works if the motion takes place in a finite region of space, for finite velocities. Otherwise the mean kinetic energy

$$\langle T \rangle = \lim_{\tau \to \infty} \frac{1}{\tau} \int_0^\tau T(t) dt \tag{2.9}$$

would grow out of bounds, meaning the element would no longer be constrained to the local (effective) potential field. In the current application, this means that the fluid elements would be accelerated out of the reference frame of the local atmospheric medium. Considering that the atmosphere is composed of several species of molecules, each of which can be assumed homogeneous in mass, this type of instability can cause atmospheric layering, where different species separate into different substrate heights. Relatively mild wind conditions are required for plume measurements since the contaminant is carried passively.

The condition for instability may be found. Since $\langle T \rangle$ is proportional to v^2, it must be positive. This means that the condition for breaking the virial theorem eq.

(2.8), and therefore having unconstrained particle oscillations, is that $\langle U \rangle$ becomes negative, i.e.

$$(\frac{\partial \rho}{\partial z}|_{element} - \frac{\partial \rho}{\partial z}|_{surroundings}) < 0, \qquad \frac{\partial \rho}{\partial z}|_{surroundings} > \frac{\partial \rho}{\partial z}|_{element}. \tag{2.10}$$

The equation of motion of a harmonic oscillator is of the form

$$\ddot{z} + \omega_B^2 z = 0.$$

It is related to the potential energy as expressed in eq.2.7 by

$$m\ddot{z} = -\frac{\partial U}{\partial z}, \qquad \ddot{z} = -\frac{1}{2}\frac{\partial}{\partial z}(z^2\omega_B^2),$$

hence the harmonic oscillation has angular frequency

$$\omega_B^2 = \frac{g}{\rho}(\frac{\partial \rho}{\partial z}|_{element} - \frac{\partial \rho}{\partial z}|_{surroundings}).$$

When the system becomes unstable, ω_B is imaginary as expected. This corresponds to exponential, rather than sinusoidal, solutions of the equation of motion. This is called *unstable stratification*. The harmonic angular frequency used in this context is known as the $Brunt - Väisälä$ frequency. See for instance (Yeh and Liu, 1972). This frequency is special, as pointed out previously, because it corresponds to small vibrations, and is therefore independent of the amplitude of oscillations.

An explicit solution of this instability criterion is in order. Consider a 'naive' view of the Troposphere where the primary composition of O_2 and N_2 is simplified to a single diatomic ideal gas. The ideal gas approximation is good for sufficiently rarefied gases, where the intermolecular interactions are negligible. This can be justified loosely on the grounds that the previously assumed incompressibility implies rarefaction, and the chemical composition is uniform. Since the molecular interactions are small, the gas number density can be formulated according to the *Boltzmann distribution*

$$n(\mathbf{r}) = n_0 e^{-U(x,y,z)/k_B T},$$

where n is the number density of molecules (§38 Landau and Lifshitz, 1980). This is transformed into the *barometric formula* by inserting the effective gravitational potential for low altitude variations,

$$n(z) = n_0 e^{-\langle m \rangle g z/k_B T}.$$

The quantity $H_n = \frac{k_B T}{\langle m \rangle g}$ is called the *density scale height*, and differentiating gives

$$\frac{1}{H_n} = -\frac{1}{n}\frac{dn}{dz} = -\frac{1}{\rho}\frac{d\rho}{dz}. \tag{2.11}$$

Figure 2.1: Illustration of unstable stratification with large up and down motions of the plume. These are conditions which should be avoided for LIDAR experiments. The figure is a single frame from a film taken at Risø National Laboratory.

For an *isothermal* ideal gas, $dp = k_B T dn$, and so for an isothermal atmosphere,

$$p = p_0 e^{-z/H_p}.$$

This is the scale where, for an isothermal atmosphere, the pressure has dropped to ~ 0.37 of its original quantity. This scale is approximately 8 km in the Troposphere. Differentiating the above gives

$$\frac{1}{H_p} = -\frac{1}{p}\frac{dp}{dz}. \tag{2.12}$$

The movement of the fluid element from its position of thermodynamic equilibrium to its new position can be assumed fast enough so that no heat was exchanged with the environment, but slow enough so that pressure $p' = p$, so that the process was *adiabatic*. For adiabatic processes,

$$p\rho^\gamma = const,$$

where $\gamma = \frac{f+2}{f} = \frac{C_P}{C_V}$, for f degrees of freedom. Differentiating this expression gives

$$\frac{1}{p}dp = \frac{\gamma}{\rho}dp. \tag{2.13}$$

Then according to eq.(2.13) and eq.(2.12),

$$\frac{\partial\rho}{\partial z}|_{element} = \frac{\rho}{\gamma p}\frac{\partial p}{\partial z} = -\frac{\rho}{\gamma H_p},$$

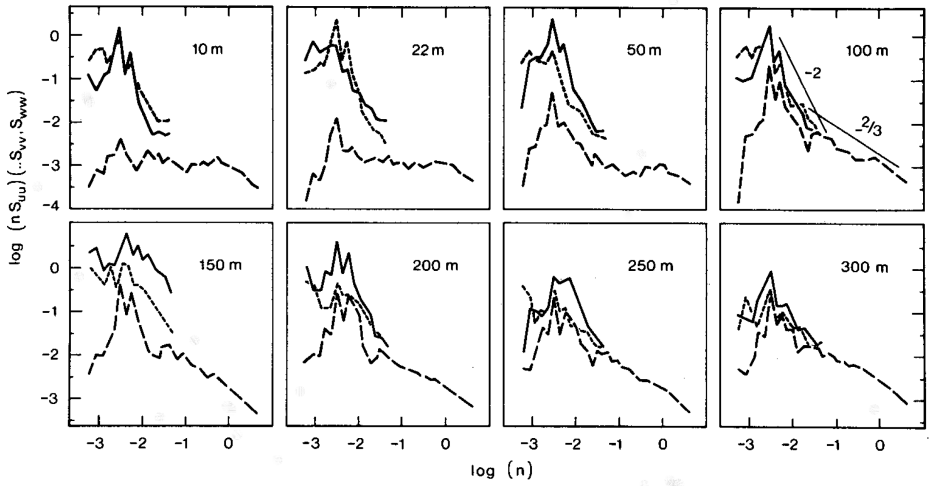


FIG. 28. Power spectra for u, v and w for each height on the tower; u is westerly, v southerly and w vertical (see text for an explanation of the notation). Spectra are plotted as log $n \times S_{uu}(\cdots S_{vv}, S_{ww})$ vs log n (etc.) where S_{uu} is the power of spectral density of u fluctuations and n is circular frequency in Hz: solid line, S_{uu}; short-dashed line, S_{vv}; long-dashed line, S_{ww}.

Figure 2.2: Horizontal spectra at different heights. Taken from (Finnigan et al., 1984), including the original figure caption. The distinct spikes (given by the full lines) at low frequencies originate from gravity waves.

and according to eq.(2.11), the density of the environment between the two heights changes according to

$$\frac{\partial \rho}{\partial z}|_{surroundings} = -\frac{\rho}{H_n}.$$

Hence,

$$\omega_B^2 = g\left(\frac{1}{H_n} - \frac{1}{\gamma H_p}\right), \qquad \langle U \rangle = \frac{1}{2}\rho z^2 g\left(\frac{1}{H_n} - \frac{1}{\gamma H_p}\right),$$

and the instability criterion is

$$\gamma H_p > H_n.$$

Using this criterion, one could theoretically predict the expected occurrence of stratification and subsequently avoid those conditions for making concentration fluctuation measurements. If we include viscosity (Yeh and Liu, 1972) we find the stable solution given above to be damped. As an intermediate case we can find neutrally stable conditions where the growth rate exactly compensates viscous damping, to give undamaged oscillations.

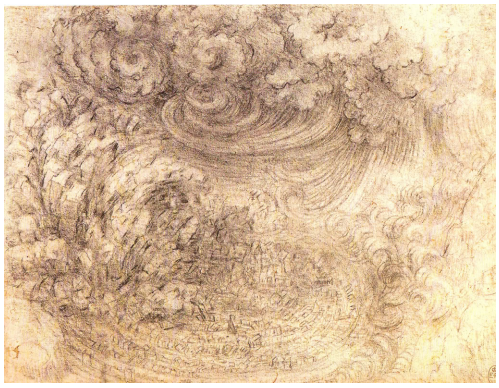
An unstably stratified atmosphere will be characterized by large scale up and down motions, as visualized for instance in fig.2.1. Such conditions are disadvantageous for LIDAR-detection, since the plume can spend significant time intervals at larger altitudes, so that it is not crossed by the beam. The data analyzed in the present thesis refer to stable or neutrally stable conditions (in this sense) where the turbulent motions are primarily in the horizontal direction (Finnigan et al., 1984),

(Einaudi and Finnigan, 1993). This information is not available to us from the MADONA experiments, but fig.2.2 serves as an illustration, also for our case.

By its derivation, the Brunt-Vaisala frequency characterizes large scale, or bulk, oscillations of the entire stratified atmosphere. If finite wavelengths are considered these waves will be called atmospheric gravity waves (Hines, 1960), following their own dispersion relation. Indications of these waves are given by the full lines at low frequencies in fig.2.2.

Chapter 3

Statistical Treatment of Turbulence



3.1 Description of Turbulence

[1] *Similarity* is about finding properties that remain *invariant* when the variables of the system are scaled by a common factor. In fluid dynamics, these more general properties allow us to characterize flow regardless of form. We are therefore interested in *dimensionless* quantities.

Typically given parameters of flow are the size of the media in at least one dimension, on the order of the length scale L, $(\mathcal{O}(L))$, the flow velocity $\mathcal{O}(U)$, and the kinematic viscosity ν. In SI units:

$$L = [m] \qquad U = \left[\frac{m}{s}\right] \qquad \nu = \left[\frac{m^2}{s}\right].$$

Dimensionless combinations of the above are powers of

$$R = \frac{UL}{\nu},$$

called the *Reynolds number*. If the lengths $\mathbf{r}$ and velocities $\mathbf{u}$ are measured in terms of L and U, dimensionless parameters $\frac{\mathbf{r}}{L}$ and $\frac{\mathbf{u}}{U}$ can be introduced, representing measurements in terms of the scales. Since the only dimensionless combination

[1] Picture above from: http://i621.photobucket.com/albums/tt294/toddsiler1/LeonardodaVincisStudyofDeluge.jpg

is Reynolds' number R,

$$\mathbf{u} = U \cdot f\left(\frac{\mathbf{r}}{L}, R\right),$$

for some unknown function f. Similarly the pressure can be described in terms of Reynolds' number as

$$p = \rho U^2 \cdot g\left(\frac{\mathbf{r}}{L}, R\right),$$

for some other function g (Landau and Lifshitz, 1989).

In the incompressible Navier-Stokes eq.(2.5), the term $(\mathbf{u} \cdot \nabla)\mathbf{u}$ represents inertial forces, while $\nu\nabla^2\mathbf{u}$ represents viscous forces. We estimate these quantities as

$$(\mathbf{u} \cdot \nabla)\mathbf{u} \sim \frac{U^2}{L} \qquad \nu\nabla^2\mathbf{u} \sim \frac{\nu U}{L^2}.$$

As a result,

$$\frac{|(\mathbf{u} \cdot \nabla)\mathbf{u}|}{|\nu\nabla^2\mathbf{u}|} \sim \frac{U^2/L}{\nu U/L^2} = \frac{UL}{\nu} = R.$$

Evidently, R signifies the ratio between inertial and viscous forces in the flow. Turbulence typically occurs at very high Reynolds' numbers, and is therefore characterized by strong non-linearities. We can also assign Reynolds numbers for *local* length scales. For instance, the smaller eddies in a turbulent field of e.g. $R \geq 5000$ are dominated by *both* viscous and inertial forces. Then we assign $R_{local} \sim 1 \neq R$. In this way, we use Reynolds's numbers to characterize *scales* of flow. Note that Reynolds's number for a turbulent flow also approximates the ratio between the molecular and turbulent time scales, $R \sim \frac{T_{mol}}{T_{turb}}$ (Tennekes and Lumley, 1972).

Since as far back as Boussinesq (Feriet and Pai, 1954), it was held as impractical to follow *each* heavily fluctuating fluid element. Instead, *statistical ensembles* of fluid elements are constructed, approximated by the time average, in the sense of eq.(2.9). Similarity solutions based on mean values can be constructed.

A turbulent velocity field can be considered a random function oscillating in some sense around a mean (Tennekes and Lumley, 1972):

$$\mathbf{u} = \langle \mathbf{u} \rangle + \tilde{\mathbf{u}},$$

where the fluctuation $\tilde{\mathbf{u}}$ is considered the result of superposition of *turbulent eddies*. The fluctuation $\tilde{\mathbf{u}}_\lambda$ for arbitrary length scale λ may be taken as the velocity of turbulent eddies at that scale. These eddies are the rotations and reverse currents created in unstable flow, for instance in high R flow past material obstacles. The largest eddies are created first. As R increases, subsequently smaller eddies are created. The largest eddy sizes are denoted $\mathcal{O}(l)$, where l is called the *external* scale, and correspond to small frequency "vibrations". They are identified with a scale l where the fluctuation $\tilde{\mathbf{u}}$ is relatively large. Most of the kinetic energy of

the fluid is retained in these eddies, and this scale is therefore called the *energy range*. The smallest eddies are on the scale where viscous effects are important. The kinematic viscosity ν is scale-independent. Since R_{local} estimates the relative importance of inertial against viscous effects at the length scale λ, corresponding to local velocity fluctuations $\tilde{u}$, if

$$R_\lambda \sim \frac{\tilde{u}_\lambda \lambda}{\nu} \sim 1$$

or lower, viscous effects are important. This *internal* scale corresponds to the smallest eddies, and is called the *dissipation range*. The kinetic energy from larger scales is here transferred to heat. The range between the energy and dissipation scales is called the *inertial* range. Fully developed turbulence occurs when a continuous range of eddies have been created here. The *Richardson cascade* (Richardson, 2007) (Richardson, 1926) postulates that energy passes with little dissipation within the inertial range, from small to large frequencies, which leads to the conclusion that similarity arguments may be employed to create scaling laws within this range.

Viscosity is not important in the *energy* and *inertial* ranges. Dimensional arguments may be used, since the large-eddy *energy dissipation*

$$\epsilon = \left[\frac{J}{kg \cdot s}\right]$$

can only be constructed through

$$\tilde{u} = \left[\frac{m}{s}\right] \qquad l = [m] \qquad \rho = \left[\frac{kg}{m^3}\right].$$

The only dimensionally valid relation is

$$\epsilon \propto \frac{\tilde{u}^3}{l}.$$

Since *Richardson cascade* assumes that the *energy dissipation* ϵ is constant above the *dissipation range*, this means that the eddy velocities are related to their respective length scale λ within the *inertial range* by Kolmogorov's and Obukhov's law (Kolmogorov, 1991):

$$\tilde{u}_\lambda \propto (\epsilon \cdot \lambda)^{\frac{1}{3}}.$$

The invariance of the energy dissipation can then be used to relate the two scales, hence:

$$\tilde{u}_\lambda \propto \tilde{u} \cdot \left(\frac{\lambda}{l}\right)^{\frac{1}{3}}, \tag{3.1}$$

which illustrates the similarity properties in the inertial range. This can be made obvious by considering the 'translation' in Reynolds' number between scales, using eq.(3.1):

$$R_\lambda \sim \frac{\tilde{u}_\lambda \lambda}{\nu} \sim \frac{\tilde{u} \cdot \lambda^{\frac{4}{3}}}{\nu \cdot l^{\frac{1}{3}}} \sim R \cdot \left(\frac{\lambda}{l}\right)^{\frac{4}{3}}.$$

For instance, let the internal scale where $R_{local} \sim 1$ be denoted λ_0.

From eq. (3.1),

$$l \left(\frac{\tilde{u}_{\lambda_0}}{\tilde{u}}\right)^3 \propto \lambda_0,$$

viz

$$\lambda_0 \propto \frac{l}{R^{\frac{3}{4}}} \qquad \tilde{u}_{\lambda_0} \propto \frac{\tilde{u}}{R^{\frac{1}{4}}}.$$

We want to know whether similarity theory is applicable to *fixed source* diffusion of a contaminant. In terms of similarity theory, the question is whether the dispersion $\langle r^2 \rangle$ is mainly dependent on the *inertial* eddies, or the *large* eddies. Consider the time variation of the one-dimensional case:

$$\frac{d\langle r^2 \rangle}{dt} = 2\langle r(t) \cdot u(t) \rangle = 2 \int_0^t \langle u(t') \cdot u(t) \rangle dt'.$$

The *correlation* $\int_0^t \langle u(t') \cdot u(t) \rangle$ is mainly dependent on $(t - t')$ rather than t or t' separately (Batchelor, 1950), and can therefore be written $S(t - t')$, such that

$$\frac{d\langle r^2 \rangle}{dt} = 2 \int_0^t S(\tau) d\tau.$$

When $\tau = 0$, $S(\tau) = \langle u^2 \rangle$ for a stationary process, and is not strongly dependent on the inertial eddies. As τ increases, the large eddies, which are dominated by the boundary conditions, are first excited (Batchelor, 1950). Thus similarity theory can not be used until the turbulence is fully developed. When this happens is hard to estimate. However, the diffusion process for very large τ tends towards classical diffusion (sec.4.2), which we can understand and simulate through random walk models (sec.4.1).

The second option is to use a completely statistical approach. It should be specified that when the dispersion $\langle r^2 \rangle$ is expressed as an expectation value, this is theoretically sampled from an *ensemble* of individual realizations under the same statistical conditions. To approach this in an experimental setting, the ensemble must be equated with the sample over space or time. Hence in practice, the ensemble integral is often the *experimental time average* over time T (Feriet and Pai, 1954). Let ω stand for samples in the measure space. Then

$$\langle f(x, t, \omega) \rangle = \int_\omega f(x, t, \omega) d\omega \sim \frac{1}{2T} \int_{-T}^{T} f(x_0, t) dt.$$

This depends on how steady the measurement conditions are, as well as whether the time average represents an ensemble average. The first is decided practically, since a long measurement time may lead to uncontrolled fluctuations due to factors that vary in the long time scale. *Ergodicity* is the hypothesis that an observation over a long time is an accurate depiction of the *phase space* of the system, which is where permutations of fluid elements's state (q, p) occur as time processes. For thermodynamics, this can be shown by Liouville's theorem, but no complete proof exists for turbulent fluids. Nevertheless it is a another assumption which is made in the statistical treatment of the subject.

The statistical *moments*, e.g. of the field velocity u, are

$$\langle u^n \rangle = \frac{1}{2T} \int_{-T}^{T} u^n(t) dt.$$

The first moment $\langle u \rangle$ is the mean flow velocity, and as has been shown above, the turbulent flow field can be deconstructed into a mean and a fluctuating part. The second moment $\langle u^2 \rangle$ is proportional to the mean energy density of the flow. Thus, important physical characteristics of the flow can be gathered through simple statistical analysis.

When we want to generalize statistical data to account for different boundary and surface conditions, the underlying ensemble can be expressed according to a (normalized) *probability density function* (PDF) $p(x)$, so that the moments of some variable x can be found through

$$\langle x^n \rangle = \int_{-\infty}^{\infty} x p(x) dx.$$

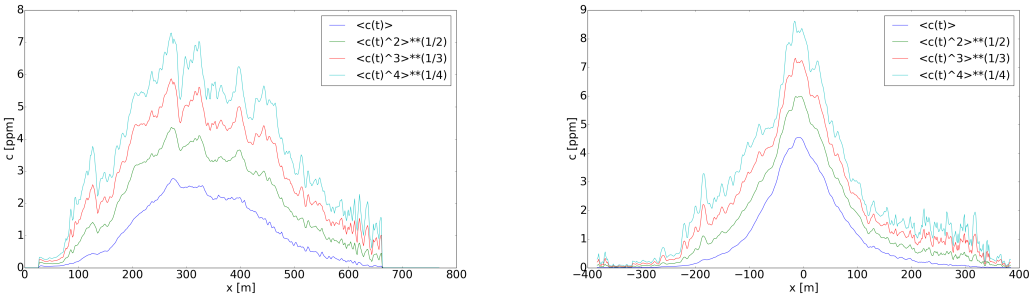
In practice, this typically means binning frequency of occurrence of variable x within $x_i \pm \Delta x$ (sec.6.1).

3.2 Statistical Moment Relations

For classical diffusion, the $\{0, 1, 2\}^{th}$ moments (blue and green in figs.3.1a, 3.1b) are sufficient for a macroscopic description (sec.4.1). This is not the case for turbulent diffusion, where the presence of extreme fluctuations and a lack of symmetry about the mean have to be accounted for. The first is related to the *kurtosis*, the second to the *skewness* of the distribution.

For a variable c with mean μ, the skewness is defined as the variance-normalized third moment:

$$\alpha_3 = \frac{\langle (c - \mu)^3 \rangle}{[\langle (c - \mu)^2 \rangle]^{\frac{3}{2}}},$$



(a) Fixed Frame (b) Center of Mass Frame

Figure 3.1: The first 4 moments of the data set K, Day 265 of measurement. The set has been normalized according to Miyake concentrations.

while the corresponding kurtosis is defined as:

$$\alpha_4 = \frac{\langle (c-\mu)^4 \rangle}{[\langle (c-\mu)^2 \rangle]^2}.$$

The normalization by variance ensures that these quantities are dimensionless.

There exists a lowest bound on the kurtosis, relating it to the skewness, courtesy of Karl Pearson (Pearson, 1916). It can be found by constructing the quadratic form (Gnedenko, 1997) in the continuous distribution function (CDF) $F(c)$. For variable c to the point c_0,

$$F(c_0) = P(c \leq c_0),$$

where $P(c \leq c_0)$ is the total probability that $c \leq c_0$. The moments are then defined in terms of the CDF as

$$\langle c^n \rangle = \int_{-\infty}^{c_0} c^n dF(c).$$

If $\nu_i(a)$ is the i^{th} moment around a:

$$J_n = \int \left[\sum_{k=0}^{n} t_k (c-a)^k) \right]^2 dF(c) = \sum_{j=0}^{n} \sum_{k=0}^{n} \nu_{k+j}(a) t_k t_j \geq 0,$$

for any combination of constants $\{t_k\}$. The first $2n$ moments must therefore satisfy, for the determinant,

$$\begin{vmatrix} \nu_0(a) & \nu_1(a) & \dots & \nu_k(a) \\ \nu_1(a) & \nu_2(a) & \dots & \nu_{k+1} \\ \dots & \dots & \dots & \dots \\ \nu_k(a) & \nu_{k+1}(a) & \dots & \nu_{2k} \end{vmatrix} \geq 0, \quad (k = 0, 1, 2, ..., n).$$

Consider this inequality for $k = 2$. The first three moments can be trivialized without loss of generality (Wilkins, 1944), by assuming

$$\nu_0 = 1 \qquad \nu_1 = 0, \qquad \nu_2 = 1.$$

This corresponds to a centralized and normalized distribution, where the variance has become reduced so that $\nu_3 = \alpha_3$ and $\nu_4 = \alpha_4$. The resulting determinantal inequality relates kurtosis and skewness:

$$\begin{vmatrix} 1 & 0 & 1 \\ 0 & 1 & \alpha_3 \\ 1 & \alpha_3 & \alpha_4 \end{vmatrix} \geq 0.$$

This expands to

$$\alpha_4 \geq \alpha_3^2 + 1. \tag{3.2}$$

Imagine that the initial release was given time to stabilize, so that the entire concentration distribution was flat. At the moment of release, we can describe the concentration distribution PDF as consisting of a delta spike representing the 0 concentration points outside of the release area, as well as a delta spike representing the constant concentration points within the release tube. This model, sometimes called a "top hat" model, is mathematically expressed as

$$p(c) = (1 - \beta)\delta(c) + \beta\delta(c - c_0).$$

It can be shown, see for instance (Bergsaker, 2012), that the skewness-kurtosis relation for this particular distribution is exactly

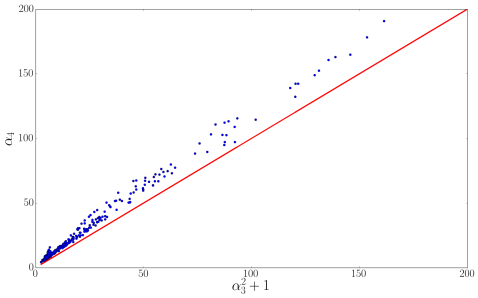
$$\alpha_4 = \alpha_3^2 + 1,$$

which means that if we assume that the initial distribution followed this model, any subsequent deviation of the skewness-kurtosis relations found from the MADONA data sets can be seen as a result of the diffusion of the distribution away from the steady state case.

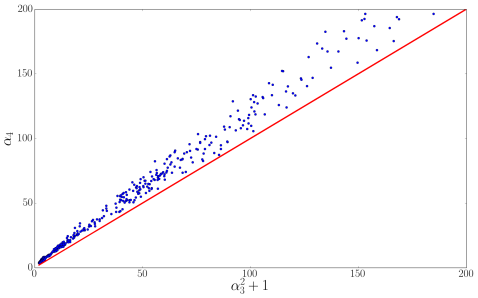
It is well known, e.g.(Squires, 2001), that the estimate of standard deviation from discrete measurements is performed as

$$\sigma^2 \approx \frac{1}{n-1} \sum_{i=1}^{n} (c_i - \mu)^2.$$

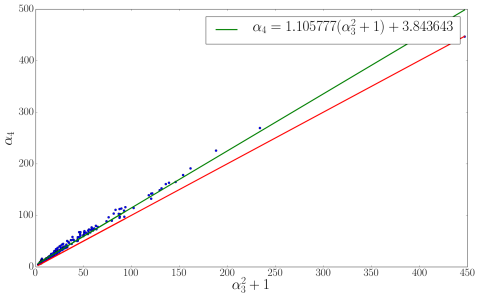
However, in this thesis we average over the hundreds of samples measured over time for each data set. For instance, there are 449 data points for each fixed



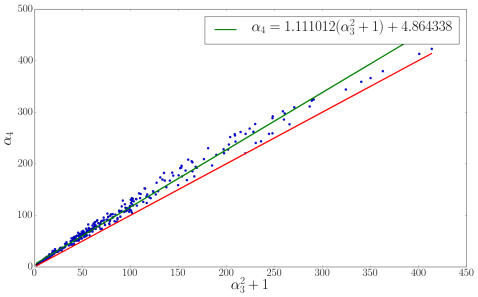
(a) Fixed Frame



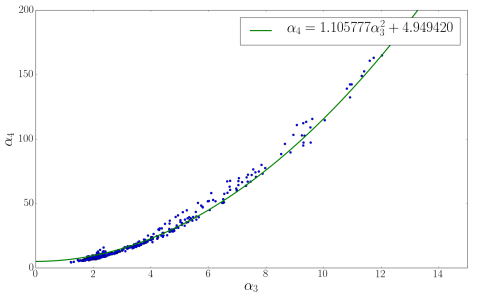
(b) Center of Mass Frame



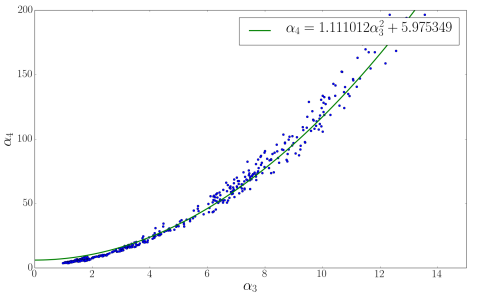
(c) FF: Interpolation reads $\alpha_4 = 1.105777(\alpha_3^2 + 1) + 3.843643$.



(d) CM: Interpolation reads $\alpha_4 = 1.111012(\alpha_3^2 + 1) + 4.864338$.



(e) FF: Interpolation reads $\alpha_4 = 1.105777\alpha_3^2 + 4.949420$.



(f) CM: Interpolation reads $\alpha_4 = 1.111012\alpha_3^2 + 5.975349$.

Figure 3.2: *mad21K*: Skewness-Kurtosis Relations. The red line in figs.3.2a and3.2b corresponds to $\alpha_4 = \alpha_3^2 + 1$. We can see that no data point goes under this line for both reference frames. Each data point here is the Skewness-Kurtosis relation at a single position, averaged over time. The models shown in figs.3.2e and3.2f are from the interpolations in figs.3.2c and3.2d, shown on $[\alpha_3\hat{x}, \alpha_4\hat{y}]$ axes.

location in *mad21K*. This means that we can let $n-1 \approx n$ in the context of our calculations. We use the expression

$$\sigma^2 \approx \left(\frac{1}{n}\sum_{i=1}^{n} c_i^2\right) - \mu^2.$$

The mean is estimated as

$$\mu \approx \frac{1}{n}\sum_{i=1}^{n} c_i,$$

and skewness and kurtosis are, respectively,

$$\alpha_3 = \frac{1}{n}\frac{\sum_{i=1}^{n}(c_i-\mu)^3}{\sigma^3}, \qquad \alpha_4 = \frac{1}{n}\frac{\sum_{i=1}^{n}(c_i-\mu)^4}{\sigma^4}. \tag{3.3}$$

The expression for the skewness can be made computationally cheaper by inserting for μ and σ. Notice that

$$\begin{aligned}\langle (c-\mu)^3\rangle &= \left\langle c^3 - 2c^2\mu + c\mu^2 - c^2\mu + 2c\mu^2 - \mu^3\right\rangle \\ &= \langle c^3\rangle - 3\langle c^2\rangle\mu + 3\mu^3 - \mu^3 \\ &= \langle c^3\rangle - \mu^3 - 3\mu\sigma^2.\end{aligned}$$

This means that the skewness can be expressed as

$$\alpha_3 = \frac{\langle c^3\rangle - \mu^3 - 3\mu\sigma^2}{\sigma^3}$$

$$\alpha_3 \approx \frac{\sum_{i=1}^{n} c_i^3}{\sigma^3} - \left(\frac{\mu}{\sigma}\right)^3 - \frac{3\mu}{\sigma}. \tag{3.4}$$

The expression for kurtosis can be simplified to

$$\alpha_4 = \frac{\langle c^4\rangle + 3\mu(\langle c^2\rangle\mu - \langle c^3\rangle) - \mu^4}{\sigma^4} - \frac{\mu\alpha_3}{\sigma}.$$

Since we are not saving the moments of c in this particular context, we will use eq.(3.3) to calculate the kurtosis.

Eqs.(3.4) and (3.3) were used to estimate the skewness-kurtosis relation from eq.(3.2). The results are plotted as $[(\alpha_3^2+1)\hat{x}, \alpha_4\hat{y}]$ in figs. 3.2a and 3.2b, for fixed frame and center of mass frame, respectively. The data points never fall below the straight line, as theoretically predicted.

We are now interested in the least squares linear interpolation through all the points which gives

$$\alpha_4 \approx a(\alpha_3^2+1) + b.$$

This expression gives a quantitative idea of the deviation away from the theoretical initial distribution expressed in the top hat model. The interpolation can be seen in figs. 3.2c and 3.2d. The model of the skewness kurtosis relation becomes

$$\alpha_4 \approx a\alpha_3^2 + (a+b). \tag{3.5}$$

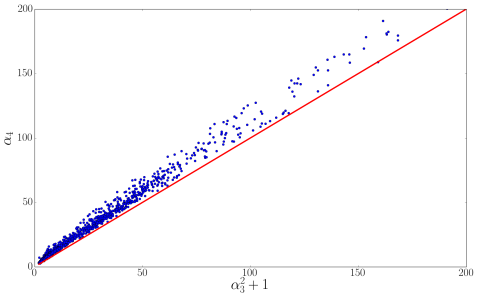
These models are shown in figs. 3.2e and 3.2f. Note the lack of difference that the conversion from FF to CM frame actually makes. We found that a relation of the form shown in eq.(3.5) gives a surprisingly robust fit to the data, even when we mix results from 5 different experimental conditions, as shown in figs.3.3a, 3.3b. The similarity of the α_4 - α_3 relation when comparing absolute and moving frame data has been observed for some different data sets in for instance(Jørgensen, Mikkelsen and Pécseli, 2010). It is also interesting that the best fit to the data is not too different from the analytical limit found in eq.(3.2). I know of no analytical basis for a universal relation between skewness and kurtosis, but many data support expressions like eq.(3.5).

Another point of interest is the relation between the skewness and the variance. This has been shown for the top hat model (Bergsaker, 2012) to correspond to

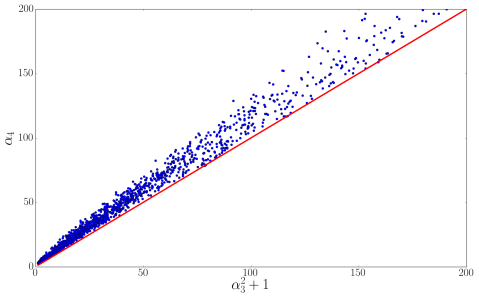
$$\frac{\sigma^2}{\mu^2} = 1 + \frac{1}{2}\left(\alpha_3^2 + \alpha_3\sqrt{\alpha_3^2+4}\right). \tag{3.6}$$

This assumes $\mu \neq 0$, and so does not apply where the concentration is 0 throughout. It would therefore be interesting to test this relation in the CM frame, and specifically note those points that are close to the CM itself. We know from fig.3.5a that these are points with low skewness values. Setting an artificial sampling cap at $\alpha_3 = 5$, we traced a linear and quadratic interpolation over the data in fig.3.3f.

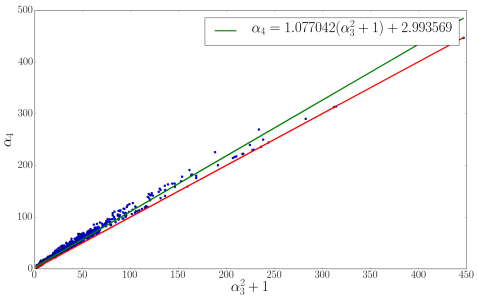
The result can be seen in fig.3.3e. This relation does not have any rigorously defined boundary as the Skewness Kurtosis relation. Nevertheless, we see that no point crosses the boundary traced by the top hat model, despite significant spread at high skewness. This could suggest a similar theoretical cap as the Skewness Kurtosis relation found in eq.3.2. Also, in this case, we find a seemingly systematic relation of σ^2 for a varying α_3, at least for positions close to the CM. Again, no analytical basis exists for such a relation, and in this case, some observations in plasma physics have given counter-examples (Bergsaker, 2012). Nevertheless, the fact that the top hat model forms a limit here, as in the skewness-kurtosis comparison, may be due to an empirical factor, for instance that the concentration can only grow more disordered if we assume that the release corresponds to a top hat distribution, or a constraint due to the fact that the concentration is never negative, because Bergsaker (2012) allowed for negative numbers for the variable in the top hat model.



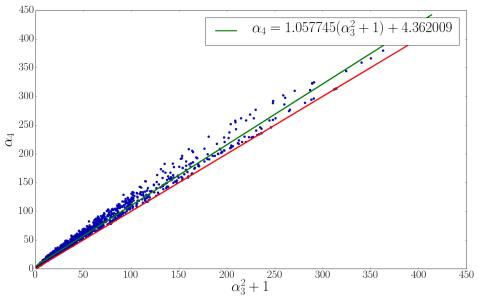
(a) Fixed Frame



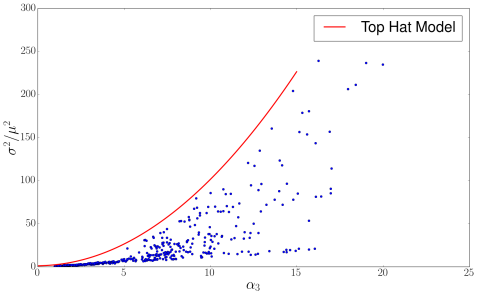
(b) Center of Mass Frame



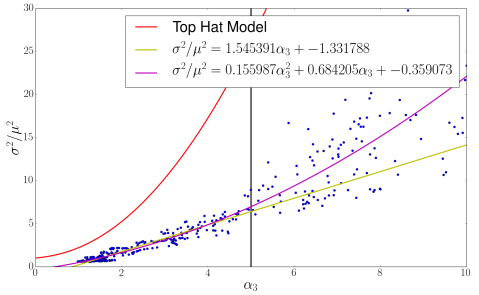
(c) FF: Interpolation reads $\alpha_4 = 1.077042(\alpha_3^2 + 1) + 2.993569$.



(d) CM: Interpolation reads $\alpha_4 = 1.057745(\alpha_3^2 + 1) + 4.362009$.

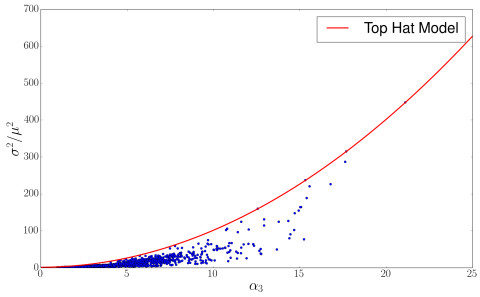


(e) CM: $\frac{\sigma^2}{\mu^2}$ plotted against α_3. Red line is the expected relation according to top hat model.

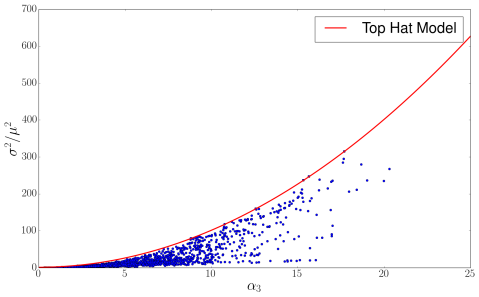


(f) CM: Linear and Quadratic fit for fig.3.3e. Linear fit reads $\sigma^2/\mu^2 = 1.545\alpha_3 - 1.332$. Quadratic Fit reads $\sigma^2/\mu^2 = 0.1560\alpha_3^2 + 0.6842\alpha_3 - 0.3591$. The black line is the truncation used for fit sampling.

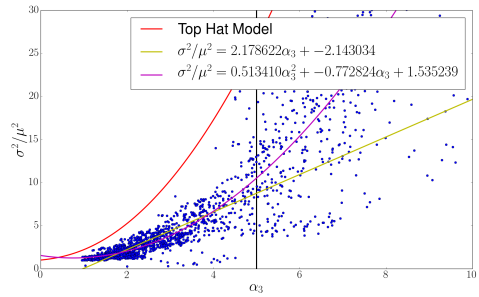
Figure 3.3: *mad21K*: Skewness-Kurtosis Relations for all 5 data sets: *mad21K, mad21H, mad21G, mad21F, mad15J*. Figs.3.3e and3.3f show the Skewness-Variance Relation for *mad21K*. The scatter in the data points gives a measure of the uncertainty on the estimated curve fits.



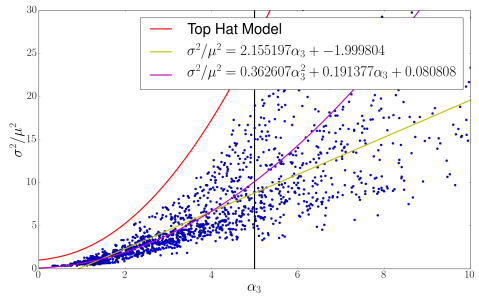
(a) FF.



(b) CM.



(c) FF: interpolated up to $\alpha_3 = 5$. Linear fit reads $\sigma^2/\mu^2 = 2.179\alpha_3 - 2.143$. Quadratic fit reads $\sigma^2/\mu^2 = 0.5134\alpha_3^2 - 0.7728\alpha_3 + 1.535$.



(d) CM: interpolated up to $\alpha_3 = 5$. Linear fit reads $\sigma^2/\mu^2 = 2.155\alpha_3 - 2.000$. Quadratic fit reads $\sigma^2/\mu^2 = 0.3626\alpha_3^2 + 0.1914\alpha_3 + 0.08080$.

Figure 3.4: Variance-skewness comparison with top hat model for the 5 data sets *mad15J, mad21F, mad21G, mad21H, mad21K*. The scatter in the data points gives a measure of the uncertainty on the estimated curve fits also here.

Table 3.1

	Red	Yellow	Green	Blue	Black
x-R [m]	[-82.5,84]	[-157.7,-84] [85.5, 159]	[-232.5, -159] [160.5, 234]	[-307.5, -234] [235.5, 309]	[-382.5, -309] [310.5, 384]
a	1.627	1.372	1.184	1.237	1.048
b	0.6433	0.6757	3.328	1.346	11.52

The top hat model in eq.3.6 is shown to hold as a limit, in figs.3.4a and 3.4b, also for a superposition of all data from the five MADONA data sets we use throughout this thesis, in CM frame and FF.

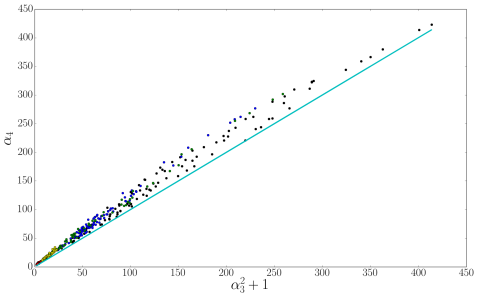
The last point of interest concerning the skewness-kurtosis relation in the MADONA data sets that we explore concerns the relative deviation from the exact relation in eq.(3.2) of points identified by their position from the center of mass. We explore this in CM frame for the data set *mad21K* in figs.3.5a-3.5f. Because we study a single data set, we can not separate the data into too small portions based on position, as we would lose statistical significance. A good selection was found to use groups of 100 data points, separated into 5 color coded subsets. We interpolated a least squares line for each subset, and compared them in table 3.1. a and b here are the interpolation coefficients defined in eq.(3.5). The three possibilities are:

(i) The relation is statistically scale invariant with distance from the CM. This suggests that we can perform the analysis on the 5 data sets superposed.

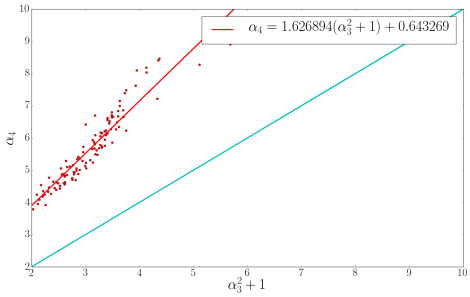
(ii) The positions closer to the CM exhibit a closer fit to eq.(3.2) than the flanks because they have been "mixed" less by the atmosphere.

(iii) The flanks exhibit better fit to eq.(3.2) than the CM. This might suggest that the flanks have concentrations which are more "settled" according to the "top hat" model. This could also suggest that the "top hat" model is a poor assumption for the initial state of release, since the plume was a mix of several compounds.

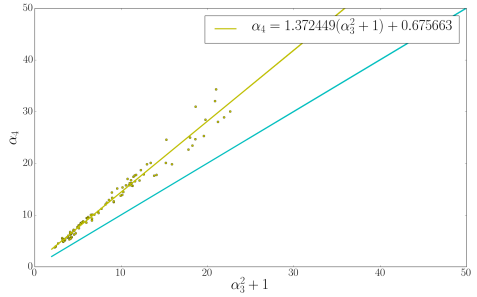
Table 3.1 shows that, apart from the *blue* data points, the rest seem to suggest option (iii). We also see from fig.3.5a that the data points further away from the CM have both larger skewness *and* kurtosis. The larger deviations in b as we move away from the CM are likely due to this larger scale. This could be used to qualitatively estimate the distance away from the CM of measurements made by continuous point measurements under steady conditions. However, such a conclusion should be corroborated by a simultaneous measurement of LIDAR and typical point measurement tools for pollution, and is therefore outside the scope of this work.



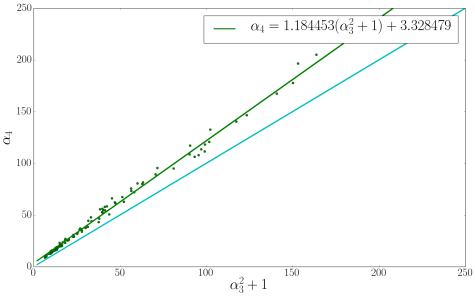
(a) All color coded points for *mad21K*. Note that the position of the points clearly differ by their distance from the CM. However, it is not clear whether there is a difference of gradient in addition to scale.



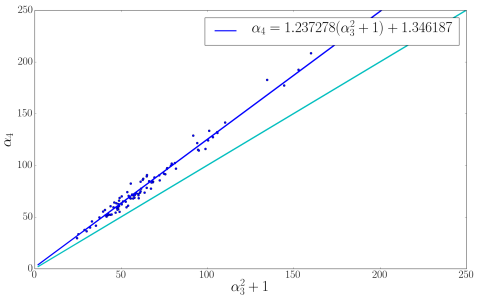
(b) Red line linear fit: $\alpha_4 = 1.627(\alpha_3^2 + 1) + 0.6433$.



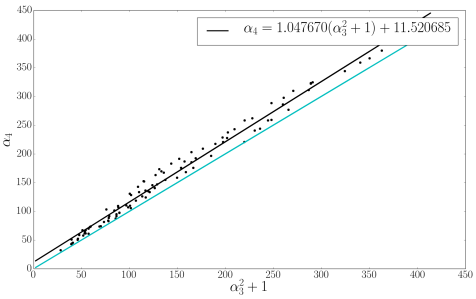
(c) Yellow line linear fit: $\alpha_4 = 1.372(\alpha_3^2 + 1) + 0.6757$.



(d) Green line linear fit: $\alpha_4 = 1.184(\alpha_3^2 + 1) + 3.328$.



(e) Blue line linear fit: $\alpha_4 = 1.237(\alpha_3^2 + 1) + 1.346$.



(f) Black line linear fit: $\alpha_4 = 1.048(\alpha_3^2 + 1) + 11.52$.

Figure 3.5: *mad21K*: CM frame. Fig.3.5a shows $\alpha_3^2 + 1$, α_4 relation in color code. See table 3.1 for numerical positions of the color codes. Figs.3.5b -3.5f show the same relation for distances away from the center of mass in descending order, left to right. With the exception of the blue line, the gradient is decreasing as we move away from the CM. The line in *cyan* is $\alpha_4 = \alpha_3^2 + 1$. The scatter in the data points gives a measure of the uncertainty on the estimated curve fits also here.

3.3 The Logarithmic Boundary Layer

Consider the boundary condition where the fluid molecules near a solid "wall" get "stuck" due to adhesive forces. This is the *no-slip* condition. The fluid molecules all feel cohesive forces, so the neighboring molecules tend to be slowed down by the "stuck" molecules. In the continuum approximation, there is a continuous decrease in the velocity field as one moves from the main flow to the boundary, where the flow is still. This entire volume is called the *boundary layer*.

Our interest lies in the pollutant distribution over time in the local environment. The near-ground surface of the Earth can be considered as a giant boundary layer, and different geographical locations can be associated with different values of a roughness parameter d. The surface is coated with varied terrain, and especially urban sites tend to have dense protrusions in the form of buildings. It therefore pays to have a means of relating the flow characteristics with the roughness parameter.

The flow in the non-viscous sublayer of the boundary layer can be worked out in a way analogous to the Richardson cascade.The mean value of the momentum flux density tensor Π_{xy} will be called σ. It is known as the Reynolds stress tensor, and originates from the velocity gradient $\frac{du}{dy}$. Assuming σ to be invariant with respect to y within this sublayer, dimensional arguments give

$$\frac{du}{dy} \propto \sqrt{\frac{\sigma}{\rho y^2}}.$$

Defining the proportionality factor as the von Kármán constant κ, and relating the momentum flux to its contributing velocity fluctuations $v_\star$,

$$\sigma = \rho v_\star^2, \tag{3.7}$$

$$\frac{du}{dy} = \frac{v_\star}{\kappa y}$$

(Landau and Lifshitz, 1989). This is readily solved, and

$$u = \frac{v_\star}{\kappa}(\ln y + c) \tag{3.8}$$

is the *logarithmic velocity profile*. Note that eq.(3.8) implies that u keeps increasing for arbitrarily large heights y. In reality, this is correct only for altitudes up to a few hundred meters. It has been speculated that the Weibull distribution offers a good experimental fit for higher altitudes.

The constant c can be found from the usual assumption that viscosity becomes important as $R \sim 1$. Denoting this distance from the boundary as y_0,

$$R \sim \frac{v_\star y_0}{\nu} \sim 1,$$

$$y_0 \sim \frac{\nu}{v_\star}.$$

The viscous sublayer extends from y_0 to 0, and is very small. Thus to a good approximation, one could let u approach 0 at the transition:

$$u = \frac{v_\star}{\kappa}\ln\left(\frac{y}{y_0}\right) = \frac{v_\star}{\kappa}\ln\left(\frac{yv_\star}{\nu}\right). \tag{3.9}$$

For rough surfaces, the velocity profile in eq.(3.9) is modified by the roughness coefficient d, since this is now the distance where the velocity becomes small. Then,

$$u = \frac{v_\star}{\kappa}\ln\left(\frac{y}{d}\right).$$

The energy flux per mass, ϵ, can be found by letting $\hat{x}$ be the flow direction, so $v_x = u + \tilde{v}_x$, $v_y = \tilde{v}_y$, $v_z = \tilde{v}_z$. Now the Reynolds stress tensor (Reynolds, 1895) is defined

$$\sigma = \rho\langle v_i v_j\rangle.$$

It has been previously established that the main momentum transfer is due to shear stresses $i \neq j$. Consider the $\hat{y}$ energy flux density q:

$$\langle q\rangle = \langle(p + \frac{1}{2}\rho v^2)v_y\rangle.$$

According to eq.(3.9), the fluctuation terms are logarithmically small compared to the main flow u, so to a good approximation,

$$\langle q\rangle = \rho u\langle\tilde{v}_x\tilde{v}_y\rangle.$$

Because the mean stress is invariant towards the boundary within the non-viscous sublayer,

$$\frac{1}{\rho}\frac{d\langle q\rangle}{dy} = \sigma\frac{du}{dy},$$

and using eq.(3.7) and eq.(3.9), and defining $\epsilon = \frac{1}{\rho}\frac{d\langle q\rangle}{dy}$,

$$\epsilon = \frac{v_\star^3}{\kappa y}.$$

ϵ is the mean energy flux per mass, like in the Richardson cascade. The form is analogous to $\epsilon \propto \frac{\tilde{\mathbf{u}}^3}{l}$. This should not be strange, since this was the only way to dimensionally construct the energy flux in the absence of viscosity. A similar logarithmic relation between the main flow and the local fluctuations in the inertial range of turbulent flow would analogously take the form

$$u \sim \tilde{u}\ln\left(\frac{l\tilde{u}}{\nu}\right) \qquad u \sim \tilde{u}\ln(R_{\text{local}}). \tag{3.10}$$

The expressions in eqs.(3.8) and (3.10) illustrate the nature of the inhomogeneity encountered in boundary layers, and in the MADONA experiment in particular. By pointing the LIDAR in a horizontal direction, as done in this experiment, the conditions will remain homogeneous along the beam direction. Changes in turbulence conditions that would result by taking the horizontal beam at different altitudes are illustrated in fig.2.2. These data are taken for different physical conditions, but the basic features will apply also to the MADONA experiments.

Chapter 4

Atmospheric Diffusion

4.1 Classical Diffusion

[1] When a dense tracer fluid is slowly injected horizontally into an unmarked, less dense fluid, the marked fluid will slowly dissolve into the unmarked. It is important that this happens under symmetrical, steady forces, as manifested in a homogeneous and isotropic laminar velocity field. The system tends towards eradicating the inhomogeneity in density, since the denser fluid has more particles traveling towards the unmarked fluid.

In seeking a quantitative description of the phenomena, the mathematical analogy of the *random walk* is convenient for later expansion, because it explains the macroscopic dynamics through statistical modeling and emphasizes the importance of the moment distributions of the underlying probability density function.

We consider here an *unobstructed* one dimensional random walk, so that each walker we release on a grid discretized with steps associated with a transition probability density function χ with equal probability density for left or right steps. The distribution of "walkers" will then be defined by its tendency to distribute away from large distributions, just as in the case of classical diffusion. The theoretical discussion below is partly based on the discussion in (Sethna, 2006).

For simplicity, let $\chi(z)$ be normalized and centered, and let l_0 be its standard deviation. The standard deviation is constant for a *stationary process*, which is true because the forces were assumed steady. The first three moments of the

[1] Picture above from http://shibiaoxu.net/wp-content/uploads/2013/04/InkDiffusion.png

variable z become:

$$\langle z^0 \rangle = \int_{-\infty}^{\infty} \chi(z) dz = 1, \tag{4.1}$$

$$\langle z^1 \rangle = \int_{-\infty}^{\infty} z\chi(z) dz = 0, \tag{4.2}$$

$$\langle z^2 \rangle = \int_{-\infty}^{\infty} z^2\chi(z) dz = l_0^2. \tag{4.3}$$

The conditional probability distribution over time and space, $\rho(x, t + \Delta t|x', t)$, is related multiplicatively to any previous state because the steps of a free random walk are by definition independent. The probability density of a single step between x and x' is the transition probability $\chi(x - x')$:

$$\rho(x, t + \Delta t|x', t) = \rho(x', t)\chi(x - x').$$

If one accounts for all origins, the distribution $\rho(x, t + \Delta t)$ becomes the probability density of any time increment. Integrating over all paths in one spatial dimension:

$$\rho(x, t + \Delta t) = \int_{-\infty}^{\infty} \rho(x', t)\chi(x - x')dx'.$$

By changing variables $z = x - x'$, ρ can be expressed in the form of the moments above:

$$\rho(x, t + \Delta t) = \int_{-\infty}^{\infty} \rho(x - z, t)\chi(z)dz.$$

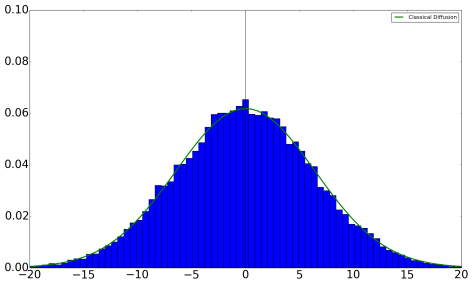
By the assumption of uninhibited random walk, the probability distribution of the particles will be broad and therefore slowly varying compared to the step size. That justifies Taylor expanding $\rho(x - z, t)$, separating the integral for x and z as well as into its first moment components:

$$\rho(x, t + \Delta t) \approx \rho(x, t)\langle z^0 \rangle - \frac{\partial \rho}{\partial x}\langle z^1 \rangle + \frac{1}{2}\frac{\partial^2 \rho}{\partial x^2}\langle z^2 \rangle$$

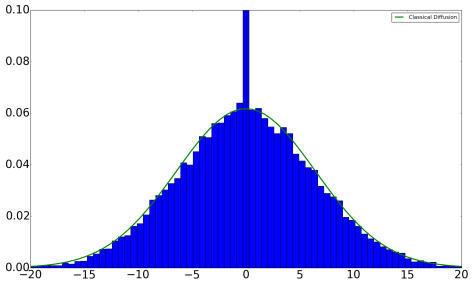
$$\rho(x, t + \Delta t) \approx \rho(x, t) + \frac{l_0^2}{2}\frac{\partial^2 \rho}{\partial x^2}.$$

$$\frac{\partial \rho}{\partial t} \approx \frac{l_0^2}{2\Delta t}\frac{\partial^2 \rho}{\partial x^2}.$$

The classical diffusion equation is found by collecting the constant terms, defining the diffusion coefficient $D = \frac{l_0^2}{2\Delta t}$. Assuming that the concentration c reflects



(a) Binned Monte Carlo simulation using 20000 walkers, 400 positions, and 499 iterations.



(b) Binned Monte Carlo simulation using 20000 walkers, 400 positions, and 500 iterations.

Figure 4.1: Comparison between simulated random walk and analytically found Gaussian solution. The green envelope is the analytical solution. The vertical slash corresponds to x=0.

the underlying probability distribution, we set $c = \rho$, and

$$\frac{\partial c}{\partial t} = D\frac{\partial^2 c}{\partial x^2}, \tag{4.4}$$

Different general solutions apply for different initial and boundary conditions (BC). For the purposes of a released tracer gas, we define a set of idealized BCs as

$$c(x, t = 0) = c_0\delta(x), \qquad \lim_{x\to\pm\infty} c(x, t) = 0. \tag{4.5}$$

This set of BCs can describe the co-moving frame of the cross section of a released tracer gas, perpendicular to the flow, if certain assumptions are made. The diffusion can in the approximate sense occur in the horizontal direction perpendicular to the flow if we assume a small and geometrically uniform release point with constant release concentration c_0, under stable wind conditions, and under atmospherically stable flow. Constant release concentration and stable wind conditions constrain the diffusion to the cross section perpendicular to the flow. We can operate in the CM frame to further constrain the diffusion to one axis. Atmospheric stability (sec.2.2), constrains the diffusion to a mainly horizontal plane. Eq.(4.5) then applies for the horizontal direction perpendicular to the flow in the co-moving frame. The solution is the system response to an impulse at $x = 0$ as given in eq.(4.5). It is known (e.g. (Pécseli, 2000)) that the diffusion eq.(4.4) solution to this set of BCs is the centered and normalized Gaussian function:

$$c(x, t) \approx \frac{1}{2(\pi t D)^{\frac{1}{2}}} e^{-\frac{x^2}{4Dt}}. \tag{4.6}$$

The Gaussian solution eq.(4.6) can be shown to correspond with unobstructed random walk of an initial concentration at $x = 0$ by a simple Monte Carlo simulation.

The random walk step PDF χ must be broad and slowly varying compared to the step size. The simplest way to achieve this is to use the uniform distribution $U(a, b)$, defined by

$$p(z) = \frac{1}{b-a}, a < z < b \qquad p(z), \text{else.}$$

We use $U(-0.5, 0.5)$ and decide that $z \leq 0$ corresponds to a leftward "walk", and $z > 0$ is a rightward "walk". This leads to a small artificial asymmetry which is trivial because the distribution is continuous. The distribution $U(-0.5, 0.5)$ must satisfy eqs.(4.1), (4.2), (4.3). This can be shown by

$$\langle z^0 \rangle = \int_a^b \frac{1}{b-a} dx = 1,$$

$$\langle z^1 \rangle = \int_a^b \frac{x}{b-a} dx = \frac{1}{2}(a+b) = 0,$$

$$l_0^2 = \langle z^2 \rangle = \frac{1}{3}(a^2 + ab + b^2) = \frac{1}{12}.$$

Note that the time increment Δt is associated with the stepwise standard deviation l_0. This means that l_0 is the step size of the random walk. Using the distribution $U(-0.5, 0.5)$ to generate "walks" and grid cutoffs suitably far away from 0, we plot the results of the analytical Gaussian solution from eq.(4.6) random walks. These can seen in figs.4.1a and 4.1b. The large spike at $x = 0$ in fig.4.1b is due to the choice of a constant step distance. This means that for even numbers of iterations, about half the walkers to the immediate left and right of the center go to the center, making it about twice as big as it "should" be. The problem does not occur when using odd numbers of steps, as seen in fig.4.1a.

The variance of the diffusion equation (4.4) is of special interest. Defining $\alpha = \frac{1}{4Dt}$:

$$\langle x^2 \rangle = \sqrt{\frac{\alpha}{\pi}} \int_{-\infty}^{\infty} x^2 e^{-\alpha x^2} dx = -\sqrt{\frac{\alpha}{\pi}} \frac{d}{d\alpha} \int_{-\infty}^{\infty} e^{-\alpha x^2} dx = -\sqrt{\frac{\alpha}{\pi}} \frac{d}{d\alpha} (\sqrt{\frac{\pi}{\alpha}})$$

$$\langle x^2 \rangle = 2Dt.$$

The variance describes the spread of a classical diffusive cloud. In particular, we notice that the variance is proportional to the time in a classical diffusive scale. Note that for a homogeneous and isotropic system of several dimensions, D will remain the same, and so each dimension becomes separable in the solution to the diffusion equation. Then the variance will become additive for each dimension, i.e. for n dimensions,

$$\langle r^2 \rangle = 2nDt. \tag{4.7}$$

It was pointed out that the classical diffusion equation (4.4) is only valid for relatively flat distributions and where the chosen time scale has independent steps. Evidently, Δt can take on the smallest possible size of this time scale. Unless the velocity field is infinitely smooth, there will always be scales below this where subsequent steps exhibit memory properties, and correlation becomes important. These are scales where classical diffusion becomes increasingly inaccurate. For systems with correlations, $\rho(x-z,t)$ is not necessarily separable into $\rho(x,t)\rho(z,t)$, and we can no longer assume a broad and slowly varying probability distribution $\rho(x,t+\Delta t)$. Higher order moments $\langle z^3\rangle$, $\langle z^4\rangle$, etc. are no longer negligible. This means that eq.(4.4) no longer applies for turbulent diffusion, which *is* characterized by correlations.

The velocity field is generally not infinitely smooth, but neither are memory properties normally detected under laminar flow, where diffusion is typically measured. In the case of fluid particles, the amount of molecular particles within each 'packet' fluctuates, in a basic approximation, according to the $\sqrt{n}$ rule. For laminar gases, the flow scale does not go below $\mathcal{O}\,(1000)$ of the molecular scale (Pope, 2000), and this amounts to a fluctuation below 3% of the molecular density of fluid particles. Thermodynamic fluctuations may disrupt the particles from following the large scale mean force field, leading to an increasingly random velocity field for smaller scales. In practice, however, this is not encountered in flow measurements, since the scales considered are much larger. The random velocity field exhibited by turbulence is also for that reason not of thermodynamic origin.

This also suggests that diffusion can be seen to naturally arise in time scales where correlation between successive steps can be thought to have been decoupled. The precise size of the scale where this occurs is dependent on the actual field smoothness of both long-range and short-range forces in the system. Below, we describe turbulent diffusion by its variance, and compare it with classical diffusion.

4.2 Single-Particle Diffusion in a Turbulent Velocity Field

Because the inertial range does not explicitly depend on boundary conditions like the energy range, or local inhomogeneity like the dissipation range, the turbulent velocity field may be considered isotropic and homogeneous as long as the turbulent diffusion in discussion is to be understood as within the inertial range of a fully developed turbulent field. In the Lagrangian frame of the "test particle", the turbulent velocity field can be defined as $\mathbf{u}(\mathbf{r}(t),t)$. The position of this particle is in this reference frame specified by $r(t,r_0)$, where r_0 is the position at a reference frame (Pope, 2000). This is important because at this level of description, the

position will evolve according to nonlinear dynamics as it moves away from the sample reference, from a linear, deterministic path, to an increasingly random path. Once a collection of these random paths become fully realized, it is expected that a statistical treatment will be able to describe their spread. It is therefore to be expected that short time scales yield deterministic paths according to the local field, whereas very long time scales should yield a distribution of random paths that behave in a diffusive manner, corresponding to the independent steps of a free random walk.

We are interested in the centered variance of the position, $\langle r^2(t)\rangle$. It can be differentiated:

$$\frac{dr^2(t)}{dt} = 2\mathbf{r}(t)\cdot\frac{d\mathbf{r}(t)}{dt} = 2\mathbf{u}(\mathbf{r}(t),t)\cdot\int_0^t \mathbf{u}(\mathbf{r}(t'),t')dt' = 2\int_0^t \mathbf{u}(\mathbf{r}(t),t)\cdot\mathbf{u}(\mathbf{r}(t'),t')dt'.$$

The ensemble average becomes:

$$\frac{d}{dt}(\langle r^2(t)\rangle) = 2\int_0^t \langle \mathbf{u}(\mathbf{r}(t),t)\cdot\mathbf{u}(\mathbf{r}(t'),t')\rangle dt',$$

where the mean within the integral is the un-normalized Lagrangian velocity correlation function. For a short time scale, the history of the turbulent velocity field over times [0,t] may be approximated at the sample time t, effectively taking the trace over the correlation tensor. The correlation tensor becomes variance $\langle u^2\rangle$. We assume a stationary process, where the variance is constant, so that:

$$\frac{d}{dt}(\langle r^2(t)\rangle) = 2\langle u^2\rangle t$$

$$\langle r^2(t)\rangle = \langle u^2\rangle t^2. \tag{4.8}$$

This is called the *ballistic limit*, the result we get by assuming the particle to follow straight lines: $\mathbf{r}(t) = \mathbf{u}t$. If the ballistic limit is not found in a set of experiments, it suggests that either the field is very rapidly changing, or the measurements are at a much larger scale and don't have the resolution to identify the short time scales, as suggested in the above section. In the context of non-linear or anomalous diffusion, this result corresponds to superdiffusive power scaling

$$\langle u^2\rangle \propto t^\alpha, \alpha > 1 \tag{4.9}$$

where in this case, $\alpha = 2$.

If we let several paths in a turbulent velocity field continue for an indefinite amount of time, a discrepancy in any of their earlier states quickly leads to the paths being radically different. However, the fact that we can predict that all

paths above *some* time scale will be de-correlated, means that for a very long time scale

$$\int_0^\infty \langle u(t') \cdot u(t) \rangle dt' = \langle u^2 \rangle \tau,$$

where τ can be taken as a constant, called the *Lagrangian integral time scale* of the correlation function. Solving for $\langle r^2(t) \rangle$,

$$\frac{d}{dt}(\langle r^2(t) \rangle) = 2 \int_0^t \langle \mathbf{u}(\mathbf{r}(t), t) \cdot \mathbf{u}(\mathbf{r}(t'), t') \rangle dt' = 2 \langle u^2 \rangle \tau$$

$$\langle r^2(t) \rangle = 2 \langle u^2 \rangle t \tau,$$

we find a diffusive scaling that is linear with time. Suppose we are in a 2 dimensional turbulent field. Defining $D = \frac{1}{2} \langle u^2 \rangle \tau$,

$$\langle r^2(t) \rangle = 4Dt,$$

which agrees with the previously found classically diffusive variance in eq.(4.7). In the context of the diffusive scaling in eq.(4.9), long time scale diffusion corresponds to $\alpha = 1$, so that the the exponent throughout the range of time scales seemingly takes values in the range $1 \leq \alpha \leq 2$.

We can compare this macroscopic view of diffusion with the previous microscopic derivation of the diffusion eq.(4.4). Note that the dimensionality is outside D in eq.(4.7) due to the separable solution, while it is contained inside $\langle \mathbf{u} \cdot \mathbf{u} \rangle$ here.

$$D = \frac{l_0^2}{\Delta t} = \frac{1}{2} \langle u^2 \rangle \tau,$$

we find a description of the long time scale turbulent diffusion through microscopic properties and the correlation length:

$$\frac{\tau}{\Delta t} = \frac{1}{\langle u^2 \rangle} \frac{a^2}{(\Delta t)^2}.$$

The limit of times much larger than the Lagrangian correlation time is called the diffusion limit. The condition that

$$t >> \Delta t$$

for classical diffusion makes sense. Then, in 2 dimensions, for long time ranges and stationary process in homogeneous and isotropic fields,

$$\langle r^2(t) \rangle = 2 \langle u^2 \rangle t \Delta t = 4Dt,$$

$$D = \frac{1}{2} \langle u^2 \rangle \Delta t.$$

A full description of turbulence encompasses all scales between the very largest and very smallest, because important quantities like energy are not generally scale independent in a turbulent field. To describe the spread of the full range of time-scales, modeling from experimentally found pdfs is necessary. However, the mathematical expression may be simplified somewhat (Brenna, 2013), for in a stationary process, all moments are constants when averaged over position, and so:

$$\langle r^2(t)\rangle = 2\langle u^2\rangle \int_0^t \int_0^{t''} R_L(t''-t')dt'dt'',$$

where R_L is the normalized Lagrangian velocity correlation function. This can be simplified by introducing the variables $\tau = t'' - t'$ and $s = t$, and performing a change of variables with Jacobian

$$J = \begin{vmatrix} \frac{\partial \tau}{\partial t''} & \frac{\partial \tau}{\partial t'} \\ \frac{\partial s}{\partial t''} & \frac{\partial s}{\partial t'} \end{vmatrix} = 1.$$

Then,

$$\int_0^t \int_0^{t''} R_L(t''-t')dt'dt'' = \int_0^t \int_0^s R_L(\tau)d\tau ds = \int_0^t \int_\tau^t R_L(\tau)dsd\tau.$$

Since there is no explicit dependence on the variable s in the correlation function, the positional variance is connected to the velocity variance by:

$$\langle r^2(t)\rangle = 2t\langle u^2\rangle \int_0^t (1-\frac{\tau}{t})R_L(\tau)d\tau.$$

We can Fourier transform the Lagrangian correlation function with

$$S_L(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} R_L(t)e^{-i\omega t}dt,$$

to get the Lagrangian power spectrum. We can therefore estimate the relation between turbulent diffusion and the time-stationary velocity fluctuations in frequency space as:

$$\langle r^2(t)\rangle = t^2\langle u^2\rangle \int_{-\infty}^{\infty} \mathrm{sinc}^2\left(\frac{\omega t}{2}\right) S_L(\omega)\, d\omega.$$

If we have the Lagrangian power spectrum, we can estimate, at least for homogeneous and isotropic turbulence, the effect of the measured fluctuation of the wind profile on the turbulent diffusion of a released toxic aerosol in the atmosphere. In reality, we would need to first express this in a Eulerian frame, which is in general

not easy, but is after all how we measure the turbulent diffusion experimentally. We also need to take into account the properties of the boundary layer, which is in general neither homogeneous nor isotropic, and varies with season and time of day. Nevertheless, this idealized discussion is important because it clarifies that in the absence of a complete theory which allows us to deduce the precise expression for the correlation function,the minimum knowledge for predicting the spread of a toxic or pollutive gas by turbulent diffusion is an experimentally verified model for the Lagrangian power spectrum, or its equivalent. In principle, the Lagrangian integral time scale τ can be estimated from a characteristic length and velocity. The length step is estimated for $mad15J$ as on the order of 25[m] in section 4.4, but unfortunately the velocity requires an independent measurement which is not available through the LIDAR data alone.

4.3 Two-Particle Diffusion

A more realistic development is to approach the case of a diffusive cloud composed of multiple fluid particles. From the lab frame, we are interested in the relative spread of the particles, rather than the RMS position of a single tracer, and the simplest possible model for this is the relative diffusion of *two* tracer particles.
In the context of a completely random (in the sense of complete lack of correlation) velocity field, every random walker in a classically diffusive gas is statistically independent. Then Bienaymés formula states that

$$Var\left(\sum_{i=1}^{n} x_i\right) = \sum_{i=1}^{n} Var(x_i).$$

Thus for the long time scale solution of two particles in a turbulent velocity field, we expect the relative diffusion of the gas to be twice what was found for a single particle:

$$\langle x^2 \rangle_2 = 2 \cdot \langle x^2 \rangle_1 = 4t\tau \langle u^2 \rangle. \tag{4.10}$$

This can be shown by defining the separation between particles at $\mathbf{r}_1$ and $\mathbf{r}_2$: $\mathbf{y}(t) = \mathbf{r}_1 - \mathbf{r}_2$, and the separation between two local velocity fields: $\Delta\mathbf{u}(t) = \mathbf{u}(\mathbf{r}_1(t), t) - \mathbf{u}(\mathbf{r}_2(t), t)$. Then

$$\mathbf{y}(t) = \mathbf{y}(0) + \int_0^t [\mathbf{u}(\mathbf{r}_1(t'), t') - \mathbf{u}(r_2(t'), t'))]dt' = \mathbf{y}_0 + \int_0^t \Delta\mathbf{u}(t')dt'$$

and

$$\frac{d}{dt}(\mathbf{y}^2(t)) = 2\Delta\mathbf{u}(t)\mathbf{y}(t) = 2\Delta\mathbf{u}(t)\mathbf{y}_0 + 2\int_0^t \Delta\mathbf{u}(t')\Delta\mathbf{u}(t)dt',$$

so the ensemble average becomes

$$\frac{d}{dt}(\langle \mathbf{y}^2(t)\rangle) = 2\int_0^t \langle \Delta\mathbf{u}(t')\Delta\mathbf{u}(t)\rangle dt' = 2\int_0^t \langle (\mathbf{u}_1(t') - \mathbf{u}_2(t'))\cdot(\mathbf{u}_1(t) - \mathbf{u}_2(t))\rangle dt', \tag{4.11}$$

which for the non-correlated time scales means that

$$\langle y^2\rangle = 4\tau t\langle u^2\rangle$$

as predicted qualitatively in equation (4.10).

Shorter time scales exhibit gradually stronger correlations due to the turbulence, and the relative diffusion is therefore in general not twice the single-particle solution for the equivalent time scale. The isotropic condition that was rather artificially imposed on the discussion simplifies the problem considerably through symmetry. An isotropic tensor of the second order is of the form:

$$Q_i(\mathbf{r}) = Ar_ir_j + B\delta_{ij}$$

for even, **scalar** functions A and B. We can write the velocity correlation tensor in the form:

$$R_{ij}(\mathbf{r}) = F(r)r_ir_j + G(r)\delta_{ij}. \tag{4.12}$$

To identify a means to measure the identity of F and G, we can assume incompressibility. This is not far-fetched considering the nature of the atmosphere, and that we already assumed a homogeneous and isotropic turbulence. Following the incompressibility condition:

$$\nabla\cdot\mathbf{u} = 0, \tag{4.13}$$

we want to relate this to velocity correlations. Noting that the normalized velocity correlation tensor for two points separated by space vector $\mathbf{r}$ is

$$R_{ij}(\mathbf{r}) = \frac{\langle u_i(\mathbf{x})u_j(\mathbf{x}+\mathbf{r})\rangle}{\langle u^2\rangle},$$

the incompressibility condition (4.13) can be related to the velocity correlation tensor, since for incompressible flow:

$$\left\langle u_i(\mathbf{x})\frac{\partial u_j(\mathbf{x}+\mathbf{r})}{\partial \mathbf{r}_j}\right\rangle = 0$$

$$\left\langle \frac{\partial u_i(\mathbf{x})}{\partial \mathbf{r}_i}u_j(\mathbf{x}+\mathbf{r})\right\rangle = 0,$$

and so explicitly

$$\frac{\partial R_{ij}}{\partial \mathbf{r}_i} = \frac{\partial R_{ij}}{\partial \mathbf{r}_j} = 0.$$

The velocity tensor formulated in equation (4.12) can now be worked out:

$$\frac{\partial R_{ij}(\mathbf{r})}{\partial \mathbf{r}_i} = r_j(4F + r\frac{\partial F}{\partial r} + \frac{1}{r}\frac{\partial G}{\partial r}) = 0,$$

where the 4 occurs instead of 3 since the trace portion yields 2 instead of 1. Since this relation must be satisfied for all indices of r, we have a relation between F and G

$$4F + r\frac{\partial F}{\partial r} + \frac{1}{r}\frac{\partial G}{\partial r} = 0,$$

showing that a single scalar function can specify isotropic turbulence (Batchelor, 1993).

The ballistic limit of relative diffusion in eq. (4.11) is formally (I have used Tr for trace to distinguish from the variable t.)

$$\mathrm{Tr}\langle y_i y_j\rangle = \mathrm{Tr}\{y_{0i}y_{0j} + t^2\langle u^2\rangle[\delta_{ij} - 2R^E_{ij}(y_0, 0)]\}$$

noting that the relative diffusion is in an Eulerian frame. In this notation, $r_i r_j = y_i y_j$ and $r^2 = y_0^2$. Explicitly,

$$\mathrm{Tr}\langle y_i y_j\rangle = \mathrm{Tr}\{y_{0i}y_{0j} + t^2\langle u^2\rangle[\delta_{ij} - 2(F y_i y_j + G\delta_{ij})]\},$$

which for a 2-dimensional case, with $\mathrm{Tr}\{\delta_{ij} = 2\}$ and $\mathrm{Tr} y_i y_j = y_0^2$, becomes

$$\langle y^2(t)\rangle = y_0^2 + 2t^2\langle u^2\rangle(1 - (F y_0^2 + 2G)).$$

The representation might not be physically intuitive, which is why one often uses, especially for experimental work, the alternative set of scalar functions defined by:

$$f(r) = \frac{\langle u_p(\mathbf{x})u_p(\mathbf{x}+\mathbf{r})\rangle}{\langle u_p^2\rangle}$$

and

$$g(r) = \frac{\langle u_n(\mathbf{x})u_n(\mathbf{x}+\mathbf{r})\rangle}{\langle u_n^2\rangle},$$

where u_p and u_n are the parallel and normal velocity components with regard to the vector separation $\mathbf{r}$, and $f(r)$ and $g(r)$ are called the longitudinal and lateral velocity correlation coefficients. They are related to $F(r)$ and $G(r)$ by (Batchelor, 1993):

$$\frac{\langle u_p(\mathbf{x})u_p(\mathbf{x}+\mathbf{r})\rangle}{\langle u_p^2\rangle} = f(r^2)$$

$$\frac{\langle u_n(\mathbf{x})u_n(\mathbf{x}+\mathbf{r})\rangle}{\langle u_n^2\rangle} = g(r^2).$$

Then it may be shown that for two dimensions in the ballistic limit (Pécseli and Trulsen, 1997):

$$\langle y^2(t)\rangle = y_0^2 + 2t^2\langle u^2\rangle(1 - [f(y_0^2) + g(y_0^2]).$$

In this case, for short time scales, the longitudinal contribution is the same as the lateral contribution. This result serves as the basis for illustrating the difference between correlated and uncorrelated diffusion processes.

Strictly speaking, the two particle separation refers to an initial particle release, i.e. a problem that differs from that of a continuous release. The physical mechanism for the particle separations and thereby the plume expansion, are however the same for both bases. We therefore use results from two particle separations in order to understand the plume expansion. As a first approximation, we can interpret the time after release as the travel time from the continuous source to the detector.

4.4 Statistical Analysis for the Plume Mean Square Width

On comparing analytical results with experimental estimates we assume that the mean square width of the plume can be adequately represented by the mean-square two-particle separation. As a summary of the foregoing analysis we can write

$$\sigma_{FF}^2 \equiv \langle (y_1 - y_2)^2\rangle = 2\langle y^2\rangle - 2\langle y_1 y_2\rangle,$$

where the last term represents a correlation between the two particles. When the separation is very large, we assume $\langle y_1 y_2\rangle \approx 0$, and the two particles diffuse independently. This will correspond to the diffusion limit discussed before. For short and intermediate times, the correlation $\langle y_1 y_2\rangle$ was discussed previously.

Just as well for mean square separation, the average position $\frac{1}{2}(y_1 + y_2)$ is also statistically varying, and also this variation has a standard deviation. We can write this as

$$\sigma_{PDF}^2 \equiv \frac{1}{4}\langle (y_1 + y_2)^2\rangle = \frac{1}{2}\langle y^2\rangle + \frac{1}{2}\langle y_1 y_2\rangle.$$

σ_{FF}^2 can be estimated by the widths of the plumes in the fixed frame of reference. At each time we can determine a center of mass of the plume, and use this as a representation for $\frac{1}{2}(y_1 + y_2)$ in that realization. Given these data at each time-sample, we can construct a PDF of the center of mass, and give an experimental estimate of σ_{PDF}^2. The first term, $\langle y^2\rangle$, represents the result we would find by following the trajectory of a single particle. We do not have this information, but

can use the meandering of the center of mass as an equivalent, by the foregoing arguments.

By taking $4\sigma_{PDF}^2$, we can add or subtract σ_{FF}^2 to obtain

$$4\sigma_{PDF}^2 + \sigma_{FF}^2 = 4\langle y^2 \rangle \quad \text{and} \quad 4\sigma_{PDF}^2 - \sigma_{FF}^2 = 4\langle y_1 y_2 \rangle.$$

In table 4.1, we show these quantities for 5 good data sets. These numbers are not dimensionless, having dimension *length*^{2}, and are therefore difficult to compare. In the last column we have used $\langle y^2 \rangle$ to normalize the correlation $\langle y_1 y_2 \rangle$, in order to make it dimensionless. If this normalized quantity is very small, we argue that the plume has reached the (classical) diffusion limit. It is readily seen that we in all cases have $\frac{\langle y_i y_j \rangle}{\langle y^2 \rangle} \gg 1$, but note that *mad15J* is distinguished by having a noticeably smaller numerical value than the other entries. For this particular experiment we will have that the width of the plume σ_{FF} is close to the horizontal "outer scale" of the turbulence, about 25[m], i.e. the separation that makes y_1 and y_2 uncorrelated. This length depends critically on turbulence conditions, and since *mad15* and *mad21* experiments were performed on different days, this may explain why the correlation value of *mad21G* is larger than *mad15J* even though their σ_{FF} are close. For the particular case *mad15J*, we can conclude that the typical step length in the diffusion limit is of the order of 25[m]. For the other cases, the data does not allow similar estimates, but we can conclude that e.g. for *mad21K*, the similar step length should be larger than 100[m].

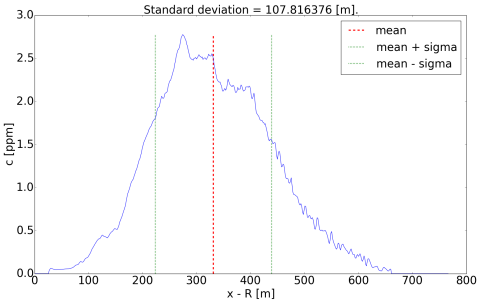
4.5 Results Applied to Experimental Data

We can in principle use the previous results to distinguish the diffusion limit from the earlier ballistic limit. Thus, in case the plume width has become large compared to the horizontal turbulent length scale, we expect two opposite positions measured from the center of mass to be statistically independent.

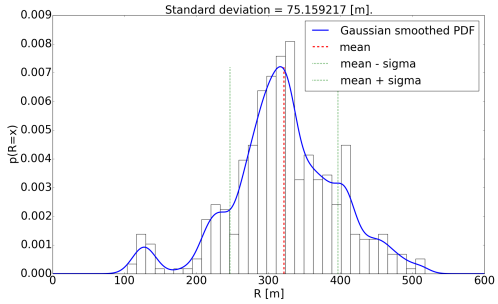
The CM-PDF is shown in fig.4.2b. We compare this with the corresponding root-mean-square (RMS) average plume widths (fig.4.2a).

Table 4.1: Correlation Estimates

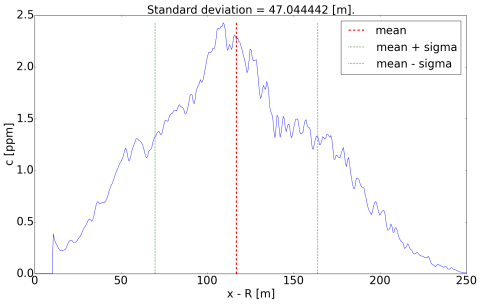
Data Set	σ_{FF}	σ_{CM}	σ_{PDF}	$\frac{\sigma_{CM}}{\sigma_{PDF}}$	$\frac{\sigma_{CM}}{\sigma_{FF}}$	$\langle y_1 y_1 \rangle$	$\frac{\langle y_1 y_2 \rangle}{\langle y^2 \rangle}$
mad21K	107.82	83.07	75.16	1.11	0.771	2740	26.8
mad21G	31.67	22.16	21.63	1.03	0.700	868	29.4
mad21F	91.20	66.33	74.23	0.894	0.727	3430	35.3
mad21H	47.04	30.69	38.76	0.792	0.652	949	18.78
mad15J	24.11	17.31	17.78	0.974	0.718	171	7.18



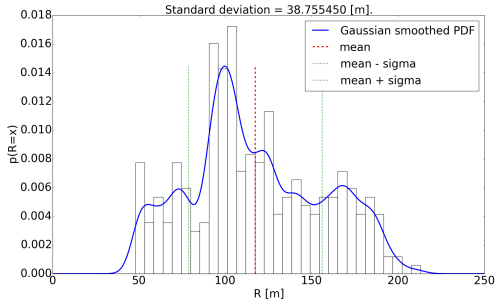
(a) *mad21K*: FF plume width.



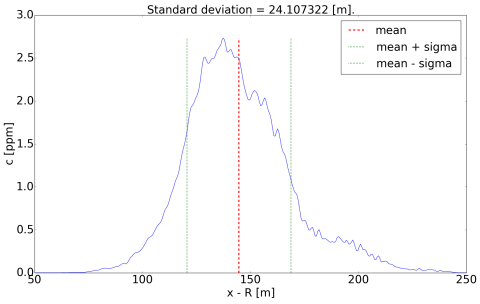
(b) *mad21K*: Meandering of the CM.



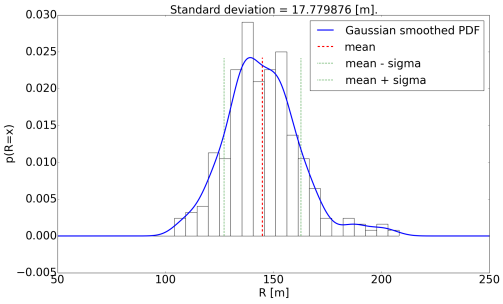
(c) *mad21H*: FF plume width.



(d) *mad21H*: Meandering of the CM.



(e) *mad15J*: FF plume width.



(f) *mad15J*: Meandering of the CM.

Figure 4.2: Comparison between meandering of the CM as an equivalent of single-particle motion and plume width in fixed frame.The space between the red and green lines in the figures denote σ_{FF} (left), and σ_{PDF} (right). The Gaussian Smoothing used for the histograms is discussed in section 6.4.

The plume width and corresponding width of meandering of estimated single-particle motions were calculated as standard deviations. For a normalized CM-PDF with variable CM R, the standard deviation is

$$\sigma_{PDF} = \sqrt{\int_{-\infty}^{\infty} R^2 p(R) dR - \langle R \rangle^2}.$$

For a *binned* PDF, this is approximated as a sum, by the size of bins ΔR, from an array of N bins, as

$$\sigma_{PDF} \approx \sqrt{\sum_{i=1}^{N} R_i^2 p(R_i) \Delta R - \langle R \rangle^2},$$

where the expectation value is precalculated as

$$\langle R \rangle \approx \sum_{i=1}^{N} R_i p(R_i) \Delta R.$$

The CM-frame and FF plume width concentrations are not normalized, so the standard procedure of calculating standard deviation from functions of variables applies. Below is the process used for the CM frame plumes. The process is exactly equivalent for the FF.

$$\sigma_{CM} = \sqrt{\frac{\int_{-\infty}^{\infty} x_{CM}^2 c(x_{CM}) dx_{CM}}{\int_{-\infty}^{\infty} c(x_{CM}) dx_{CM}} - \langle x_{CM} \rangle^2},$$

approximated by

$$\sigma_{CM} \approx \sqrt{\frac{\sum_{i=1}^{N} x_{CM}^2 c(x_{CM})}{\sum_{i=1}^{N} c(x_{CM})} - \langle x_{CM} \rangle^2},$$

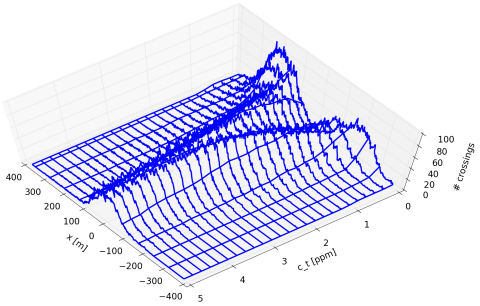
given

$$\langle x_{CM} \rangle \approx \frac{\sum_{i=1}^{N} x_{CM,i} c(x_{CM,i})}{\sum_{i=1}^{N} c(x_{CM,i})}.$$

Table 4.1 shows the equivalent raw and normalized correlations. In addition to the standard deviation in the CM-frame, σ_{CM}, and its ratio with the other standard deviations were added for comparison. We note in fig.4.2d that the distribution might contain two components, one wide and one more narrow. In general, one would interpret this as a change in wind dynamics during the data acquisition, but direct inspection of the data shows nothing conspicuous, so we assume this feature to be a consequence of using a data set of finite duration.

Chapter 5

Data Analysis



5.1 Interpolation for Thresholds

In order to more precisely pinpoint the time at which crossings over concentration thresholds occur, a two-point interpolation is used between neighbors on two sides of the concentration threshold. This is triggered in the same way as in the "brute force" implementation, but now a linear interpolation is computed in order to find a more accurate time of crossing. This affects $\%t$ and $\langle T \rangle$, but not f_{cross}. The continuous threshold used in this section is discussed in sec.5.2.

Given coordinates (t_0, c_0) and (t_f, c_f) for the two neighboring points, a linear interpolation would be computed by

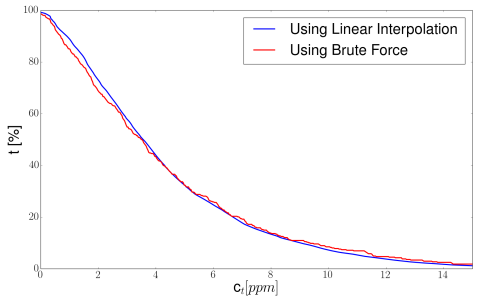
$$c = c_0 + \alpha(t - t_0), \qquad \alpha = \frac{c_f - c_0}{t_f - t_0}.$$

We now wish to have $t(c_t)$, the exact time of crossing according to the linear interpolation, and we find

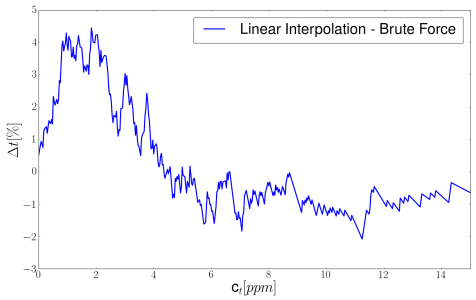
$$t(c_t) = t_0 + (c_t - c_0)\frac{t_f - t_0}{c_f - c_0}.$$

Since the "brute force" script already counts time spent in *integers* according to a counter, we only need to add the fraction of that integer spent above the threshold at the moment of crossing. Thus, $t_f - t_0 = 1$, and what we want to compute is

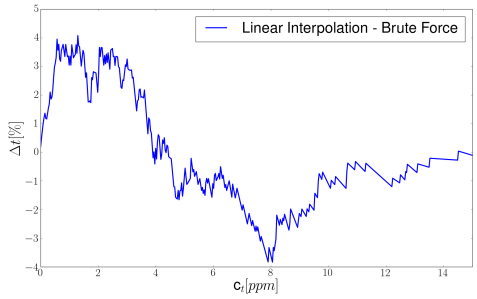
$$t(c_t) - t_0 = \frac{c_t - c_0}{c_f - c_0}.$$



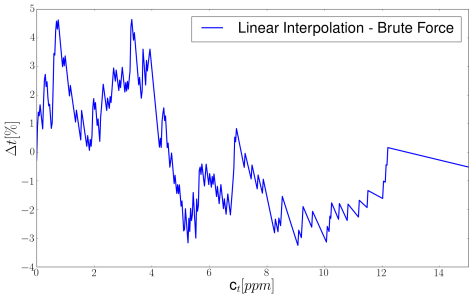
(a) *mad21K*: %t at x = R in CM frame. The "brute force" curve calculates a lower percentage of time above thresholds at low concentrations, and a higher percentage at high concentrations. This can lead to underestimation of time spent above low concentration thresholds, and should be avoided.



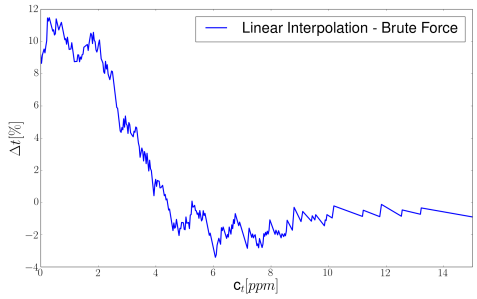
(b) *mad21K*: Percentage time difference between the two curves shown in fig.5.1a, shown as Interpolated calculation minus "brute force" calculation. The difference is never higher than 5% , and peaks at a low concentration. This is partially explained by the fact that the error of a "brute force" method scales with the number of crossings, which according to fig.5.2a peak before 5 ppm.



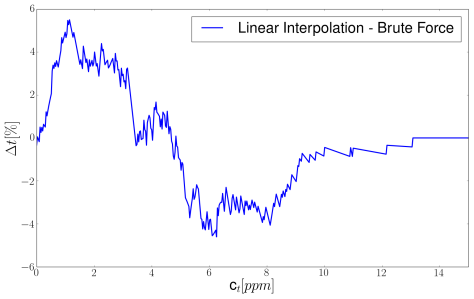
(c) *mad21H*.



(d) *mad21F*.

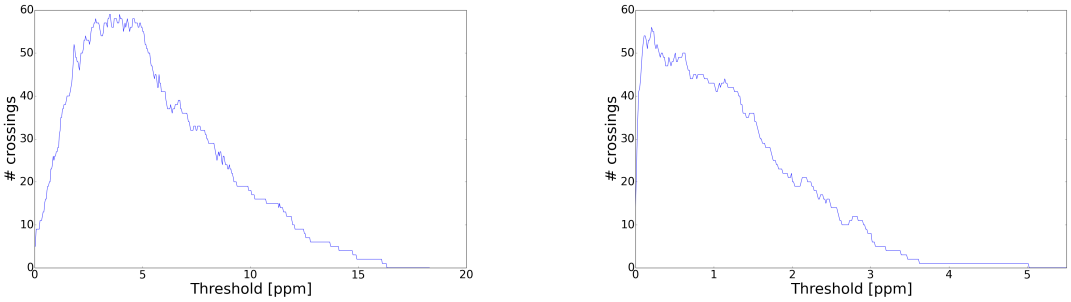


(e) *mad21G*.



(f) *mad15J*.

Figure 5.1: Comparison of the "brute force" method with Linear Interpolation at x = R in CM frame.



(a) Crossings at x = R (b) Crossings at x = R + σ

Figure 5.2: *mad21K* CM frame.

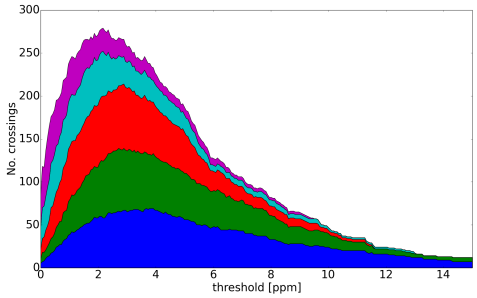
Note that on the "upwards" crossing, the fraction of unit time spent above the threshold is $t_f - t(c_t) = 1 - (t(c_t) - t_0)$, while on the "downwards" crossing it is $t(c_t) - t_0$. This is a result of the direction of the interpolation.

A python function that computes this time, triggered by crossing the given threshold, is sufficient for our purposes. The difference from the "brute force" method is shown in fig.5.1a. The resulting plot is smoother because it is able to accommodate the higher resolution of the "continuous" threshold. This smoothness is to a certain degree manufactured, and is only good to the extent that linear interpolation at this resolution is feasible. The largest differences between the two plots occurs near 2 [ppm], as can be seen in fig.5.1b. This is partially explained by the higher number of crossings here. For *mad21K*, the highest difference does not exceed 5% , although *mad21G* shows an 11% maximum difference between "brute force" and linear interpolation at low concentrations.

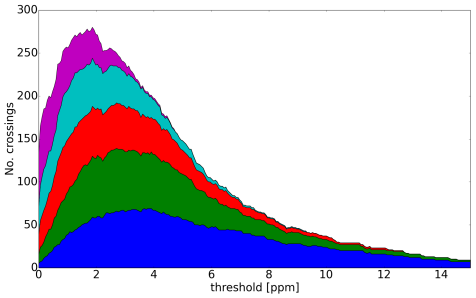
There is a sign change in the error between high and low concentrations. From figs.5.1b-5.1f, it seems that the "brute force" chronically underestimates time at low concentrations, and overestimates at high concentrations. Since low concentration fluctuations are important in many cases of toxic contamination, the underestimation in particular is grounds for using interpolation rather than simply counting.

5.2 Continuous Threshold Sampling

During the "brute force" calculations, two thresholds were used. Now we introduce a continuous range of thresholds. We do this by selecting a number, typically around 500, of c_t along the range 0-20 [ppm], which is near the maximum of occurring concentrations in CM frame. These numbers were found to produce good results by varying the parameters based on the data set in question. Sampling around 500 c_t allowed us to pick up on individual fluctuations at the tail.



(a) Crossings over c_t for positions towards the LIDAR from the CM.



(b) Crossings over c_t for positions away from the LIDAR from the CM.

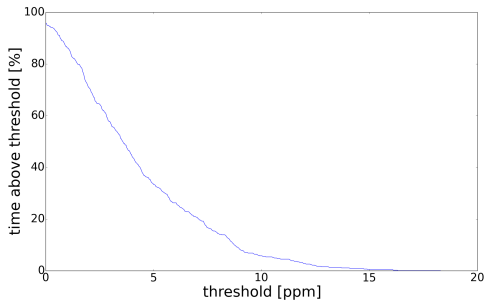
Figure 5.3: Stackplots of *mad21K* in CM frame. Spaced at $\frac{1}{4}\sigma$ from the CM. Blue is at CM in both figures.

Figs.5.2a and 5.2b show the number of crossings in CM frame for *mad21K* at $x = R$ and $x = R + \sigma$ as defined for the "brute force" calculations. We retain σ rather than recalculate an actual standard deviation because we compare with the results of the "brute force" method, where these distances served to illustrate points of interest. For a discussion of how such standard deviations are calculated, see sec.4.5.

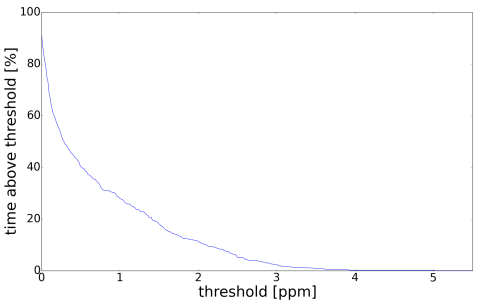
There is a clear shift of maximum number of crossings towards lower concentrations away from the CM. The maximum fluctuation levels are on the same order for the two positions, but fluctuations quickly reduce past 1 [ppm] for $x = R + \sigma$, while for $x = R$ the maximum occurs near 5 [ppm]. There is a line above 0 in fig.5.2b from 4 to 5 [ppm]. This is because of a single 5 [ppm] point at the position, which manifests in the entire range between 4 to 5 [ppm]. It is important to realize that the number of crossings do not necessarily reflect the fluctuations *at that concentration*, but rather fluctuations *over* that threshold.

Figs.5.3a and 5.3b show in total 9 positions spaced in the range $-\sigma < x - R < \sigma$ in CM frame. The distribution of crossings is largely symmetrical on both sides of the CM with small differences, such as the "notch" in fig.5.3b, and the difference in size between the figures of the red and cyan distributions. We know from the "brute force" calculations that a small amount of skewness is to be expected, even in CM frame.

Figs.5.4a and 5.4b show the $\%t$ spent above a threshold. This was found by counting time in units of time samples, as explained in sec.5.1, and normalizing against the number of samples for that position. We see that the shape of the curve is different between the two figures. In particular, fig.5.4a is concave for low concentrations, while fig.5.4b is convex. If the curve becomes increasingly convex as we move away from the CM, it is correspondingly important to know associated exposure times, like those in table 1.1, for low concentrations. Ideally,

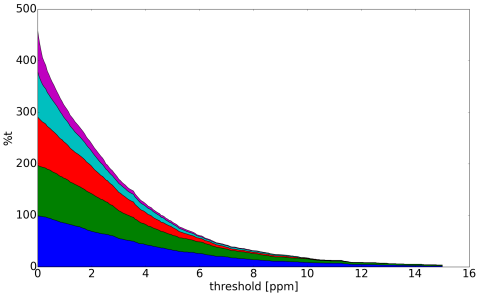


(a) $\%t$ above c_t at x = R.

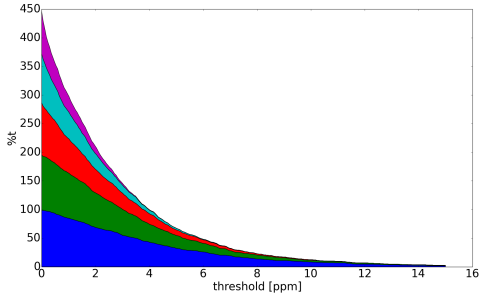


(b) $\%t$ above c_t at x = R + σ.

Figure 5.4: *mad21K*-CM frame: R is the CM. σ is the estimated Gaussian standard deviation we used in sec.1.3, which we keep using to compare with that section.

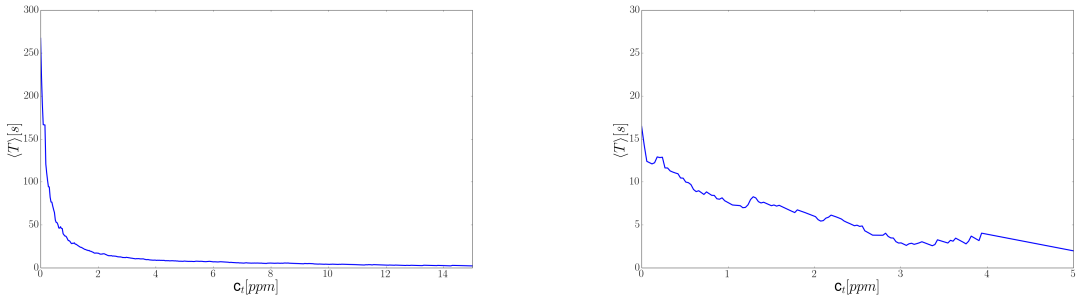


(a) $\%t$ above c_t for positions towards the LIDAR from the CM.



(b) $\%t$ above c_t for positions away from the LIDAR from the CM.

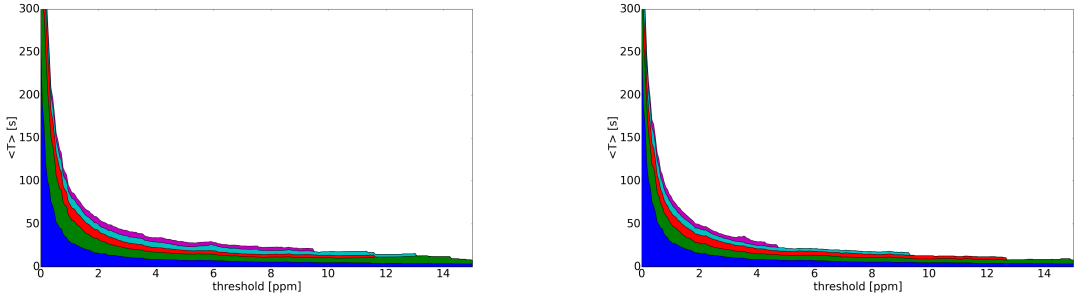
Figure 5.5: Stackplots of *mad21K* in CM frame. Spaced at $\frac{1}{4}\sigma$ from the CM. Blue is CM in both figures.



(a) $\langle T \rangle$ above c_t at x = R.

(b) $\langle T \rangle$ above c_t at x = R + σ.

Figure 5.6: *mad21K*-CM frame: R is the CM. σ is the estimated Gaussian standard deviation we used in sec.1.3, which we keep using to compare with that section.



(a) $\langle T \rangle$ above c_t, for positions towards the LIDAR from the CM.

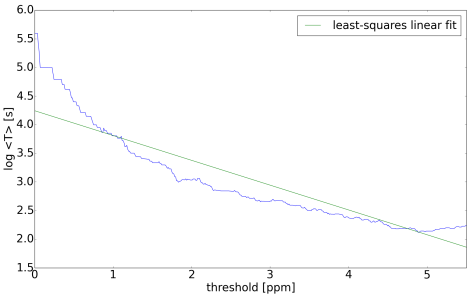
(b) $\langle T \rangle$ above c_t, for positions away from the LIDAR from the CM.

Figure 5.7: Stackplots of *mad21K* in CM frame. Spaced at $\frac{1}{4}\sigma$ from the CM. Blue is CM in both figures.

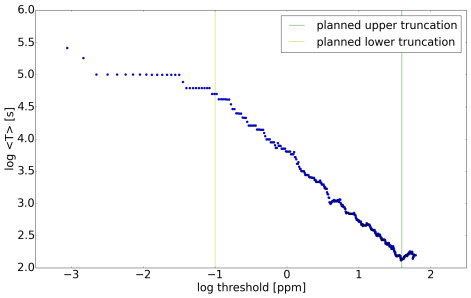
we want estimates of exposure times for continuous concentration thresholds of various pollutants. Such estimates could be compared to the MADONA data by scaling $\%t$ to "simulate" variable time spent in a toxic plume, and could form the grounds for an eventual interdisciplinary project, but are outside the scope of a master's thesis.

Figs.5.5a and 5.5b show stack plots with the same range of positions as the stack plots for crossings. The two plots are symmetrical in form, and display for instance no asymmetrical "notches" like fig.5.3b. Time spent above thresholds depends only on the distribution of concentrations, which we discuss in the form of the PDF $p(c)$ in sec.6.1, while crossings depend on short-term fluctuations. Since $\%t$ is symmetrical, we assume that the underlying turbulent fluctuations can be considered as a stationary process. If true, this suggests that our theoretical discussion from sec.4.2 and later is valid for *mad21K*.

$\langle T \rangle$ was found by dividing the time spent above c_t, in [s], by the number of crossings. As an average number, it is the most generalizable of the three types of



(a) Natural logarithm of vertical axis, fitted by linear regression. Clearly not exponential throughout, although 2-5 [ppm] is a potential fit.



(b) Natural logarithms of vertical and horizontal axes. The section marked by the yellow and green truncation lines is a candidate for linear fit.

Figure 5.8

quantities we consider.

Figs.5.6a and 5.6b show $\langle T \rangle$ at $x = R$ and $x = R + \sigma$ respectively. The high number of crossings from 2 to 5 [ppm] heavily split up the total time in this region, making the $\langle T \rangle$ of the entire range of positions convex, as is suggested by the stack plots in figs.5.7a and 5.7b. This also shortens the relevant c_t range considerably. We have used a range between 0 and 5.5 to accommodate the concentration threshold with known exposure time at 5 [ppm].

The convex form of all the sample positions in the stack plots suggests that we try power law fits. We found the best way to perform both power law and exponential fits to be in log or log-log space. Linear fits in log space correspond to exponential fits, while linear fits in log-log space corresponds to power-law fits. Using log or log-log space, we can easily see for what range a fit is linear, and truncate outside that range. This can be shown for power law fits:

$$\ln y = a \ln x + b,$$

$$y = Cx^a,$$

and for exponential fits:

$$\ln y = ax + b,$$

$$y = Ce^{ax},$$

where $C = e^b$ is a constant coefficient.

Consider fig.5.6a, at the CM. Taking first the natural logarithm along the vertical axis, we see that an exponential fit fails outside 2 to 5 [ppm]. exponential (fig.5.8a). Upon also taking the logarithm of the horizontal axis, we find by inspection that the section between 4.95 ppm and $\frac{1}{e}$ [ppm] in fig.5.8b, between the yellow and green lines, is a candidate for linear fit. This is a better fit than what

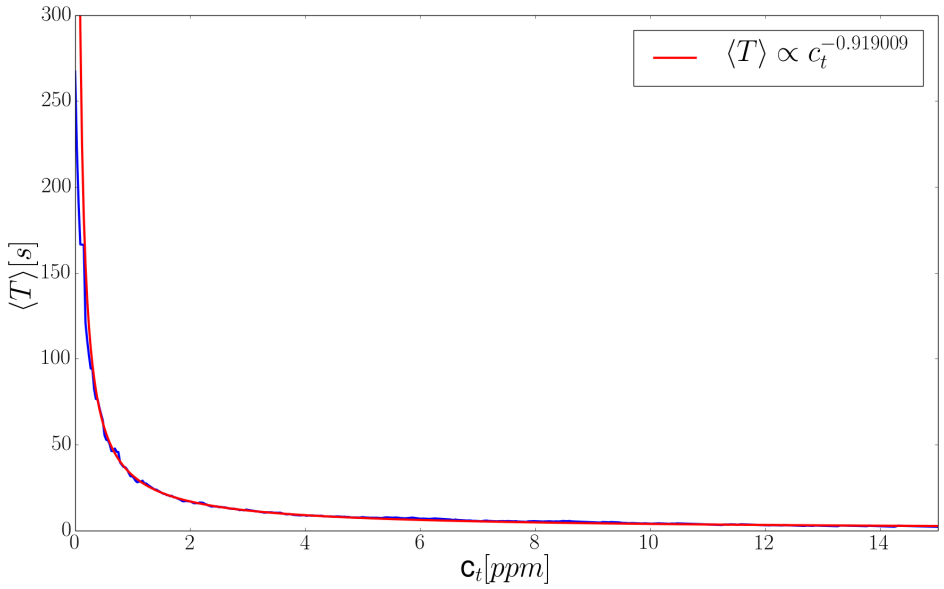


Figure 5.9: Truncated and fitted by linear regression. $\langle T \rangle \propto c_t^{-0.919}$.

we could achieve in fig.5.8a. We performed a truncation for these limits, and fitted by linear regression. The resulting fit can be seen in fig.5.9. For this sample, the linear regression gives

$$\langle T \rangle \approx 32.1 c_t^{-0.919}.$$

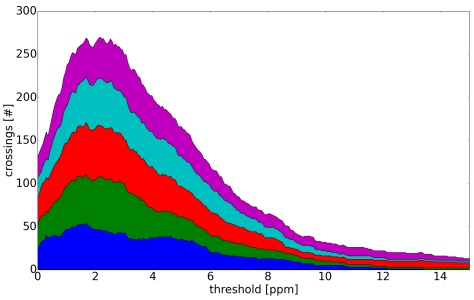
This is close to

$$\langle T \rangle \sim \frac{1}{c_t}. \tag{5.1}$$

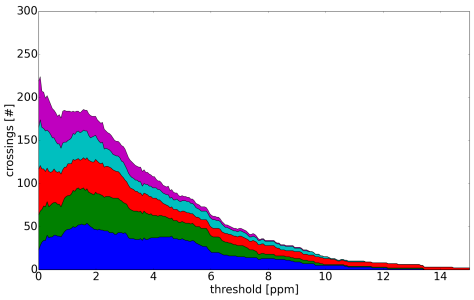
The fit becomes better for large c_t. The consequences of this are discussed in sec.6.4.

$\langle T \rangle$, $\%t$ and number of crossings were also found in FF. Their stack plots are shown in figs.5.10a-5.10f. We do not expect any symmetry in the fixed frame, and especially the number of crossings show that the concentration fluctuations act different on either side of the center of measurement. This results in the small-concentration $\langle T \rangle$ on the side away from the LIDAR being noticeably smaller, since the fluctuations divide the total time above c_t more heavily closer to $c_t = 0$ on this side.

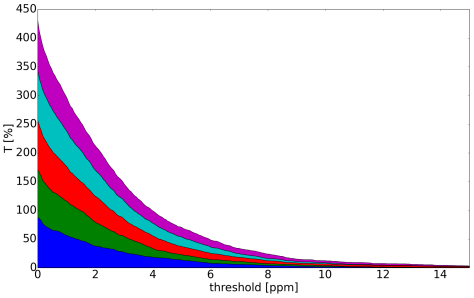
Figs.5.11b, 5.11d and 5.11f show the power-law model generated by a least squares fit in logarithmic space for distances away from average concentration, as argued in sec.6.4, and terminated before reaching the high fluctuation tail of the data. The power-law relation between time expectation and threshold concentration on the LIDAR side, relative to the center of measurement, is closer to



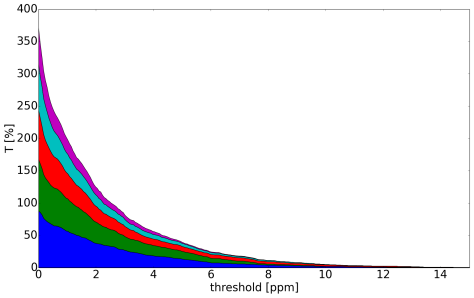
(a) Crossings over c_t on the LIDAR side of center of measurement.



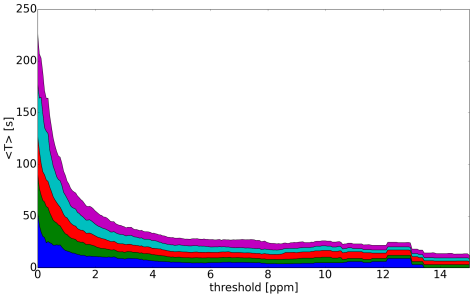
(b) Crossings over c_t on the side opposite to the LIDAR from the center of measurement.



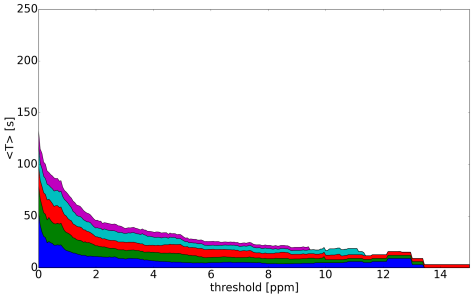
(c) $\%t$ over c_t on the LIDAR side of center of measurement.



(d) $\%t$ over c_t on the side opposite to the LIDAR from the center of measurement.

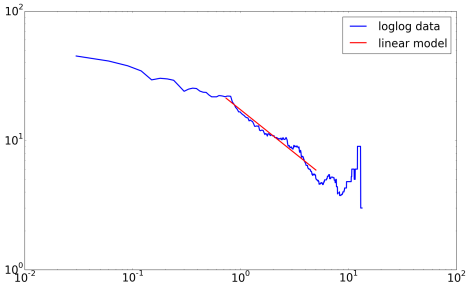


(e) $\langle T \rangle$ on the LIDAR side of center of measurement.

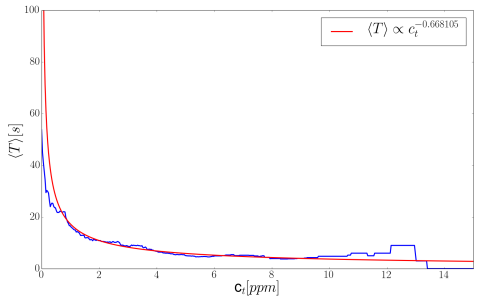


(f) $\langle T \rangle$ over c_t on the side opposite to the LIDAR from the center of measurement.

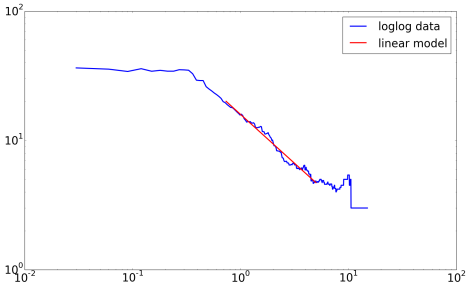
Figure 5.10: Stackplots of *mad21K* in FF. Spaced at $\frac{1}{4}\sigma$ from the CM. Blue is CM in all figures.



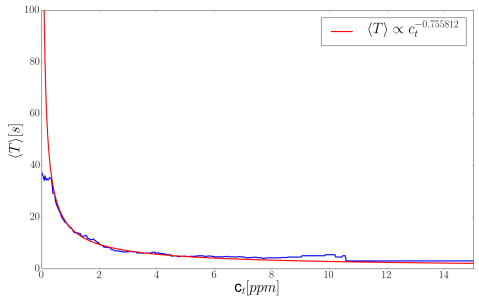
(a) x = 384 [m]. Sample range of linear model in log-log space.



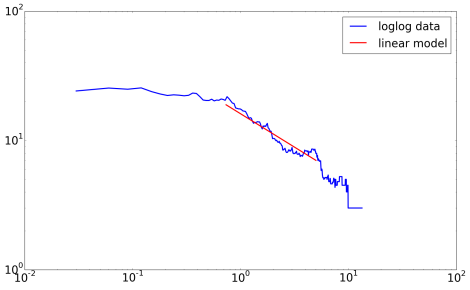
(b) x = 384 [m]. Power-law fit in linear space. The legend reads $\langle T \rangle \propto c_t^{-0.668105}$.



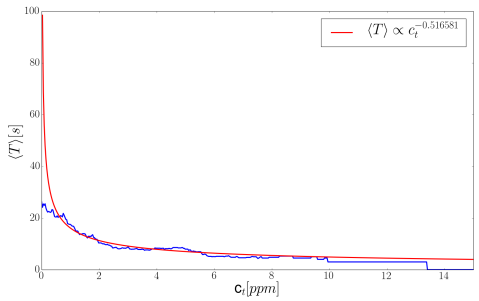
(c) x = 354 [m]. Sample range of linear model in log-log space.



(d) x = 354 [m]. Power-law fit in linear space.The legend reads $\langle T \rangle \propto c_t^{-0.755812}$.

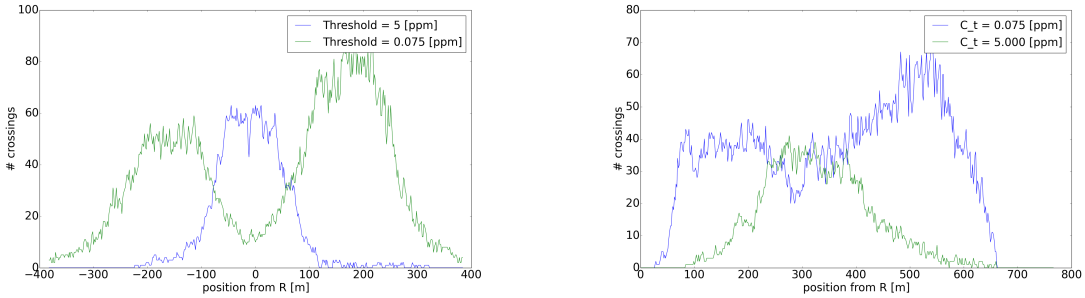


(e) x = 414 [m]. Sample range of linear model in log-log space.



(f) x = 414 [m]. Power-law fit in linear space. The scaling legend $\langle T \rangle \propto c_t^{-0.516581}$.

Figure 5.11: Power law fit of $\langle T \rangle$ against c_t in linear and logarithmic space. Uncertainty in the fit depends on the extension of the linear fit in log space. This depends on what we consider as noise, as well as the fact that logarithms "squeeze" data tighter for high log values. This means that the uncertainty is driven by qualitative estimates. Calculating a fit with the smallest possible standard deviation of data would therefore not be optimal in this case. The error in the exponent was estimated to be within ±10%.



(a) CM frame. Green is $c_t = 0.075$ [ppm], blue is $c_t = 5$ [ppm].

(b) FF. Green is $c_t = 5$ [ppm], blue is $c_t = 0.075$ [ppm].

Figure 5.12: *mad21K*. Crossings for c_t with known exposure time.

inversely proportional than the opposite side. This may be due to the asymmetry observed for a majority of the data sets investigated, suggesting a systematic variation. Since it is unlikely that the plumes are systematically skewed in one direction of the LIDAR beam, a more likely interpretation implies an error in the data noise reduction. We believe that the error lies in the correction formula for the extinction of the laser beam intensity as it propagates through the gas. If this is true, the LIDAR side concentrations are statistically more representative for the plume as a whole than the opposite side. This complicates the evaluation of the skewness of the actual plume.

We note that the uncertainty of the power-law fits in this section is dependent on the truncation range. This depends on factors such as the noise for high concentrations in log space, as well as considerations of statistical independence as explored in sec.6.4. This means that a simple standard deviation analysis of the error in the fit would be superficial and misleading. We have therefore decided to omit that.

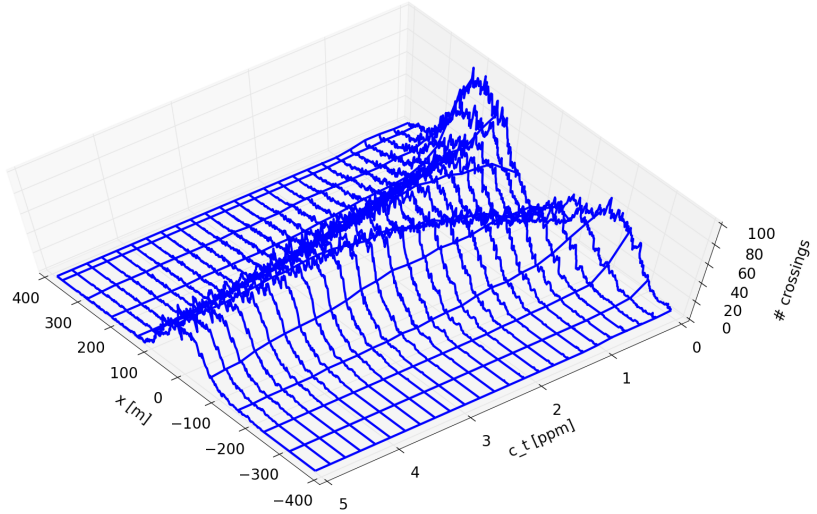
5.3 Continuous Spatial Sampling

In addition to sampling along a continuous concentration threshold at a fixed position, we also sampled the data at positions along the LIDAR line of sight, first in CM frame, then in FF. This sampling was done at all positions defined by the MADONA data arrays. Plotting crossings over positions in CM frame for several c_t, we found a change of form for small concentrations, so that the crossings split into two "bumps". Fig.5.12a shows crossings over positions in CM frame. There is a small "cleft" for $c_t = 5$ [ppm], noticeable in CM frame near $x = R$, which starts dividing as we sample over lower thresholds. Crossings are maximized near 1.5σ for $c_t = 0.075$ in CM frame. In the case of the SO_2 plumes in Miyake, this manifests as a passive pollution for up to 2σ away from the CM, as found

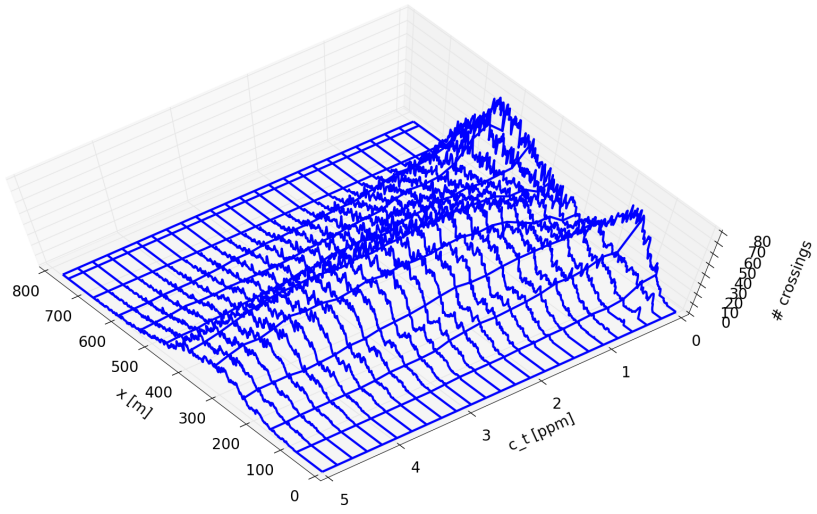
in sec.1.3. This high fluctuation at the wings becomes a large problem when we consider deadlier atmospheric contaminants (than SO_2). Note that the low level of crossings measured for $c_t = 0.075$ [ppm] near the CM is due to most concentration samples being *above* 0.075 [ppm], while the low number of crossings at the edges is due to the lower overall concentration here. This is noticeable when comparing with the FF fig.5.12b, where the "cleft" in the middle for $c_t = 0.075$ [ppm] is less pronounced, since the convecting effect of the wind is included. The "bump" on the side away from the LIDAR is also significantly larger than the "bump" on the opposite side for small c_t.

To investigate this "forking" of crossings for lower concentrations, we plotted the results in 3D wire frame representation. This allowed us to see the beginnings of a bifurcation at $c_t = 3$ [ppm] in CM frame, fig.5.13a. In FF, the results are more spread out, as expected, since in the FF all eddy sizes contribute to the spreading out, while in the CM frame essentially only turbulent eddies smaller than or comparable to the width of the cloud will have an effect. There is no clear bifurcation, but lower c_t are dominated by the "forked" structure. We made the color coded surface plots seen in figs.5.14a and 5.14b in order to show that the beginning of the bifurcation itself does not necessarily represent the start of the interesting structural change of the data. The color plots show that the splitting occurs in FF as well, but is noticeably more asymmetric. The splitting in FF occurs near $c_t = 0.5$ [ppm], while the splitting in CM frame occurs near $c_t = 1.5$ [ppm] in this representation. This representation is therefore better for showing the "forked" regions, while the wire frame plots can show the beginnings of bifurcations. These correspond to different physical interpretations. The small region for low c_t nestled between the two "forked" legs of the structure represents positions where c is almost continuous above c_t. We see that this regions is "skewed" from the center of the structure for FF. This means that if we stand up to 100 [m] away from what we observe as the center of the plume movement as shown in red, we could still be in a region guaranteed to have a *stable* concentration up to $c = 0.5$ [ppm], as shown in the nestled green section in the same plot. On the other hand, the beginning of bifurcation in fig.5.13a is interesting if we know at what concentrations recommended exposure times drastically *increases*. If, for instance, this occurs for concentrations higher than 3 [ppm], we could advise that standing a distance of about 100 [m] is *sufficient* to decrease risk of exposure considerably.

Figs.5.15a and 5.15b show the percentage time spent above a given concentration threshold for the data sample *mad21K* in CM frame and FF, respectively. As expected, the peak is at the center of mass for CM frame, while it is skewed towards the LIDAR side in FF. We also see that the side away from the LIDAR has higher local variation in $\%t$. We also see that the shape is more spread out in FF. As we mentioned earlier, this makes concentration fluctuations less predictable

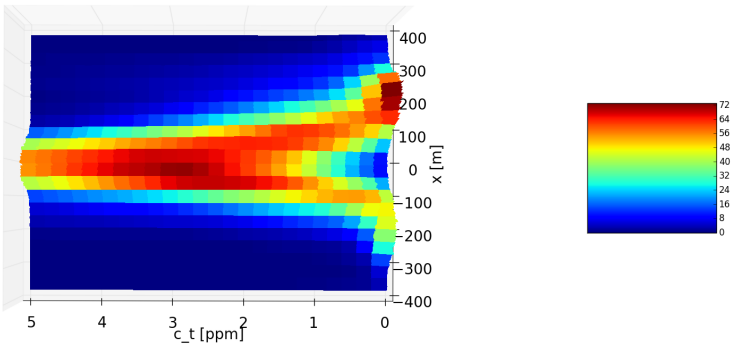


(a) *mad21K*-CM frame. Distance measured from R.

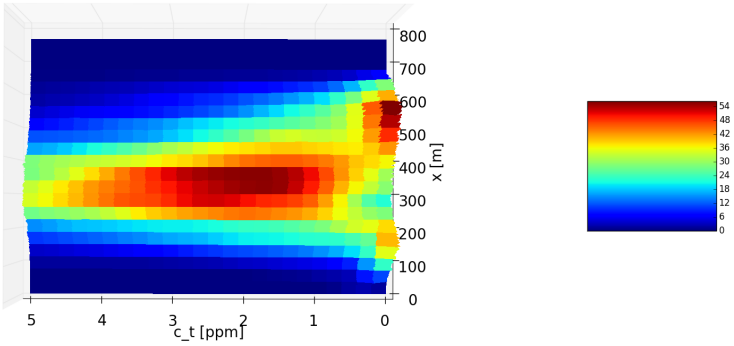


(b) *mad21K*-FF. Distance measured from LIDAR side.

Figure 5.13: Wire frame plot of number of crossings against c_t and distance.

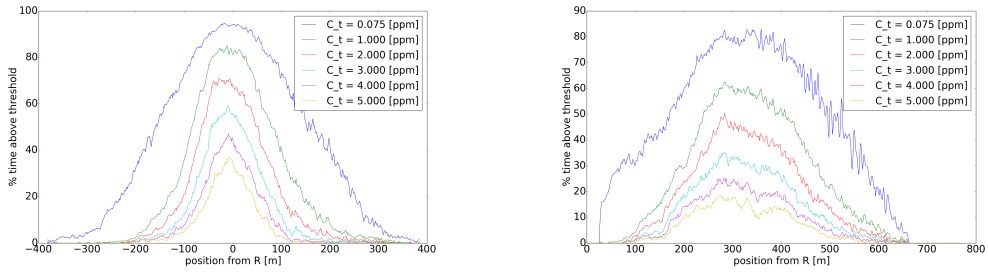


(a) Color coded surface plot of fig.5.13a, seen from above.



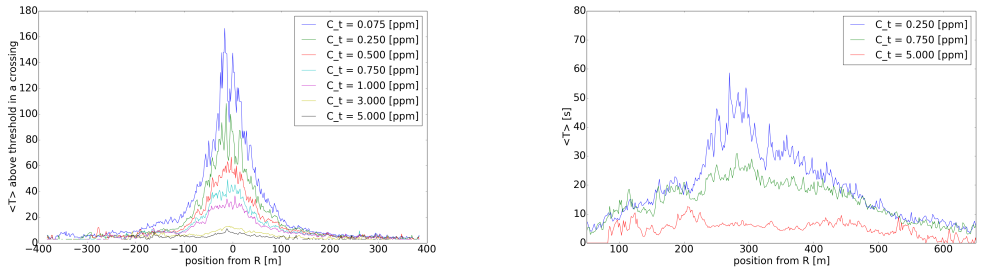
(b) Color coded surface plot of fig.5.13b, seen from above.

Figure 5.14: Color coded representation of crossings against c_t and distance.



(a) CM frame. (b) FF.

Figure 5.15: *mad21K*: Comparison of the percentage time spent above a selection of concentration thresholds.



(a) CM frame. (b) FF.

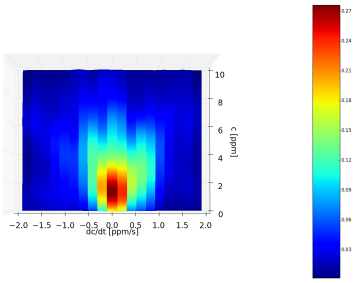
Figure 5.16: $\langle T \rangle$ above c_t.

away from the CM, but we see now that it also results in a smaller percentage of total time being spent above each c_t.

$\langle T \rangle$ for a continuous variation of positions is shown in figs.5.16a and 5.16b for CM frame and FF, respectively. While $\%t$ varied regularly with constant spacings of c_t, it is clear from fig.5.16a that $\langle T \rangle$ quickly falls off from low c_t. Note that $\langle T \rangle$ is significantly spread out in FF. This makes sense because the crossings were seen to be less affected by the change in reference frame than $\%t$.

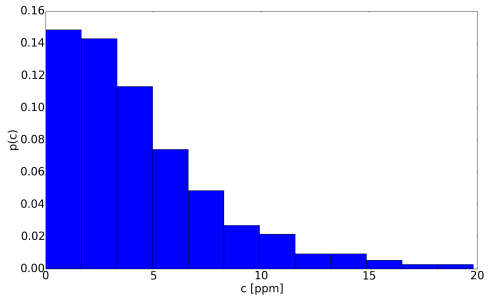
Chapter 6

Probability Distribution Estimates

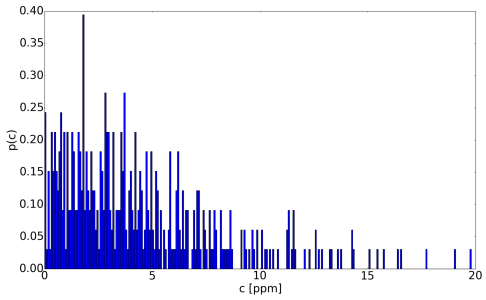


6.1 Histogram PDFs

PDFs of the form $p(c)$ were created for the data set $mad21K$, scaled by the Miyake data, by sampling the concentrations at fixed positions over time into equally sized bins. Fig.6.1a represents a typical low-resolution PDF. It is ordered in the sense that the gradual decrease of occurrences of higher concentrations is well represented, but any local maximum of the PDF not at $c = 0$ is hidden by the low resolution. Fig.6.1b represents on the other hand an excessively high resolution PDF. The local maximum around $c = 0.0005$ is visible, but the resolution is too



(a) PDF at center of mass using 12 bins.



(b) PDF at center of mass using 270 bins.

Figure 6.1

high compared to the sampling, and artificial fluctuations appear. The tail exhibits equally tall fluctuations. Evidently, the height is the lowest unit of counting within the pdf, and a synthetic fit will exhibit larger than necessary errors in the tail. The over-representation of $c = 0$ is also artificially created by the high resolution, as smaller distances between sampled concentration spacings ultimately lead to concentrations with no possible occurrences. 270 bins was chosen as an example where the frequency of sampling is guaranteed to exceed the smallest meaningful concentration distance in the sample.

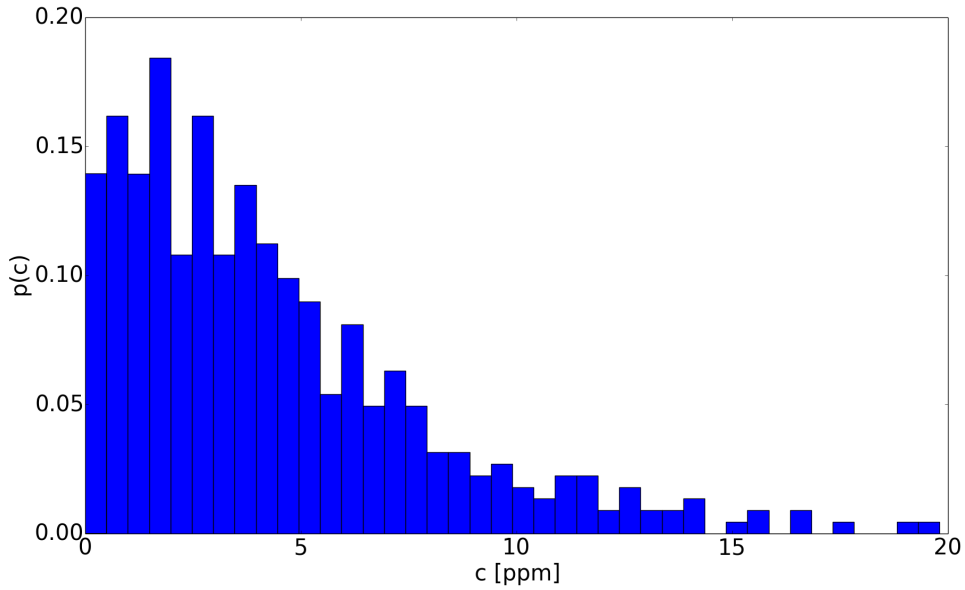


Figure 6.2: PDF at center of mass using 40 bins. This resolution was found to be good for modeling purposes.

For modeling purposes, a high resolution is less important than the potential errors from fitting to synthetic errors. Hence a resolution of 40 bins was found to be good. Fig.6.2 shows $p(c)$ at the resolution which is used for the rest of the section.

Based on the results from sec.6.4, we try a Gaussian PDF with mean and standard deviation calculated from the raw data via

$$\langle c \rangle = \frac{\sum_i^N c_i}{N} \qquad \sigma^2 = \langle c_i^2 \rangle - \langle c \rangle^2 = \frac{\sum_i^N c_i^2}{N} - \langle c \rangle^2,$$

where N is the number of points in temporal space for the given sample. The Gaussian model then becomes

$$p(c) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(c-\langle c \rangle)^2}{2\sigma^2}}.$$

The Gaussian fit is visualized in fig.6.3. In this case, the mean and standard

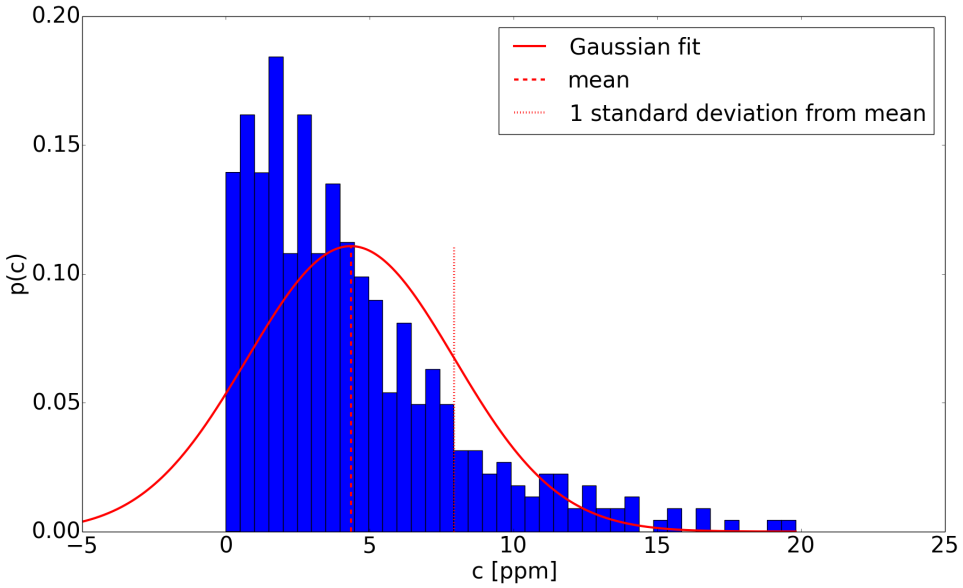


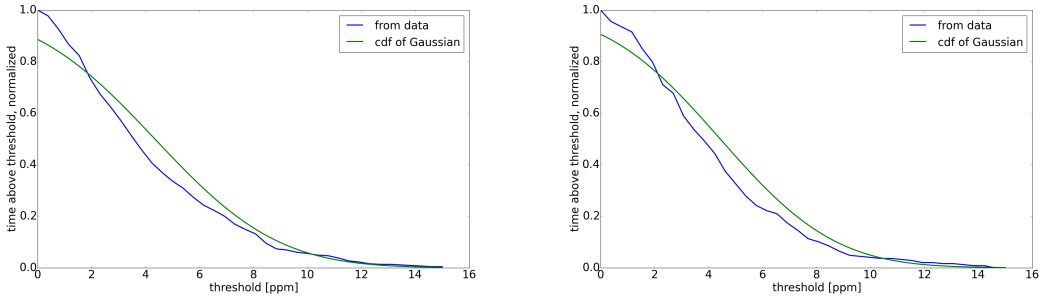
Figure 6.3: PDF at center of mass using 40 bins, using Gaussian fit with parameters found directly from data.

deviation were found to be $\langle c \rangle \approx 4.34$ p.p.m. and $\sigma = 3.79$ p.p.m. The slope on the right side of the mean corresponds approximately to the actual slope of the PDF. It is larger than the actual PDF by a constant number until about 11 p.p.m. Note that the relative weight to the left of the mean is much smaller than that on the long tail on the right. Because negative concentrations are unphysical, the curve is truncated at 0. More importantly, the large errors in the PDF closer to 0 from about 1 standard deviation to the left of the mean are offset by their relatively small weighting in terms of c. In addition, for practical purposes, the CDF will be considered, and it stacks cumulatively from left to right. Thus the large error could in fact serve to minimize error at large distances to the right of the mean. Thus, one can assume that the CDF will show a good fit from a distance to the right of the mean and throughout the distribution tail.

It is known that the integral of a Gaussian distribution gives a form proportional to the conjugate error function. To compare the implications of a Gaussian model, note that

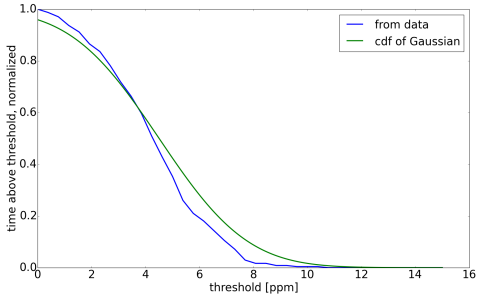
$$t_{norm} = \int_{c_t}^{\infty} p(c)dc,$$

where t_{norm} denotes the percentage time spent above c_t shown in fig.5.4a, made

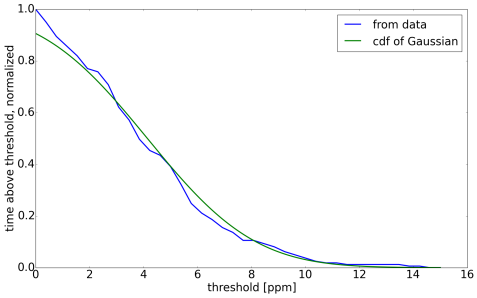


(a) Data set *mad21K*

(b) Data set *mad21G*.



(c) Data set *mad15J*.



(d) Data set *mad21F*.

Figure 6.4: Integrated Gaussian from infinity to threshold at center of mass compared to fraction of time spent above thresholds found from raw data, normalized to a unit instead of in terms of percentages. Note that the integrated Gaussian time fraction does not reach 1 at $c = 0$. This is a result of the Gaussian fit extending left beyond $c_t = 0$. Since the numerical integration is performed from ∞ to c_t, we must also make sure to create a Gaussian fit for a large enough range.

unitless by dividing by 100, since the PDF is normalized according to

$$\int_{-\infty}^{\infty} p(c)dc = 1.$$

t_{norm} is therefore the normalized CDF, corresponding to $p(c)$, of continuous threshold values at a given position. This applies as long as $p(c)$ is sampled along temporal space.

Fig.6.4a shows the fitted CDF compared to $\%t$ from for 512 threshold positions between 0 and 15 p.p.m.

The CDF of three other good sets, *mad21G* (fig.6.4b), *mad15J* (fig.6.4c) and *mad21F* (fig.6.4d), show that the closeness of the Gaussian fit is not isolated to the data set *mad21K*. One could assume that there was something universal for Gaussian fits of the PDF at the center of mass, at least under the general parameters of the MADONA campaign. This is also a reminder that even when measured values that depend on the PDF show distributions that seem to fit well with a known PDF, it is no guarantee that the actual PDF is similar to the synthetic

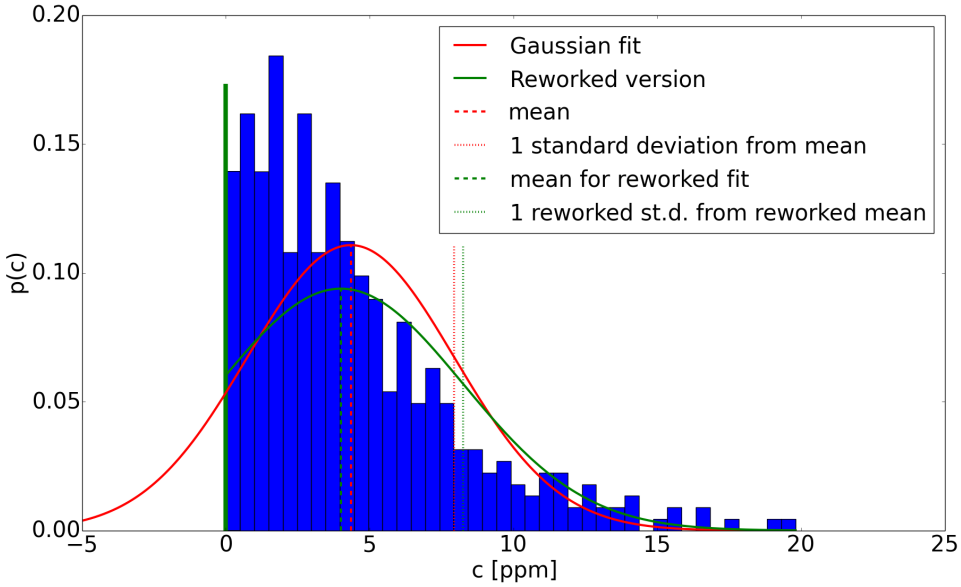


Figure 6.5: *mad21K*: CM frame at $x = R$. Gaussian fit was reworked here to illustrate how a Gaussian tail for negative concentrations could be avoided. The tail is terminated at $c = 0$, where a δ spike is set up instead. We then recalculate a new mean and standard deviation in order to create a Gaussian function which keeps the same distribution when integrated from ∞ to 0. This therefore does not change the results of the fits in fig.6.4a onwards, except that the fitted line would jump suddenly to $time = 1$ at $c = 0$.

model. Consider the threshold points close to 0. In the context of this thesis, if a model was developed that followed a Gaussian distribution because of the generally promising fit, low concentration values would be chronically underestimated. For long periods of time, this underestimation could lead to large-scale, slow poisoning of an entire community.

The Gaussian model can be reworked to get around the negative concentrations. This requires that negative concentrations manifest as a dirac delta spike at $c = 0$. The reworked fit can be seen in fig.6.5.

It should be noted that model fittings of concentration PDFs detected by LIDAR have been made before, for instance by (Munro et al., 2003). In their paper, Munro et al. focused on *maximum likelihood fits*, which yields the best overall form. In our case, the philosophy has been to use fits that emphasize the heavier weighting at concentrations much larger than the mean, but smaller than the single occurrence fluctuations at the tail of the distribution. This allows for a discussion of physical implications in section 6.4, and emphasizes the threshold observations in figs. 6.4a, 6.4b, 6.4c and 6.4d.

6.2 Gaussian Smoothing

The PDFs created from data have a tradeoff between resolution and realism. In order to create PDFs with added detail, but smoother than the jaggedness of high resolution PDFs, synthetic smoothing could be introduced. Arguments here are that since a high resolution introduces artifacts in any case, a lower resolution with synthetic smoothing corresponding to the same level of detail as the higher resolution will only introduce false data on the same order as the errors in the higher resolution PDF. The main reason is to facilitate visual shape estimation, since smooth functions mesh better with the human intuition of shapes, and it is also easier to see whether a given model fit is good if the estimated PDF is smooth.

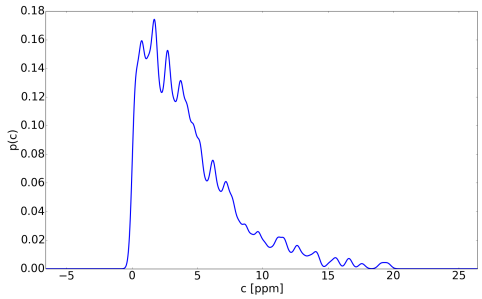
An important criterium for reducing synthetic errors is the shape of the smoothing function. The method of Gaussian smoothing, mainly attributed to Silverman, Wand and Jones, does not change the statistical moments of the PDF, and minimally interferes with the distribution apart from smoothing out the shape. We refer to (Silverman, 1986) and (Wand and Jones, 1995) for detailed discussion of the method. The method has been used in different fields, for instance by our plasma physics group (Larsen, Hanssen, Krane, Pécseli and Trulsen, 2002). The idea is to approach each data point as the mean of a Gaussian distribution, and the distance from a point to the next as the standard deviation, σ. The size of the Gaussian is normalized in $(-\infty, \infty)$ so that its area is the same as that of the bin it represents. The Gaussians superpose when they meet. The result is a PDF that has the same large-scale statistical properties as the original histogram, but is locally smooth. Setting σ to the distance, d, between two histogram bins was found by trial and error to be the best choice. For instance, $\sigma = \frac{d}{2}$ results in a jagged plot. Compare figs. 6.6a and 6.6b, which use $\sigma = \frac{d}{2}$, with figs. 6.6c and 6.6d, which use $\sigma = d$. These 4 plots are for the same histogram, but the smoothing is noticeably more comfortable in the two latter plots.

Note that the method of scaling is not trivial, and there is a discussion of this in (Larsen et al., 2002). In the context of this thesis, the easiest normalization has been used. For a bin of height h and length d,

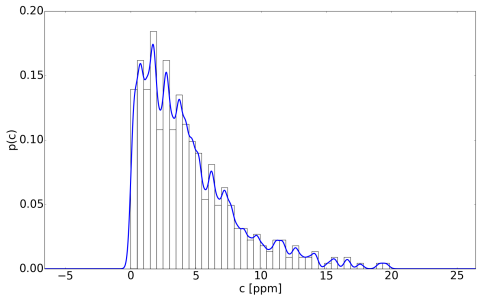
$$hd = \int_{-\infty}^{\infty} \frac{A}{\sigma\sqrt{2\pi}} e^{-\frac{(c-\langle c \rangle)^2}{2\sigma^2}} = A,$$

where A is the scaling parameter.

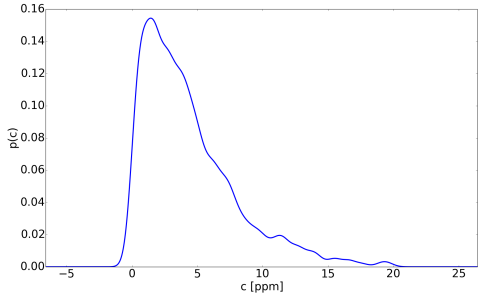
The Gaussian form has a domain 3 times as large as the histogram, a redundancy that guarantees that the leftmost and rightmost tails of the Gaussian superposition are included in the final smoothed distribution. The domain can be shortened if computational expense is too high, and we often truncate it for visualization. However, the Gaussian domain should be large if we want to use the Gaussian smoothed form to compute statistical moments. A loop calls up



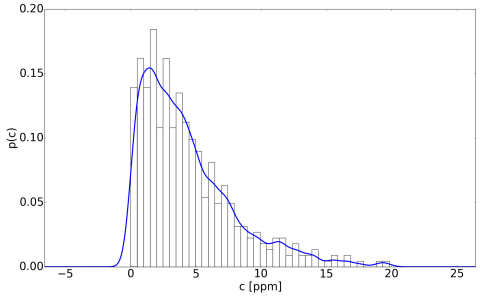
(a) Center of mass for $\sigma = \frac{d}{2}$ without histogram bins.



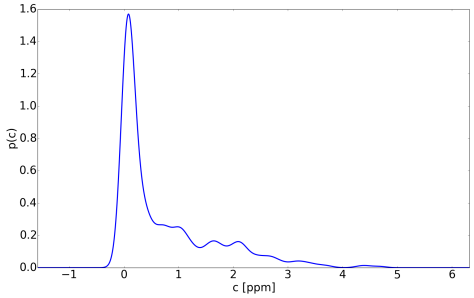
(b) Center of mass for $\sigma = \frac{d}{2}$ with histogram bins.



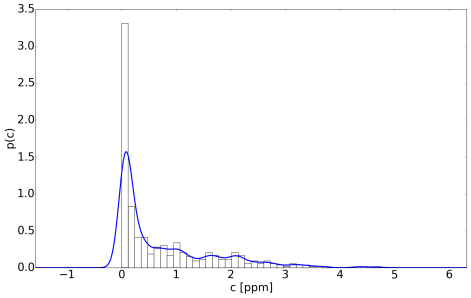
(c) Center of mass for $\sigma = d$ without histogram bins.



(d) Center of mass for $\sigma = d$ with histogram bins.



(e) One standard deviation away from center of mass, for $\sigma = d$, without histogram bins.



(f) One standard deviation away from center of mass, for $\sigma = d$, with histogram bins.

Figure 6.6: All figures above were made for the data set *mad21K* using 40 histogram bins. The small negative concentration region of the envelope is an unavoidable consequence of the smoothing by the superposition of small Gaussian pulses.

and creates a Gaussian for the center of each bin, and the Gaussians are meshed together directly as they are defined on the same domain. Each Gaussian form was arbitrarily given number of points equal to 100 times the number of the histogram bins to ensure a high relative smoothness. The results in figs.6.6c, 6.6d, 6.6e and 6.6f show that when an appropriate scaling has been chosen, Gaussian smoothing is a good way to make the PDF more aesthetically pleasing and easier to visually evaluate, while losing little information in the smoothing process. Note that for figs.6.6e and 6.6f, at one Gaussian standard deviation to the right of the CM, smoothing significantly clarifies the form of the distribution.

Given that part of the point with Gaussian smoothing is to enhance the information in a systematic manner, certain shortcomings must be pointed out. First, there is no justification for a locally Gaussian form of the distribution. Often, concentrations that are close to each other in density are measured right after each other, which in a turbulent system implies that they are locally correlated. If many of the samples in the PDF are made from such points, a locally Gaussian form of each bin becomes unfeasible, since a Gaussian form requires uncorrelated subgroups. For significantly denser PDFs, and when creating PDFs *en masse*, there is also computational expense to be considered. Finally, negative concentrations are unphysical, but unavoidable when using Gaussian smoothing.

6.3 Excess statistics

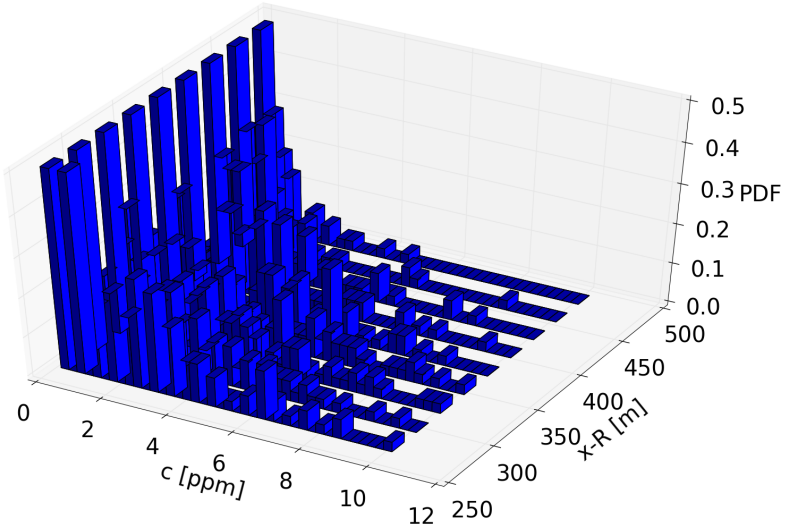
Studies of excess statistics is a general point of interest of all randomly varying signals, also space-time varying concentrations. Basic studies of this problem were carried out by Rice in (Rice, 1944) and (Rice, 1945), with a slightly more accessible disposition by Bendat in (Bendat, 1958). Here, we briefly summarize the arguments.

Consider the expected number of crossings of a level, c_0, in a time interval dt. Defining α equals the expected amount of time spent in the interval dc for a given dc/dt in time dt, and β equals the time τ required to cross once for a given dc/dt in the interval dc. For time stationary random processes,

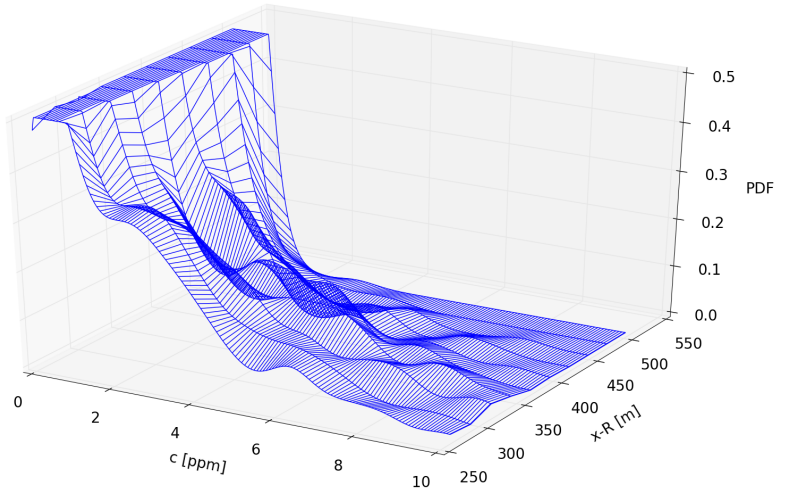
$$n(c_0|dc/dt) = \frac{\alpha}{\beta}.$$

Referring to fig.6.9, we can identify $\beta = dc/c'$ for the given derivative of the concentration signal $c(t)$. The value of α is proportional to dc, dc', and dt, with a constant of proportionality equaling the joint probability density $p(c_0, c')$. By this argument we find

$$n(c_0|dc/dt) = \frac{p(c_0|c')dcdc'dt}{dc/c'} = c'p(c_0, c')dc'dt.$$

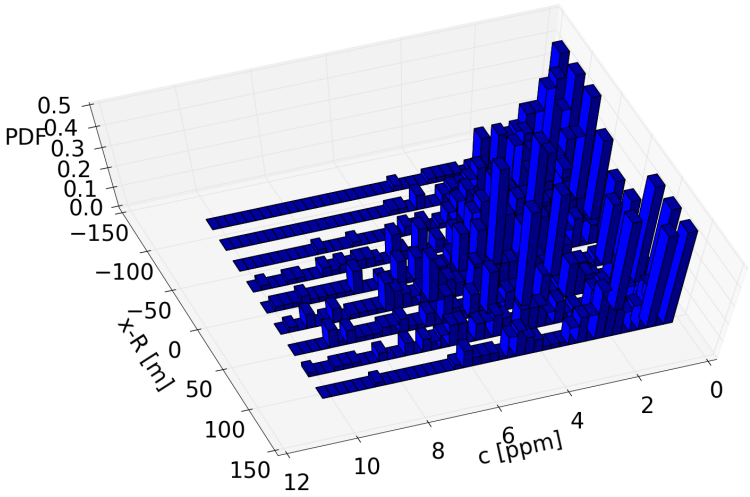


(a) FF-PDF 40 bins.

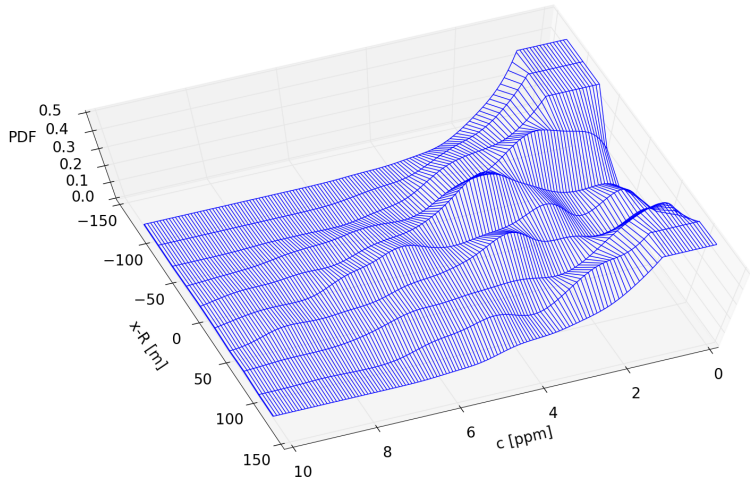


(b) Representation of fig.6.7a using Gaussian smoothing. Cutoff points at c=0 [ppm] and c=10 [ppm] and p=0.5.

Figure 6.7: *mad21K*-CM. Histogram plots without and with Gaussian smoothing, sampled at 10 positions between $x \approx 260$ [m] to $x \approx 502$ [m], around the center of measurement, for 40 bins of threshold concentrations between 0 [ppm] and 10 [ppm]. PDF was capped at p=0.5.



(a) CM-PDF 40 bins.



(b) Representation of fig.6.8a using Gaussian smoothing.

Figure 6.8: *mad21K*-FF. Histogram plots without and with Gaussian smoothing, sampled at 10 positions between $x = R \pm 120$ [m] for 40 bins of threshold concentrations between 0 [ppm] and 10 [ppm]. PDF cutoff at p = 0.5.

This result is conditional in the sense that it refers to one specific value of c'. Results valid for level crossings irrespective of the slope of the signal are obtained by integrating over all positive c', i.e. all upwards crossings. Integration over negative signs would give the same results, since $\langle c' \rangle = 0$ for stationary random processes. This result becomes

$$n(c_0) = dt \int_0^\infty c' p(c_0, c') dc'.$$

This result is directly proportional to the time interval. The level crossing frequency is the number of crossings per time unit, giving

$$\nu = \int_0^\infty c' p(c_0, c') dc'.$$

An average duration $\langle T \rangle$ of excesses can be estimated as the fraction of the time spent above the selected level, and the expected frequency of level crossings (Kristensen, Casanova, Courtney and Troen, 1991):

$$\langle T(c_0) \rangle = \frac{\int_{c_0}^\infty p(c) dc}{\int_0^\infty c' p(c_0, c') dc'},$$

where the fraction of time spent above a selected level is given by $\int_{c_0}^\infty p(c) dc$. The conclusion from the analysis summarized here is that the JPDF, $p(c, c')$, contains all the necessary information to predict the average level crossing frequency and average excess time duration, given a prescribed level c_0.

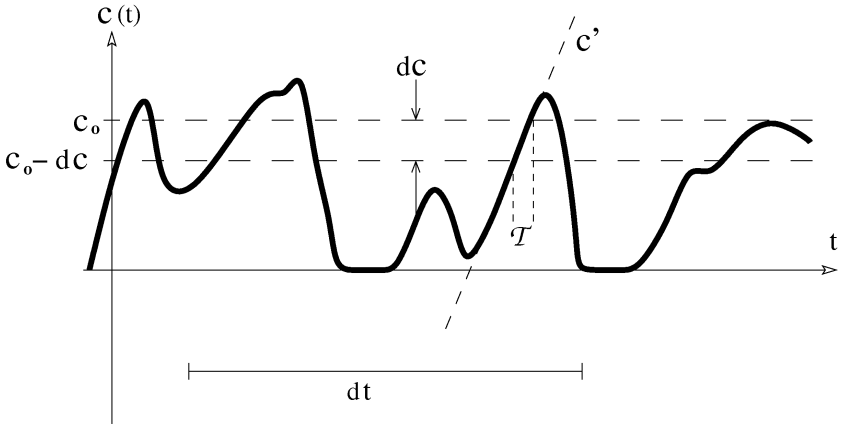


Figure 6.9: Basic Idea for how to use JPDF to estimate $\langle T \rangle$. Made by Bjørn Lybekk. $c' = \frac{dc}{dt}$

6.4 Gaussian Concentration Distribution at Center of Mass

The neat interpolation result for expectation times $\langle T \rangle$ above a continuously varying threshold at the center of mass found in fig.5.9 can be explained if the concentration PDF at the center of mass is assumed to follow a Gaussian form. The time expectation above a given concentration threshold can be estimated as the time spent above the threshold, divided by the number of one-way crossings over the threshold. Formally, this is expressed through the PDF labeled $p(c)$ and the JPDF labeled p(c,c'), where $c' = \frac{dc}{dt}$, and where the threshold is given by c_t, as

$$\langle T \rangle = \frac{\int_{c_t}^{\infty} p(c)dc}{\int_0^{\infty} c' p(c_t, c')dc'}. \tag{6.1}$$

Assuming a Gaussian fit,

$$p(c) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(c-\langle c \rangle)^2}{2\sigma^2}},$$

$$p(c') = \frac{1}{\sqrt{2\pi\sigma'^2}} e^{-\frac{(c')^2}{2(\sigma')^2}}.$$

The JPDF is separable:

$$p(c, c') = p(c)p(c'),$$

since for stationary processes,

$$\langle cc' \rangle = \frac{1}{2}\frac{d}{dt}\langle c^2 \rangle = 0,$$

and uncorrelated Gaussian variables are statistically independent (Pécseli, 2000). This simplifies the expression in eq. (6.1) to

$$\langle T \rangle = \frac{\int_{c_t}^{\infty} p(c)dc}{p(c_t)\int_0^{\infty} c' p(c')dc'}.$$

Inserting for $p(c')$ explicitly,

$$\langle T \rangle = \frac{\int_{c_t}^{\infty} p(c)dc}{p(c_t)\frac{\sigma'}{\sqrt{2\pi}}}.$$

By a change of variables,

$$t = \frac{c - \langle c \rangle}{\sigma\sqrt{2}} \qquad dt = \frac{dc}{\sigma\sqrt{2}},$$

$$\langle T\rangle = \frac{\frac{1}{2}\mathrm{erfc}(\frac{(c_t-\langle c\rangle)}{\sigma\sqrt{2}})}{p(c_t)\frac{\sigma'}{\sqrt{2\pi}}}.$$

It is known that the error function and its conjugate may be series expanded (see e.g. BOAS):

$$\mathrm{erfc}(x) = 1 - \mathrm{erf}(x) \approx \frac{e^{-x^2}}{x\sqrt{\pi}}(1 - \frac{1}{2x^2} + \frac{3}{(2x^2)^2} - \frac{15}{(2x^2)^3} + \dots).$$

The error in the truncation is $O(\frac{1}{c_t^3})$. Therefore, for $|\frac{c_t-\langle c\rangle}{\sigma\sqrt{2}}| >> 1$,

$$\mathrm{erfc}(\frac{c_t - \langle c\rangle}{\sigma\sqrt{2}}) \approx \frac{e^{-(\frac{c_t-\langle c\rangle}{\sigma\sqrt{2}})^2}}{\frac{c_t-\langle c\rangle}{\sigma\sqrt{2}}\sqrt{\pi}},$$

so that

$$\langle T\rangle \approx \frac{\sigma\sqrt{2}}{2\sqrt{\pi}(c_t - \langle c\rangle)} / \frac{\sigma'}{\sqrt{2\pi}},$$

and since it is implicitly assumed that $\frac{c_t}{\sigma} >> \frac{\langle c\rangle}{\sigma}$, in this limit

$$\langle T\rangle \approx \frac{\pi\sigma}{\sigma' c_t},$$

i.e.

$$\langle T\rangle \propto \frac{1}{c_t}.$$

6.5 Joint Probability Density Function

The joint probability density function (JPDF) of the variables $c, \dot{c}$ can be written

$$p(c,\dot{c}) = p(\dot{c}|c)p(c) = p(c|\dot{c})p(\dot{c}).$$

A method of computational differentiation is to use Fast Fourier Transform (FFT). Let F and F^{-1} be the operation for the Fourier transform and its inverse. Let j be the complex unit, and

$$F\{c(t)\} = C(\omega) \qquad c(t) = F^{-1}\{C(\omega)\}.$$

Then,

$$\frac{dc(t)}{dt} = \frac{d}{dt}\left[\frac{1}{2\pi}\int_{-\infty}^{\infty} C(\omega)e^{j\omega t}d\omega\right] = \frac{1}{2\pi}\int_{-\infty}^{\infty} j\omega C(\omega)e^{j\omega t}d\omega = \frac{1}{2\pi}F^{-1}\{j\omega F\{c(t)\}\}.$$

(a) *mad21K*-CM: x=R. JPDF constructed using the 'forward' interpolation from eq.6.2.

(b) *mad21K*-CM: x=R. JPDF constructed using the 'backward' interpolation from eq.6.3.

Figure 6.10: We see that the choice of interpolation direction matters noticeably. There is especially a triangular section without data which flips between the two methods.

However, FFT by its nature jumbles the sampling domain, it can not be used to track the corresponding concentration c at the point. Instead, we use linear interpolation between points to assess $\dot{c}$ here. Since $\dot{c}_i$ is a tangent between the two sample points, we can make the arbitrary choice of associating it with the leftmost or rightmost point. We can estimate

$$\dot{c}(t_i) \approx \frac{c_{i+1} - c_i}{t_{i+1} - t_i} = \frac{1}{3}(c_{i+1} - c_i), \tag{6.2}$$

or

$$\dot{c}(t_i) \approx \frac{c_i - c_{i-1}}{t_i - t_{i-1}} = \frac{1}{3}(c_i - c_{i-1}). \tag{6.3}$$

Using these interpolations, one data point is left blank. We can also use the average of the two methods, which becomes

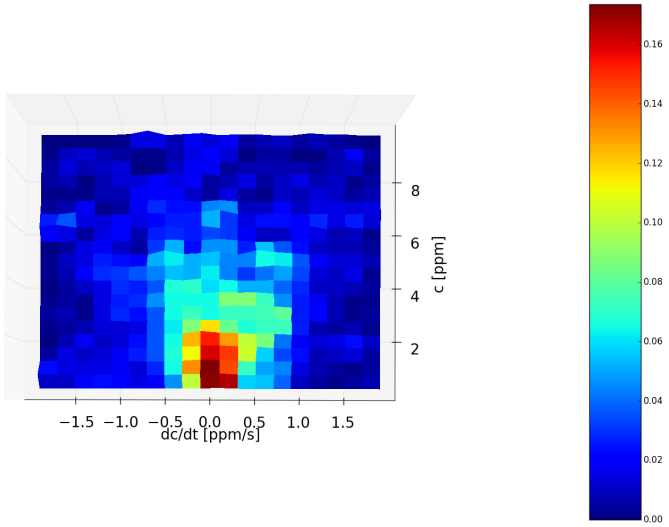
$$\dot{c}(t_i) \approx \frac{1}{2}\left[\frac{c_{i+1} - c_i}{t_{i+1} - t_i} + \frac{c_i - c_{i-1}}{t_i - t_{i-1}}\right] = \frac{1}{6}(c_{i+1} - c_{i-1}). \tag{6.4}$$

This last option in effect introduces a smoothing by overlap. In theory we could calculate the speed without overlap, but this would lower the JPDF resolution to $\frac{1}{4}$ its original, since valid concentration points would also be halved. The JPDF is found by looping over all combinations of histogram bins and seeing if combinations $(c(t_i), \dot{c}(t_i))$ fit any of the squares. All JPDFs have been normalized by volume.

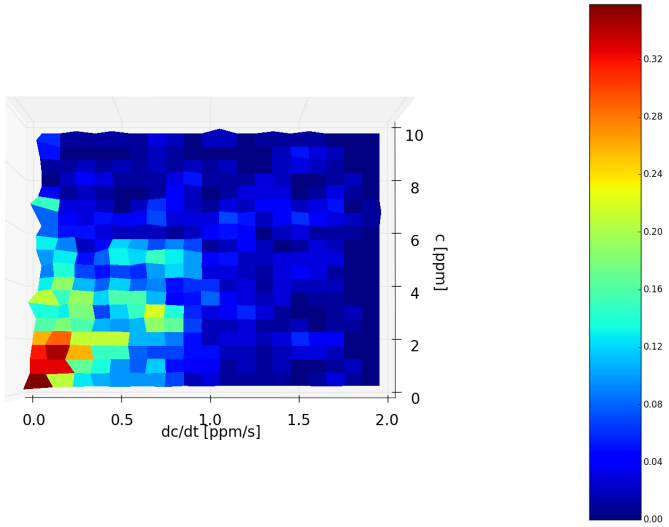
Both eq.6.2 and eq.6.3 were tested, and are shown in fig.6.10a and fig.6.10b.

All the JPDF figures are in CM frame, at x=R. This is because the resolution is already very bad. Instead, we try to establish grounds for generalization. Further steps will be suggested in the conclusion.

There is a noticeable difference in the left- and rightmost bottom triangular sections, corresponding to high speeds and low concentrations. This is because concentration can not go below $0[ppm]$, so a concentration $X[ppm]$ can not be associated with an immediate rate of change which is more negative than $-X/T[ppm/s]$,

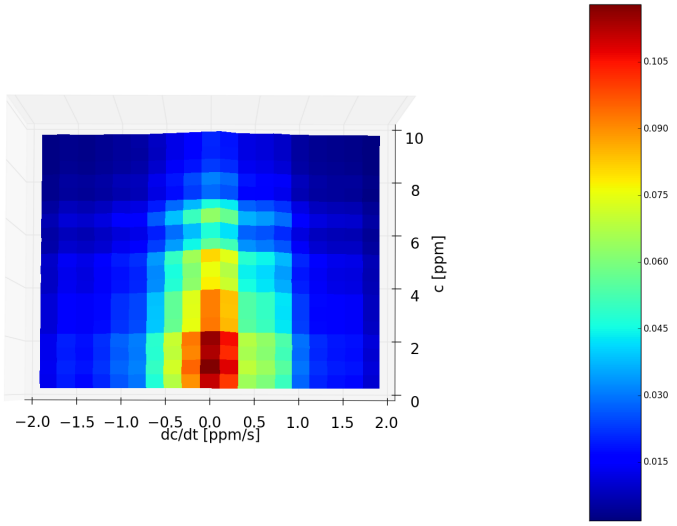


(a) *mad21K*-CM: x=R. JPDF constructed using the interpolation from eq.6.4.

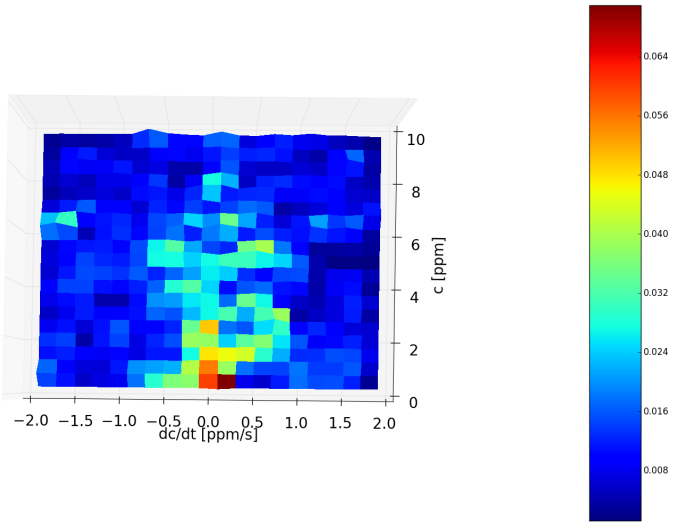


(b) *mad21K*-CM: x=R. JPDF constructed by sampling along abs(dc/dt) for the same resolution. Capped at p = 0.45.

Figure 6.11: The triangular shape of green histogram points was observed for several data sets.



(a) *mad21K*-CM: x=R. Artificial JPDF assuming independence: $p(c, dc/dt) = p(c)p(dc/dt)$.



(b) *mad21K*-CM: x=R. The absolute value difference between the JPDF shown in fig.6.11a and the artificial independent JPDF shown in fig.6.12a.

Figure 6.12: Fig.6.12b shows a clear pattern of correlation in the low velocity sections of the JPDF. However, note that it is barely above the noise level around p = 0.02 in fig.6.11a. Higher velocities show negligible correlation. This accounts for the fact that the Gaussian, uncorrelated PDF(c) model was successful at higher concentrations.

where $T = 3[s]$ is the time resolution of the data sets. This is why the triangles are sharply defined and have a 1:3 ratio on the dc/dt:c dimensions. The blank triangle in fig.6.10a, corresponding to 'forward' interpolation, is according to the above interpretation because it is unphysical for a concentration to jump to a negative concentration, while the blank triangle in fig.6.10b represents the fact that it is unphysical for a concentration to have been arrived at *from* a negative concentration. However, it is not unphysical for the local rate of change to be within these triangles when measured by two points on either side. If we superpose figs.6.10a and 6.10b and divide by 2, we get the JPDF according to eq.6.4.

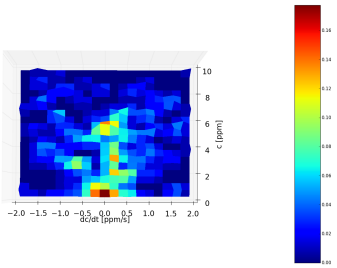
Another problem is the low resolution needed unless we want to introduce additional noise. PDFs were found to have optimal bin size around 40. For the JPDF, maximum bin size without introducing too much noise was found to be around 20. This accounts for the "grainyness" of the figures. We know of three ways to improve the resolution based on the existing data sets.

(i) Use the absolute value of the velocity, superposing the left and right side of fig.6.11a. This allows us to get twice the resolution along dc/dt for half the domain, but sacrifices information about asymmetricality around dc/dt = 0. This is shown in fig.6.11b

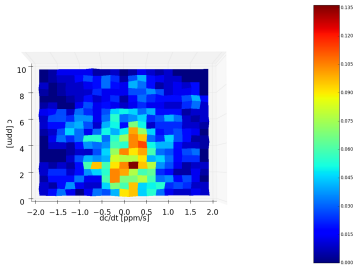
(ii) Superpose the JPDF of the same position in CM frame for different data sets. In this thesis, we have assumed that the turbulent diffusion of the data set concentrations over time is driven by an underlying velocity field which is assumed to be a stationary process. This is no longer true as wind conditions change. We have so far assumed that each data set is driven by such a stationary process, but we might extend this to all data sets from experiments performed on the same day. Figs. 6.13a and 6.13b show that the JPDF is similar in shape on these days. The normalized JPDF of the superposed data is shown in fig.6.13c. Since 4 data sets were used, we used 40 bins along each axis. Each data set was separately scaled according to the Miyake concentration value used throughout this thesis. This was done consciously because release concentrations might have been changed between experiments. The results show more clearly that high concentration oscillations occur more often at higher concentrations, as expected, and the better resolution more clearly shows the shape of the JPDF.

(iii) Use Gaussian smoothing. The script written for section 6.4 was modified to smooth along the c direction of the JPDF. This direction was consciously chosen over dc/dt because the interpolation method we use already smooths by overlap, as mentioned earlier. Figs. 6.14a and 6.14b show the Gaussian smoothed histogram shown in fig.6.11a. Note that Gaussian smoothing does not change the statistics of the structure. However, the smoothing makes the underlying structure clearly discernible.

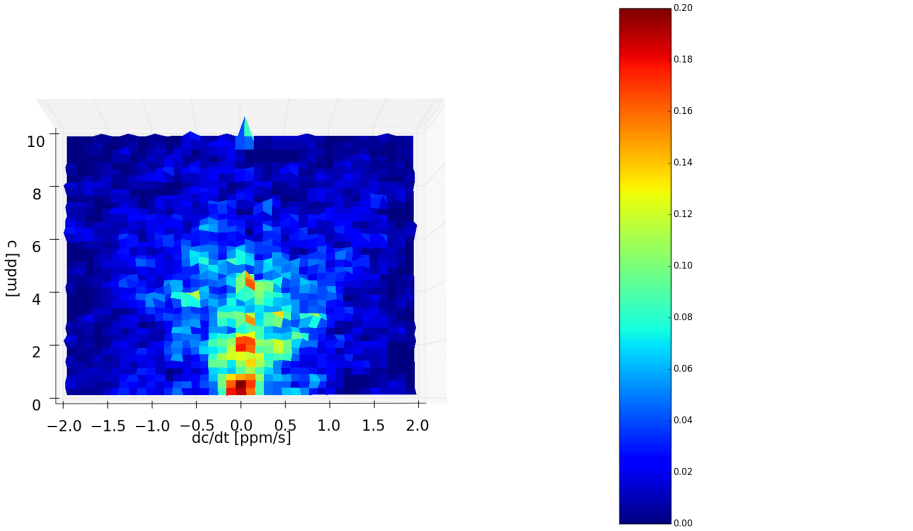
The three methods above all have their drawbacks. Folding the structure (i) re-



(a) *mad21F*-CM: x = R.

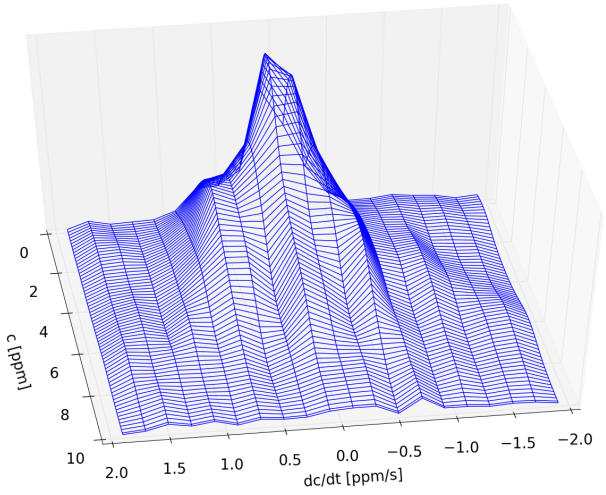


(b) *mad21G*-CM: x = R.

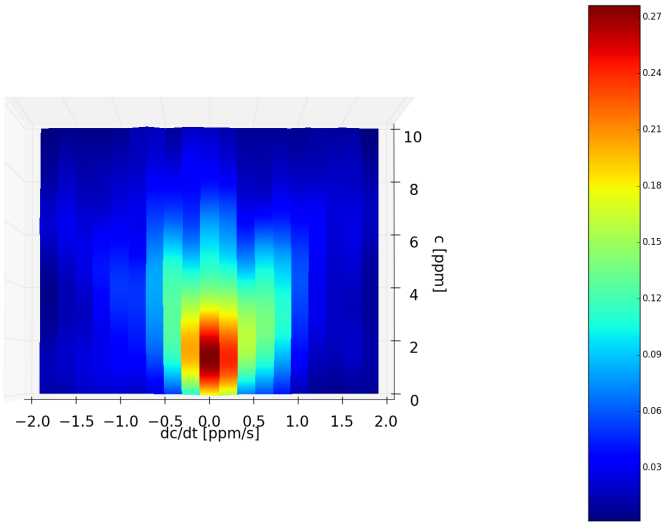


(c) Superposed JPDF in CM frame: x=R.

Figure 6.13: Figs. 6.13a and 6.13b show JPDFs of *mad21F* and *mad21G* respectively. Fig.6.13c shows the JPDF of the superposed data points from sets *mad21K*, *mad21F*, *mad21G* and *mad21H*, all measurements performed on the same day.

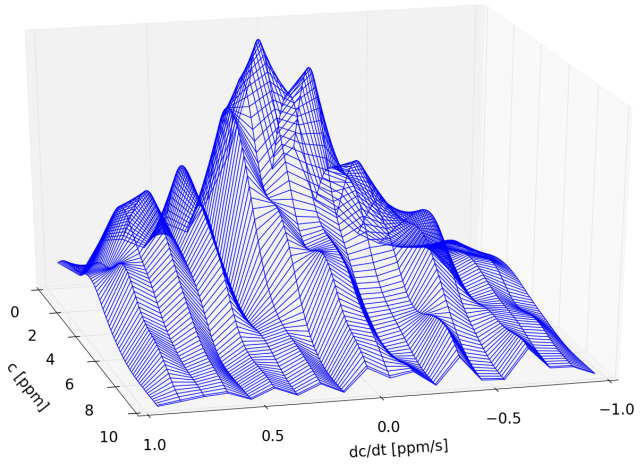


(a) *mad21K*-CM: x = R. Wireframe plot.

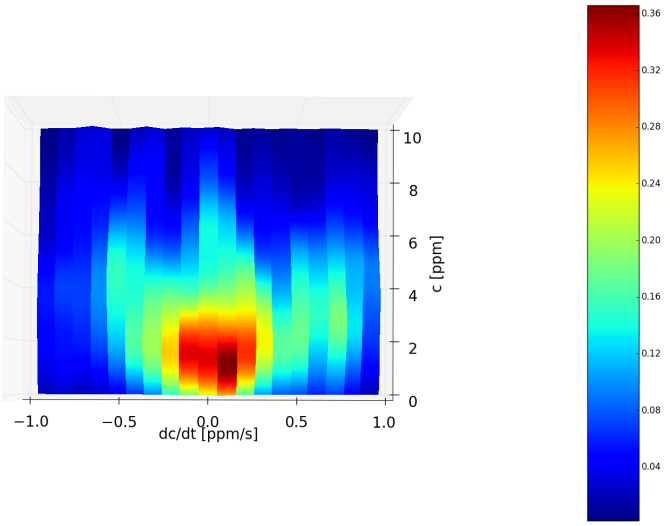


(b) *mad21K*-CM: x=R. Color plot.

Figure 6.14: JPDF using Gaussian smoothing. The shape of the structure is clearly discernible.

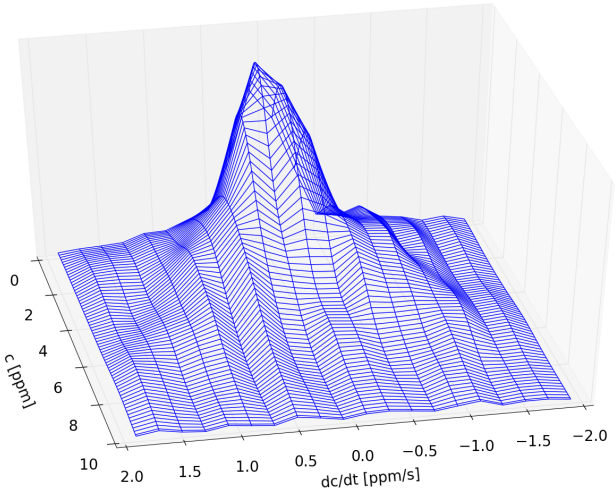


(a) *mad21K*-CM: x = R. Wireframe plot.

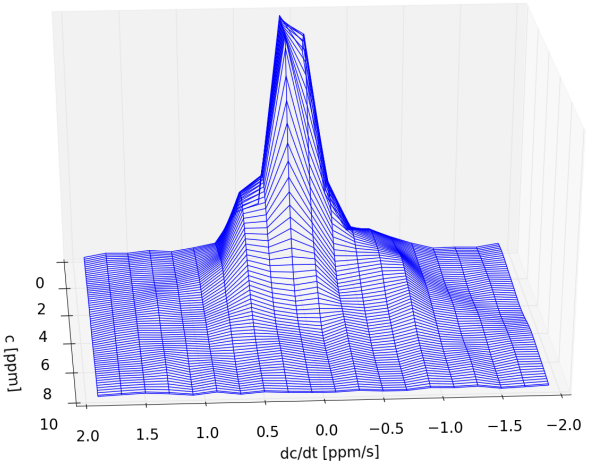


(b) *mad21K*-CM: x=R. Color plot.

Figure 6.15: JPDF using Gaussian smoothing. The same amount of bins as previously, 20 by 20, but constrained along $dc/dt = -1$ [ppm/s] to $dc/dt = 1$ [ppm/s].



(a) *mad21K*-CM: x = R + 30 [m].



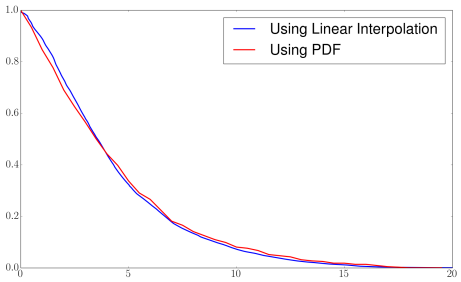
(b) *mad21K*-FF: x=382.5 [m]. Center of measurement.

Figure 6.16: JPDF using Gaussian smoothing at a distance away from R in CM frame, as well as the center of measurement in FF.

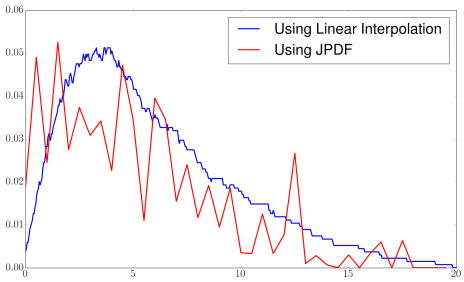
moves information about concentration change direction. Using more data sets (ii), even from the same day, increases the chance of the wind being a non-stationary process. The Gaussian method (iii) smooths over the local structure. It would therefore be inadvisable to use several of the methods at the same time without being aware of what each does to the data. For instance, figs.6.15a and 6.15b show the JPDF with the same amount of bins, but for $dc/dt = [-1, 1]$ [ppm/s] instead of $dc/dt = [-2, 2]$ [ppm/s]. It would be tempting to suggest that this shows more of the JPDF structure than the lower resolution at $[-2, 2]$ [ppm/s] if we did not already know that 20 by 20 bins is about as good a resolution as we can get without introducing a lot of noise. Each of the methods also gives us different information. Folding the structure (i) oversamples the symmetrical parts of the structure, and a triangular structure with vertices at (0,0), (1,6) and (0,6) is more clearly visible. The same structure can be seen in the superposition of the 4 JPDFs in fig.6.13c (ii). The superposition of 4 JPDFs shows the complexity of the problem. It also separates more clearly the sections that arise from a stationary process, because the rest will manifest as noise. Finally, the Gaussian smoothing (iii) allows us to see the structure of the JPDF. We notice three fingers at high concentrations, and the structure is seen to be more symmetric about $dc/dt = 0$ than the histogram JPDF in fig.6.11a seems to suggest. Fig.6.14a also shows the outline of the triangular structure mentioned above. Gaussian smoothing also allows us to estimate the shape of positions away from R in CM frame, as well as FF positions, as long as we remember that the actual resolutions here might be bad. Fig.6.16a shows an example of the former, for the CM frame at $x = R + 30$ [m], while fig.6.16b shows an example of the latter, with a JPDF in FF at $x = 382.5$ [m].

It is useful to know whether the JPDF is independent. As shown in section 6.4, it allows for a simplification in the expression for $\langle T \rangle$, and also means that we can use FFT to compute $p(dc/dt)$, since we do not have to keep track of position along the sampling domain. Independence means that $p(c, dc/dt) = p(c)p(dc/dt)$. One condition of independence is that $\langle c\frac{dc}{dt} \rangle = 0$. However, for a stationary process this is automatically fulfilled by $\frac{1}{2}\langle (\frac{dc}{dt})^2 \rangle = 0$. The other condition is that the underlying distribution is Gaussian. We found $p(dc/dt)$ using eq.6.4. This artificial, independent JPDF can be seen in fig.6.12a. The absolute value of its difference with fig.6.11a can be seen in fig.6.12b. There is a clear correlation pattern near low velocities which decreases for higher concentrations. This means that the independence assumption is better for higher concentrations. This argument is backed by the results of section 5.2, where we found the cumulative distribution (CDF) of a Gaussian PDF to be a good fit for higher concentrations.

Another reason to find the JPDF is to separate it into $p(\dot{c}|c) = p(c, \dot{c})/p(c)$. If this could be generalized, we could use measurements of $p(c)$ alone in other experiments to generate JPDFs. However, the best resolution of our JPDFs seem



(a) *mad21K*-CM frame: $t(c_t)_{excess}$ found from $p(c)$ compared to the same value found by direct sampling along c_t.



(b) *mad21K*-CM frame: $\nu(c_t)$ found from $p(c)$ compared to the same value found by direct sampling along c_t.

Figure 6.17

insufficient for such a generalization.

6.6 JPDF Estimate of Excess Statistics

Section 6.3 explains the method used to find theoretical estimates of excess statistics. In section 6.1 we showed that a Gaussian fit to the PDF gives a surprisingly good estimate for time spent above a given threshold. Given the bad resolution of the JPDF, it is inadvisable to adopt a similar philosophy here as well. Higher values of both concentration and concentration fluctuations dc/dt have considerable fluctuations which is often indistinguishable from the background noise levels. Instead, we estimate $\%t$ and ν_{excess}, the normalized version of crossing frequency, directly from the normalized $p(c)$ and $p(c, c')$, where $c' = dc/dt$. We have that

$$t(c_t)_{excess} \approx \sum_{c_i=c_t}^{c_i=c_{max}} p(c_i)\Delta c,$$

for the fractional excess time,

$$\nu(c_t) \approx \sum_{c'_i=0}^{c'_{max}} c'_i p(c_t, c'_i)\Delta(c, c'),$$

for the frequency of crossings at concentration thresholds, and

$$\langle T(c_t)\rangle \approx \frac{t(c_t)_{excess}}{\nu(c_t)}$$

for the expected excess time at c_t. These can be plotted against c_t and compared to earlier results from sampling the data directly through concentration thresholds.

$t(c_t)_{excess}$ is reproduced and compared against the earlier results at the CM in fig.6.17a. As expected, the fit shows only slight variation from the directly sampled results, due to the effects of binning. Fig.6.17b shows the reproduced ν made from the JPDF in fig.6.11b. We chose to use the "folded" JPDF because the experimentally obtained $p(c, dc/dt)$ is not perfectly symmetric about $dc/dt = 0$, even for CM frame. However, the results show that the estimate does not reproduce f_{cross} well. Only the general form of increasing towards lower concentrations, and then decreasing again close to $c_t = 0$, was preserved. The heavy fluctuations are due to the low resolution of the JPDF, while the generally bad fit can be attributed to bad dc/dt estimation. This could also mean that "folding" the JPDF is a bad idea, since the concentration of a diffusive plume may have a preference for the direction of its fluctuation at that point.

We have in any case hit a limit in what these LIDAR data sets are capable of *on their own*. Ways to extend their usefulness are discussed in sec.7.2.

Chapter 7

Conclusion

> Academic disciplines can define themselves either by their objects of study or by their style of inquiry. Physics is firmly in the second camp. Physicists make it their business to ask certain kinds of questions about Nature and to seek certain kinds of answers... "thinking like a physicist" is supposed to mean something...
>
> *William Bialek*
> Biophysics: Searching for Principles

7.1 Summary and Discussion

The aim of this project has been to study the statistical properties of LIDAR measured concentration data sets in order to extend them to discussions of released toxic contaminants diffused within the atmospheric boundary layer. The analysis was fruitful, although generalization through probability density functions met limitations. These limitations can be overcome by doing experiments geared towards the analysis presented in this thesis. The analysis in the present thesis demonstrates that meaningful results can be obtained by use of LIDAR data obtained from turbulent atmospheres, in spite of the spatial averaging by the laser pulse and the potentially reduced (compared to point measurements) temporal sampling frequency. The remote sensing without physical objects disturbing the

local flow conditions makes the LIDAR technique appealing in many other aspects, and it is comforting to note the good quality of the resulting data.

7.1.1 Comparison with realistic pollution measurements

Figure 7.1: Miyake residents watching a gas eruption from afar without gas masks. Our analysis suggests that this can be a bad idea, since large concentration fluctuations can be experienced even at large distances from a source, even though this happens rarely. [1]

Using SO_2 concentration and medical data collected on Miyake island, we found that an approximate CM to CM scaling according to time-averaged point measurement concentrations on the island could be performed. This scaling allowed us to check that CM concentration measurements regularly exceeded the concentration threshold at $c_t = 5$ [ppm] corresponding to the 15 [min] exposure limit. According to results in table 1.2, a 44 [min] stay in the CM of such a plume is sufficient to damage a person's lungs. The "brute force" calculations also predicted high levels of concentration fluctuations at distances up to 2σ away from the CM for low concentration thresholds. According to results in table 1.2, a stay of 5 hours and 25

minutes as far away as 2σ to the right of the CM is sufficient to exceed the exposure limit. This can easily be achieved by long term residents of the area. Medical data from Miyake suggested that the residents suffered from long term effects of SO_2 pollution despite competency in using gas masks during SO_2 outbreaks. This suggests that high fluctuations for low concentrations at large distances away from the CM of the plume may be to blame, and is not something a point measurement (in space and time) tool could pick up on by itself.

Similar but less dramatic problems, such as smell pollution from farms or industry, can also be studied by similar methods (Nielsen et al., 2002), (Mikkelsen and Jørgensen, 2002).

7.1.2 Analytical and Data Processing Tools and Methods

We looked at analytical concepts in fluid dynamics and turbulence and found that the complexity of the issue requires a statistical approach to the problem. Our analytical survey of correlated and uncorrelated diffusion suggested that turbulent – unlike classical – diffusion processes, are not easily computationally or analytically solved, for instance through random walk simulations. Our statistical analysis for the plume mean square width suggested that at least 4 of the 5 best data sets we had to choose from, including our primary choice *mad21K*, corresponded to turbulent diffusion. We therefore rely instead on experimentally derived statistical distributions, such as the PDF $p(c)$ and JPDF $p(c, dc/dt)$ found during this project.

The three quantities of particular interest to us were percentage time ($\%t$) above c_t, crossings (f_{cross}) over c_t, and the expected time ($\langle T \rangle$) above c_t for each crossing. Tools were developed for visualization and fitting of continuous thresholds and continuous positions. We discussed the effect on $\%t$ above c_t of linear interpolation as opposed to simple counting (sec.5.1). We concluded that we had no basis for using "fancier" interpolation techniques.

Due to the rough nature of the scaling, and the fact that our ultimate goal is to generalize our findings, only quantitative results with qualitative significance were discussed. In particular, we found that the form of $\langle T \rangle$ for the CM in CM frame and for measurements near the center of measurement for FF had a power-law scaling close to $\langle T \rangle \propto \frac{1}{c_t}$. This scaling becomes noticeably better for higher concentration thresholds. We attribute this to statistical independence between c and dc/dt for higher concentrations. Linear fits in log and log-log space showed clear sections of flat and linear scaling, as well as noise. Truncations along c_t for upper and lower limits of the fits were evaluated on a case by case basis.

A three dimensional visualization of the crossings showed a "forking" (e.g.

[1] http://www.accidental.tv/images/miyakejima_watching.jpg

fig.5.13a) of high fluctuations as the concentration threshold was lowered. This result confirms earlier suspicions from the "brute force" calculations at 2σ. The analysis suggests that future studies should compare against a relation between continuous concentration thresholds and exposure limit for known contaminants. These continuous relations can be worked out in a multi-disciplinary effort between medical expertise and experimentally obtained data of toxic releases carried by the atmosphere, especially at low average concentrations. The "forking" structure was observed for both CM frame and FF. Results in FF had a random asymmetry as compared to the CM frame, as expected, due to the limited sampling time. The statistical variability in the FF will always be larger than the moving CM frame. An illustrative case supporting this argument is a limit where a fixed, compact plume is moved by a random velocity field: following the plume we will find no statistical variability, while a fixed observer would find a randomly varying concentration field.

7.1.3 Generalization and Limitations

We considered limitations of LIDAR measurements of concentration fluctuations. Our analytical discussion of atmospheric stability and the atmospheric boundary layer concluded that LIDAR measurements must be done during stable atmospheric conditions and kept as horizontal as possible due to the logarithmic wind profile in the atmospheric boundary layer, and how it affects local turbulence fluctuations. This is especially important if we want our concentration samples over time to reflect the underlying statistical ensemble for constant parameters throughout the LIDAR line of sight.

We looked at relations between variance, skewness and kurtosis in FF and CM frame (sec.3.2), and found a systematic relation between skewness and kurtosis. Data points were constrained by the known relation $\alpha_4 \geq \alpha_3^2 + 1$, where equality corresponds to a single-concentration release situation known as the "top hat" model. Another relation between deviation from $\alpha_4 = \alpha_3^2 + 1$ and distance from center of mass is suggested by the data. Variance and skewness was also found to be constrained by the single-concentration release model, but no clear relation was found beyond this. This makes sense because high kurtosis distributions have a thick tail, and this is related to the skewness of the plot, since an asymmetrical distribution has a thicker tail on one side. Variance is a measure of the thickness of the spread, and does not heavily rely on asymmetry.

We put occurrences of different concentrations over time into histogram bins and treated these as a an estimate of the underlying true PDF of concentrations. A good resolution was again found through trial and error, and discussions with supervisor. As a test of our analysis for $\langle T \rangle \propto \frac{1}{ct}$, we fitted a Gaussian form of the same mean and standard deviation as found directly from the data. The

Gaussian fit was bad in PDF space, and also had probability densities associated with negative concentration, which is unphysical. However, integrating the fitted PDF in fig.6.3 and comparing against $\%t$ yielded surprisingly good fits, especially for data set *mad21F*.

In order to facilitate visualization of PDFs, and in anticipation of the JPDFs, we developed a one dimensional Gaussian smoothing algorithm. It was retroactively fitted to the histograms showing the meandering of the CM in sec.4.5, where it helped give an "intuition" that *mad15J* was the best candidate for reaching the limiting case of classical diffusion, as established in sec.4.2. Several 3D visualizations of $p(c, x)$ were generated, using Gaussian smoothing along c. These showed that the PDF typically peaked close to 0, except for regions close to the CM, and only in CM frame. For FF, we saw that peak occurrences always occurred for lower concentrations than in the CM frame.

In order to generalize the results based on our knowledge of excess statistics, we found the JPDF $p(c, dc/dt)$ by binning according to both c and dc/dt. dc/dt was found using linear interpolation. The best possible resolution without introducing considerable noise was found to be 20 by 20 bins. We tried three ways of enhancing this resolution: considering only absolutes of velocities, using several data sets from the same day, or using Gaussian smoothing. Again, each method had its own interpretation. Gaussian smoothing, in particular, gave satisfying representations, but was capable of misleading someone who did not know the actual resolution of the binning. We also found that the independence assumption $p(c, dc/dt) = p(c)p(dc/dt)$ became better as concentration increased.

We were not able to reproduce f_{cross} from the excess statistics. We attribute this to the fact that our dc/dt calculations are an order worse than the measured $c(x, t)$. The f_{cross} reproduced through the JPDF was found to loosely follow the shape made from direct sampling of the data, but oscillated greatly. In comparison, the reproduced $\%t$, which depends on $p(c)$ only, was a much better fit compared to direct sampling. We therefore conclude that LIDAR backscatter data alone are not capable of generalizing $p(c, dc/dt)$, which is needed for a full discussion of excess statistics. It has, however, proven to be extremely valuable in creating generalizable statistics on the concentration level, as well as serving as a template for analysis given point measurements samples. Gaussian smoothing was also found to be especially useful by facilitating estimation of histogram shapes by the human eye.

By inspection of data like those in fig.1.6a, we noted a systematic enhancement of the density downstream of the LIDAR beam, giving rise to a slight "skewing" for the concentration profile. We attribute this effect to a slight error in the correction formula for the LIDAR intensity depletion as a pulse passes through the scattering cloud. This error has a slight consequence for low concentration

threshold samplings, because the "skewing" manifests as artificial fluctuations at the cloud tail away from the LIDAR. Future use of the experimental setup for these studies should make corrections for this, with reference to (Jørgensen et al., 1997).

7.2 Future Perspectives

Work on this project has been constrained both by the 1 year allotted to my Master's thesis, as well as by the focus on LIDAR data alone. Here, the focus has been on data set *mad21K*.

(i) Additional work that could be done in a later project include performing the same analysis for each of the other 4 good data sets. Based on the results from sec.4.5, data set *mad15J* has a plume width closer to the classical diffusion scale. A random walk simulation could be performed to compare against the data in CM frame, and extended to account for the wind through comparisons in FF. Model fittings could be performed for f_{cross} in particular. It would be interesting to find an *analytical* explanation for the "fork" shape found in for instance fig.5.13a. In this thesis, we have chosen to limit fits to where we could give a testable physical interpretation. An extension of the project could be to find the mathematically *closest* fit (without worrying about physical interpretations) for PDFs and the quantities f_{cross}, $\%t$ and $\langle T \rangle$, and see for instance how the integration of $p(c)$ for such a fit is compared with $\%t$. Quantities other than f_{cross}, $\%t$ and $\langle T \rangle$ can also be interesting for discussions of toxic contaminants. For instance, *local* fluctuations, as opposed to fluctuations *above* a threshold, could be computed between *two* variable concentration thresholds. The data could then be sampled for fluctuations of variable locality. Further analysis of the data sets could also include conditional sampling. For instance, we can try out thresholds to the JPDF "hits" to remove noise, or we can sample lower concentrations with smaller bins, in order to maximize the overall resolution. We can also perform cross-parametric conditional sampling, setting a minimum dc/dt in order to represent only data points corresponding to large fluctuations in concentration. This can separate *real* combinations of high c and dc/dt from noise in the JPDF. This can also be used to study the effects of high dc/dt on f_{cross}.

(ii) The main constraint in this project has been the low resolution of the JPDF. As long as dc/dt must be estimated from $c(t)$, p(dc/dt) will have a resolution one order worse than $p(c)$. The constraint is therefore due to only using LIDAR backscatter data. It could be interesting to have one Wind LIDAR for estimating dc/dt through the underlying wind velocity, together with a LIDAR measuring concentration. LIDAR data seem to also be capable of representing concentration fluctuations which are usually obtained by point measurements. Localized detectors could therefore be added as a comparison to such an experiment.

These could determine the effects of the spatial averaging of the LIDAR pulse, as well as developing more sophisticated ways to compare LIDAR data with the point measurement devices normally installed near natural toxic contaminant sites. This would also ascertain how best to *scale* concentrations measured by single-positional measurement devices to LIDAR data of similar plumes. With better resolution in the JPDF, we could also find the c_t where the statistical independence of c and dc/dt starts to "branch off", and see if this is a universal relation.

(iii) Additional experiments could vary height in order to see the effects of the logarithmic boundary layer on atmospheric diffusion. If several LIDAR scans could be performed simultaneously at different distances downstream on the same plume, qualitative changes in the concentration distribution could be found. These can advise on the viability of LIDAR data fits to specific cases of toxic contaminants if the distance from the source is known in both cases.

(iv) Sec.3.2 concluded with the skewness and kurtosis of flank positions of concentration distributions exhibiting closer fit to the "top hat" model. Experiments could find out if this was due to the initial mixed state of the aerosols, or if it is due to the flank concentrations in fact becoming more "settled". Could relationships between kurtosis, skewness and standard deviation be used to estimate distance from the CM of a *single* data point? Such an experiment could be performed for LIDAR scans in conjunction with point measurement devices. It would also be interesting to note if especially the relationship between skewness and kurtosis changes systematically downstream.

(v) Finally, the possibility of extending the analysis to *industrial* SO_2 releases can be considered. Industrial smog is a large problem in urban centers in for instance China, and the release of SO_2 plumes from factory pipes is a large contributing factor. Such a comparison means we have to take into account urban roughness levels and/or higher vertical position of plumes. Especially $\%t(c_t)$ could be used to compare against medical estimates of exposure limits for continuous concentration thresholds. $\%t$ could then be scaled according to total time under a toxic plume, and an estimate of the amount of time needed for e.g. 99% of the medical estimate to be covered by $\%t$ curve could be found. This would be a multi-disciplinary effort, as noted above, and could also be a multi-national effort countries struggling with pollution could be interested in.

Bibliography

Acheson, D. J. (1990). *Elementary Fluid Dynamics*, Oxford University Press.

Batchelor, G. K. (1950). The application of the similarity theory of turbulence to atmospheric diffusion, *Quarterly Journal of the Royal Meteorological Society* **76**(328): pp. 133–146.
URL: *http://onlinelibrary.wiley.com/doi/10.1002/qj.49707632804/abstract*

Batchelor, G. K. (1993). *The Theory of Homogeneous Turbulence*, Press Syndicate of the University of Cambridge.

Batchelor, G. K. (2000). *An Introduction to Fluid Dynamics*, Cambridge University Press.

Bendat, J. S. (1958). *Principles and Applications of Random Noise Theory*, John-Wiley & Sons, New York.

Bergsaker, A. S. (2012). *Anomalous transport in a toroidal plasma*, Master's thesis, Universitetet i Oslo.
URL: *https://www.duo.uio.no/handle/10852/34695*

Brenna, H. (2013). *Two dimensional vortex structures in magnetized plasmas*, Master's thesis, Universitetet i Oslo.
URL: *https://www.duo.uio.no/handle/10852/37158*

Cionco, R. M. et al. (1999). An overview of MADONA: A multinational field study of high-resolution meteorology and diffusion over complex terrain, *Bulletin of the American Meteorological Society* **80**(1): 5–19.
URL: *http://dx.doi.org/10.1175/1520-0477(1999)080¡0005:AOOMAM¿2.0.CO;2*

Einaudi, F. and Finnigan, J. J. (1993). Wave-turbulence dynamics in the stably stratified boundary layer, *J. Atmos. Sci.* **50**: 1841–1864. doi:10.1175/1520-0469(1993)050¡1841:WTDITS¿2.0.CO;2.

Feriet, J. K. and Pai, S. I. (1954). Introduction to the statistical theory of turbulence. i, *Journal of the Society for Industrial and Applied Mathematics* **2**(1): pp. 1–9.
URL: *http://www.jstor.org/stable/2099096*

Finnigan, J. J., Einaudi, F. and Fua, D. (1984). The interaction between an internal gravity wave and turbulence in the stably-stratified nocturnal boundary layer, *J. Atmos. Sci.* **41**: 2409–2436. doi:10.1175/1520-0469(1984)041¡2409:TIBAIG¿2.0.CO;2.

Gnedenko, B. V. (1997). *Theory of Probability*, Overseas Publishers Association.

Hazeltine, R. D. and Waelbrock, F. L. (1998). *The Framework of Plasma Physics*, Perseus Books.

Hines, C. O. (1960). Internal atmospheric gravity waves at ionospheric heights, *Canadian J. Phys.* **38**: 1441–1481. doi:10.1139/p60-150.

Iwasawa, S. et al. (2009). Effects of SO2 on Respiratory System of Adult Miyakejima Resident 2 Years after Returning to the Island, *Journal of Occupational Health* **51**: 38–47.
URL: *http://www.ncbi.nlm.nih.gov/pubmed/18987426*

Jørgensen, H. E., Mikkelsen, T., Streicher, J., Herrmann, H., Werner, C. and Lyck, E. (1997). Lidar calibration experiments, *Applied Physics B* **64**: 355–361. doi:10.1007/s003400050184.

Jørgensen, H., Mikkelsen, T. and Pécseli, H. (2010). Concentration fluctuations in smoke plumes released near the ground, *Boundary-Layer Meteorology* **137**(3): 345–372.
URL: *http://dx.doi.org/10.1007/s10546-010-9532-x*

Kolmogorov, A. N. (1991). The local structure of turbulence in incompressible viscous fluid for very large reynolds numbers, *Proceedings: Mathematical and Physical Sciences* **434**(1890): pp. 9–13.
URL: *http://www.jstor.org/stable/51980*

Kristensen, L., Casanova, M., Courtney, M. S. and Troen, I. (1991). In search of a gust definition, *Boundary-Layer Meteorol.* **55**: 91–107. doi:10.1007/BF00119328.

Landau, L. D. and Lifshitz, E. M. (1976). *Course of Theoretical Physics, Volume 1: Mechanics*, Elsevier Butterworth-Heinemann.

Landau, L. D. and Lifshitz, E. M. (1980). *Course of Theoretical Physics, Volume 5: Statistical Physics*, Elsevier Butterworth-Heinemann.

Landau, L. D. and Lifshitz, E. M. (1989). *Course of Theoretical Physics, Volume 6: Fluid Mechanics*, Elsevier Butterworth-Heinemann.

Larsen, Y., Hanssen, A., Krane, B., Pécseli, H. L. and Trulsen, J. (2002). Time-resolved statistical analysis of nonlinear electrostatic fluctuations in the ionospheric E region, *J. Geophys. Res.* **107**: 1005. doi:10.1029/2001JA900125.

MET (2014). Beaufort wind force scale, `http://www.metoffice.gov.uk/weather/marine/guide/beaufortscale.html`. U.K. Meteorological Office.

Mikkelsen, T. and Jørgensen, H. E. (2002). Spredning af lugt fra svinestalde meteorologi- og lidar feltmålinger fra fuldskala tracerdiffusionsforsøg fra stald ved Roager i juni 1999 og 2000, *Technical Report Risø-R-1325(DA)*, Risø National Laboratory, Roskilde, Denmark. ISBN: 87-550-3016-5.

Munro, R., Chatwin, P. and Mole, N. (2003). A concentration pdf for the relative dispersion of a contaminant plume in the atmosphere, *Boundary-Layer Meteorology* **106**(3): 411–436.
URL: *http://dx.doi.org/10.1023/A%3A1021209622648*

NAAQS (2014). National ambient air quality standards, `http://www.epa.gov/airquality/sulfurdioxide/`. United States Environmental Protection Agency.

Nielsen, M., Chatwin, P., Jørgensen, H. E., Mole, N., Munro, R. and Ott, S. (2002). Concentration fluctuations in gas releases by industrial accidents: final summary report, *Technical Report COFIN project*, Risø National Laboratory, Roskilde, Denmark. pp. 19.

Pearson, K. (1916). Mathematical Contributions to the Theory of Evolution. XIX. Second Supplement to a Memoir on Skew Variation, *Royal Society of London Philosophical Transactions Series A* **216**: 429–457.

Pécseli, H. L. (2000). *Fluctuations in Physical Systems*, Cambridge University Press.

Pécseli, H. L. (2013). *Waves and Oscillations in Plasmas*, CRC Press.

Pécseli, H. L. and Trulsen, J. (1997). Eulerian and Lagrangian velocity correlation in two-dimensional random geostrophic flows, *J. Fluid Mech.* **338**: 249–276. doi:10.1017/S0022112097004904.

Pope, S. B. (2000). *Turbulent Flows*, Cambridge University Press.

Reynolds, O. (1895). On the dynamical theory of incompressible viscous fluids and the determination of the criterion, *Philosophical Transactions of the Royal Society of London. A* **186**: pp. 123–164.
URL: *http://www.jstor.org/stable/90643*

Rice, S. O. (1944). Mathematical analysis of random noise, I, *Bell System Tech. J.* **23**: 282–332. reprinted by Wax, N. in: Selected Papers on Noise and Stochastic Processes (1954) Dover, New York.

Rice, S. O. (1945). Mathematical analysis of random noise, II, *Bell System Tech. J.* **24**: 46–156. reprinted by Wax, N. in: Selected Papers on Noise and Stochastic Processes (1954) Dover, New York.

Richardson, L. F. (1926). Atmospheric diffusion shown on a distance-neighbour graph, *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character* **110**(756): pp. 709–737.
URL: *http://www.jstor.org/stable/94463*

Richardson, L. F. (2007). *Weather Prediction by Numerical Process*, Cambridge University Press.

Seinfeld, J. H. (1983). Atmospheric diffusion theory, *in* J. Wei (ed.), *Advances in Chemical Engineering*, Vol. 12, Academic Press, New York, pp. 209–299.

Sethna, J. P. (2006). *Statistical Mechanics: Entropy, Order Parameters and Complexity*, Oxford University Press.

Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis*, Chapman & Hall.

Squires, G. L. (2001). *Practical Physics*, Cambridge University Press.

Tennekes, H. and Lumley, J. L. (1972). *A First Course in Turbulence*, The MIT press, Cambridge, Massachusetts.

Wand, M. P. and Jones, M. C. (1995). *Kernel Smoothing*, Chapman & Hall.

Wilkins, J. E. (1944). A note on skewness and kurtosis, *The Annals of Mathematical Statistics* **15**(3): 333–335.
URL: *http://dx.doi.org/10.1214/aoms/1177731243*

WMO (2014). World record wind gust, `http://www.wmo.int/pages/mediacentre/infonotes/info_58_en.html`. World Meteorological Office.

Yeh, K. C. and Liu, C. H. (1972). *Theory of Ionospheric Waves*, Vol. 17 of *International Geophysics Series*, Academic Press, New York and London.

Appendices

Appendix A

Python Scripts

All python scripts written to handle the data are appended here for completeness. Most of the scripts are straightforward, and therefore sparsely commented. I have tried to comment on what a section does, and *why* a certain choice was made in several places (as much as time allowed). Most of the functions have been placed into the file *madona_mods.py*.

I have used the *numpy* library for array handling, *matplotlib* for plotting, and *mpl_toolkits* for 3D plots. The scripts read the address of the data file in the command line, and ask for further conditions, such as FF or CM frame. I apologize in advance for any redundancy in the code which makes it less readable and longer than necessary. Especially the setup of each script is almost identical. The code with most of the functions used throughout the data processing part of this project is at the very end of the appendix.

A.1 Chapter 1

Listing A.1: A1_datarep.py

```
#Plots raw mad files in 3D for either FF or CM
     frame.
#Enter python A1_datarep.py <location of data
     file>

from madona_mods import *
from matplotlib.pyplot import *
from numpy import *
from mpl_toolkits.mplot3d import Axes3D

c = store_data()
time = len(c)
#"time" is amount of samples after missing
     scans have been removed from data
dx = find_dx()
R = find_center_mass(c,time)

frame = ask_user()
if 0 == frame:
    #fixed frame
    c_work = c
    X = linspace(0,511*dx,512)
else:
    #CM frame
    c_work = create_centered(c,time,dx, R)
    X = linspace(-255*dx,256*dx,512)
Y = linspace(0, time*3, time)

#Make 3D plot
fig = figure()
ax = fig.gca(projection='3d')
X,Y = meshgrid(X,Y)
surf = ax.plot_surface(X, Y, c_work,rstride
     =10, cstride=10, linewidth=1, color='w')
for item in ([ax.title, ax.xaxis.label, ax.
     yaxis.label, ax.zaxis.label]
    + ax.get_xticklabels() + ax.
         get_yticklabels() + ax.
         get_zticklabels()):
    item.set_fontsize(20)
ax.set_xlabel('x [m]') #meters
ax.set_ylabel('t [s]') #seconds
ax.set_zlabel('c') #raw data without scaling
show()
```

Listing A.2: A1_samples.py

```
#Plots of scaled raw data over time with c_t
     lines.

from madona_mods import *
from matplotlib.pyplot import *
from numpy import *

c = store_data()
time = len(c)
dx = find_dx()
R = find_center_mass(c,time)
c_R = create_centered(c,time,dx, R)
miyake = miyake_scale(c_R, time) #always
     scaled against present data set
offset = 255 #input position x = 0 should
     correspond to CM or center of measurement

frame = ask_user()
if 0 == frame:
    #fixed frame
    c_work = c*miyake
else:
    #CM frame
    c_work = c_R*miyake

position = ask_position()
t = linspace(0, time*3, time)

#define thresholds
ct_low = zeros(len(t))
ct_low.fill(0.075)
ct_high = zeros(len(t))
ct_high.fill(5)


#plot all samples at position
plot(t,c_work[:,offset+position],'b')
hold('on')
plot(t,ct_low,'r')
plot(t,ct_high,'g')
show()
```

Listing A.3: A1_avg.py

```
#Plot <c> vs t in CM frame or FF.

from madona_mods import *
from matplotlib.pyplot import *
from numpy import *

c = store_data()
time = len(c)
dx = find_dx()
R = find_center_mass(c,time)
c_R = create_centered(c,time,dx, R)
miyake = miyake_scale(c_R, time) #always
     scaled against present data set


frame = ask_user()
if 0 == frame:
    #fixed frame
    c_work = c*miyake
    x = linspace(0,511*dx,512)
else:
    #CM frame
    c_work = c_R*miyake
    x = linspace(-255*dx,256*dx,512)

#find mean
mean = zeros(512) #positional data are always
     512 points
for i in range(time):
    for j in range(512):
        mean[j] += c_work[i][j]
mean = mean/(1.*time)

#plot
rc('xtick', labelsize=30)
rc('ytick', labelsize=30)
rc('text', usetex=True)
plot(x, mean, linewidth=3, label='\langle c \\
     rangle ')
xlabel('x [m]', fontsize='30')
ylabel('c [ppm]', fontsize='30')
legend(prop={'size':30})
show()
```

Listing A.4: A1_brute.py

```
#Brute force calculation of \%t, <T>, and
     crossings.
#Based on simple counting for c_t = 0.075 [ppm
     ] and 5 [ppm] in CM frame.


from madona_mods import *
from matplotlib.pyplot import *
from numpy import *

c = store_data()
time = len(c)
dx = find_dx()
R = find_center_mass(c,time)
c_R = create_centered(c,time,dx, R)
miyake = miyake_scale(c_R, time)
offset = 255
frame = ask_user()
```

```
position = ask_position()
#take only 1D array at the point this time
if 0 == frame:
    c_work = c[:,offset+position]*miyake
else:
    c_work = c_R[:,offset+position]*miyake

#define thresholds
ct_low = 0.075
ct_high = 5.

#compute crossings and time above threshold
# f is 1-way crossings, pt is time above, ext
    is <T>
f_low = 0
f_high = 0
pt_low = 0
pt_high = 0
ext_low = 0
ext_high = 0

pt_low, f_low = time_cross_wo(c_work, ct_low,
    time)
pt_high, f_high = time_cross_wo(c_work,
    ct_high, time)

#find <T> = time_above / one_way_crossings [s]
#pt multiplied by time between samples 3 [s]
    to take time dimension
#try-except in case f is 0
try:
    ext_low = pt_low*3./f_low
except:
    ext_low = 0
try:
    ext_high = pt_high*3./f_high
except:
    ext_high = 0


#make pt into percentage time above
pt_low = 100*pt_low/(1.*time)
pt_high = 100*pt_high/(1.*time)

print f_low, f_high
print ext_low, ext_high
print pt_low, pt_high
```

A.2 Chapter 3

Listing A.5: A2_skvar.py

```
#Plot alpha_3 ^2 + 1 against alpha_4, alpha_3
    against alpha_4,
#and alpha_3 against sigma^2 with
    interpolations.
#Last option is to plot alpha_3 ^2 + 1 against
    alpha_4
#in ranges of distance from the center of
    measurement or CM.

from madona_mods import *
from matplotlib.pyplot import *
from numpy import *
from math import *

def ask_plot():
    n = int(raw_input("Type␣0␣for␣SS+1␣vs␣K.␣
        Type␣1␣for␣S␣vs␣K.␣Type␣2␣for␣
        skewness␣vs␣std.␣Type␣3␣for␣color␣
        code␣positions␣SK:␣"))
    if n==0 or n==1 or n==2 or n==3:
        return n
    else:
        print "Something␣else␣than␣0,␣1,␣2␣or␣
            3␣written."
        exit()

c = store_data()
time = len(c)
dx = find_dx()
R = find_center_mass(c,time)
c_R = create_centered(c,time,dx, R)
miyake = miyake_scale(c_R, time)
offset = 255

query = ask_user()
if query == 0 :
    c_work = c*miyake
    x = linspace(0,511*dx, 512)
else:
    c_work = c_R*miyake
    x = linspace(-255*dx,256*dx, 512)

mu = get_mean(c_work, time)
sigma = get_sigma(c_work, time, mu)
skew = get_skewness(c_work, time, mu, sigma)
kurt = get_kurtosis(c_work, time, mu, sigma)

#scrub NAN numbers from list
skew_list = []
kurt_list = []
sigma_list = []
mu_list = []
for i in range(512):
    if isnan(skew[i]) == False:
        skew_list.append(skew[i])
        sigma_list.append(sigma[i])
        mu_list.append(mu[i])
    if isnan(kurt[i]) == False:
        kurt_list.append(kurt[i])
mu = asarray(mu_list)
sigma = asarray(sigma_list)
skew = asarray(skew_list)
kurt = asarray(kurt_list)

#Linear interpolation
interpolant = skew**2 + 1
A = array([interpolant, ones(len(interpolant))
    ])
w = linalg.lstsq(A.T,kurt)[0] # obtaining the
    parameters by least squares
line = w[0]*interpolant + w[1]

#model with more points
model_x = linspace(0,15,1000)
new_b = w[0] + w[1]
model = w[0]*model_x**2 + new_b

#skewness-variance plot
top_hat = zeros(len(skew))
top_hat = 1+.5*(model_x**2 + model_x*((model_x
    **2+1)**(1./2)))
var = (sigma**2)/(mu**2)

sorted_skew = []
sorted_var = []

#truncation for alpha3 alpha2 fits
#it is low because the points quickly spread
    past alpha3 = 5
for i in range(len(skew)):
    if skew[i] < 5:
        sorted_skew.append(skew[i])
        sorted_var.append(var[i])

sorted_skew, sorted_var = zip(*sorted(zip(
    sorted_skew, sorted_var)))
new_skew = asarray(sorted_skew)
new_var = asarray(sorted_var)
linearfit = polyfit(new_skew,new_var,1)
quadfit = polyfit(new_skew,new_var,2)
svar_domain = linspace(0,10,1000)
var_linfit = linearfit[1] + svar_domain*
    linearfit[0]
var_quadfit = quadfit[2] + svar_domain*quadfit
    [1] + quadfit[0]*svar_domain**2

#Plotting
rc('xtick', labelsize=30)
rc('ytick', labelsize=30)
rc('text', usetex=True)
```

```
plotflag = ask_plot()
if plotflag==0:
    #Plot S^2 + 1 vs K
    plot(interpolant,kurt,'bo',linewidth=3)
    hold('on')
    plot(interpolant,interpolant,'r',linewidth
        =3)
    plot(interpolant, line, 'g', linewidth=3,
        label='\\alpha_-4_=_\%f_(\\alpha_-3_
        ^2_+1)_+_\%f' \%(w[0],w[1]))
    legend(fontsize='40')
    xlabel('\\alpha_-3_^2_+_1', fontsize='40')
    ylabel('\\alpha_-4', fontsize='40')
    show()

if plotflag==1:
    #Plot S vs K with interpolated model
    plot(skew,kurt,'bo',linewidth=3)
    hold('on')
    plot(model_x, model, 'g', linewidth=3,
        label='\\alpha_-4_=_\%f_\\alpha_-3_^2
        _+_\%f' \%(w[0],new_b))
    xlabel('\\alpha_-3_', fontsize='40')
    ylabel('\\alpha_-4_', fontsize='40')
    legend(fontsize='40')
    ylim(0,200)
    xlim(0,15)
    show()

if plotflag==2:
    #plot comparison and interpolation of
        skewness and std.
    plot(skew, var, 'bo', linewidth=3)
    hold('on')
    vlines(5, 0, 30, 'k', linewidth=3)
    plot(model_x, top_hat, 'r', linewidth=3,
        label='Top_Hat_Model')
    plot(svar_domain, var_linfit, 'y',
        linewidth=3, label='\\sigma_^2_/_\\mu
        _^2_=_\%f_\\alpha_-3_+_\%f' \%(
        linearfit[0], linearfit[1]))
    plot(svar_domain, var_quadfit, 'm',
        linewidth=3, label='\\sigma_^2_/_\\mu
        _^2_=_\%f_\\alpha_-3_^2_+_\%f_\\alpha
        _-3_+_\%f' \%(quadfit[0], quadfit[1],
         quadfit[2]))
    xlabel('\\alpha_-3_', fontsize='40')
    ylabel('\\sigma_^2_/_\mu_^2', fontsize='40
        ')
    legend(fontsize='40')
    xlim(0,10)
    ylim(0,30)
    show()

if plotflag==3:
    #plot SS+1 vs K with color coded positions
        .
    black_interpol = asarray(ndarray.tolist(
        interpolant[0:49])+ndarray.tolist(
        interpolant[462:511]))
    black_kurt = asarray(ndarray.tolist(kurt
        [0:49])+ndarray.tolist(kurt[462:511])
        )
    blue_interpol = asarray(ndarray.tolist(
        interpolant[50:99])+ndarray.tolist(
        interpolant[412:461]))
    blue_kurt = asarray(ndarray.tolist(kurt
        [50:99])+ndarray.tolist(kurt
        [412:461]))
    green_kurt = asarray(ndarray.tolist(kurt
        [100:149])+ndarray.tolist(kurt
        [362:411]))
    green_interpol = asarray(ndarray.tolist(
        interpolant[100:149])+ndarray.tolist(
        interpolant[362:411]))
    yellow_interpol = asarray(ndarray.tolist(
        interpolant[150:199])+ndarray.tolist(
        interpolant[312:361]))
    yellow_kurt = asarray(ndarray.tolist(kurt
        [150:199])+ndarray.tolist(kurt
        [312:361]))
    red_interpol = asarray(ndarray.tolist(
        interpolant[200:311]))
    red_kurt = asarray(ndarray.tolist(kurt
        [200:311]))

    #interpolation by color
    #this was done manually for each color by
        commenting out section
    current_interpol = black_interpol
    current_kurt = black_kurt
    D = array([current_interpol, ones(len(
        current_interpol))])
    d = linalg.lstsq(D.T,current_kurt)[0]
    line = d[0]*interpolant + d[1]

    plot(black_interpol, black_kurt, 'ko',
        linewidth=3)
    #plot(blue_interpol, blue_kurt, 'bo',
        linewidth=3)
    #plot(green_interpol, green_kurt, 'go',
        linewidth=3)
    #plot(yellow_interpol, yellow_kurt, 'yo',
        linewidth=3)
    #plot(red_interpol, red_kurt, 'ro',
        linewidth=3)
    hold('on')

    plot(interpolant,interpolant,'c',linewidth
        =3)
    plot(interpolant, line, 'k', linewidth=3,
        label='\\alpha_-4_=_\%f_(\\alpha_-3_
        ^2_+1)_+_\%f' \%(d[0],d[1]))
    xlabel('\\alpha_-3_^2_+_1', fontsize='40')
    ylabel('\\alpha_-4', fontsize='40')
    legend(fontsize='40')
    show()
```

Listing A.6: A2_skvar_all.py

```
#Same as A2_skvar, but for 5 data sets. No
    color coded regions.

from madona_mods import *
from math import *

from matplotlib.pyplot import *
from numpy import *

def ask_plot():
    n = int(raw_input("Type_0_for_SS+1_vs_K._
        Type_1_for_S_vs_K._Type_2_for_
        Variance_Skewness:_"))
    if n==0 or n==1 or n==2:
        return n
    else:
        print "Something_else_than_0_or_1_or_2
            _written."
        exit()


#Loop through 5 data sets specified by user
query = ask_user()
time = 0
dx = 0
skew_list = []
kurt_list = []
sigma_list = []
mu_list = []
for k in range(5):
    c, dx = store_data_prompt()
    time = len(c)
    R = find_center_mass(c,time)
    c_R = create_centered(c,time,dx, R)
    miyake = miyake_scale(c_R, time)
    offset = 256
    if query == 0 :
        c_work = c*miyake
    else:
        c_work = c_R*miyake
    mu = get_mean(c_work, time)
    sigma = get_sigma(c_work, time, mu)
    skew = get_skewness(c_work, time, mu,
        sigma)
```

```
    kurt = get_kurtosis(c_work, time, mu,
        sigma)
    #scrub NAN
    for i in range(512):
        if isnan(skew[i]) == False:
            skew_list.append(skew[i])
            sigma_list.append(sigma[i])
            mu_list.append(mu[i])
        if isnan(kurt[i]) == False:
            kurt_list.append(kurt[i])
skew = asarray(skew_list)
kurt = asarray(kurt_list)
sigma = asarray(sigma_list)
mu = asarray(mu_list)

#Linear interpolation
interpolant = skew**2 + 1
A = array([interpolant, ones(len(interpolant))
    ])
w = linalg.lstsq(A.T,kurt)[0] # obtaining the
    parameters
line = w[0]*interpolant + w[1]

#model with more points
model_x = linspace(0,25,1000)
new_b = w[0] + w[1]
model = w[0]*model_x**2 + new_b

#skewness-variance plot
top_hat = zeros(len(skew))
top_hat = 1+.5*(model_x**2 + model_x*((model_x
    **2+1)**(1./2)))
var = (sigma**2)/(mu**2)

#fits
sorted_skew = []
sorted_var = []

#truncation for fit
for i in range(len(skew)):
    if skew[i] < 5:
        sorted_skew.append(skew[i])
        sorted_var.append(var[i])

sorted_skew, sorted_var = zip(*sorted(zip(
    sorted_skew, sorted_var)))
new_skew = asarray(sorted_skew)
new_var = asarray(sorted_var)
linearfit = polyfit(new_skew,new_var,1)
quadfit = polyfit(new_skew,new_var,2)
svar_domain = linspace(0,10,1000)
var_linfit = linearfit[1] + svar_domain*
    linearfit[0]
var_quadfit = quadfit[2] + svar_domain*quadfit
    [1] + quadfit[0]*svar_domain**2

'''
Plotting
'''
rc('xtick', labelsize=30)
rc('ytick', labelsize=30)
rc('text', usetex=True)

plotflag = ask_plot()
if plotflag==0:
    #Plot S^2 + 1 vs K
    plot(interpolant,kurt,'bo',linewidth=3)
    hold('on')
    plot(interpolant,interpolant,'r',linewidth
        =3)
    plot(interpolant, line, 'g', linewidth=3,
        label='\\alpha␣_4␣=␣\%f␣(\\alpha␣_3␣
        ^2␣+1)␣+␣\%f' \%(w[0],w[1]))
    legend(fontsize='40')
    xlabel('\\alpha␣_3␣^2␣+␣1', fontsize='40')
    ylabel('\\alpha␣_4', fontsize='40')
    ylim(0,200)
    xlim(0,200)
    show()

if plotflag==1:
    #Plot S vs K with interpolated model
    plot(skew,kurt,'bo',linewidth=3)
    hold('on')
    plot(model_x, model, 'g', linewidth=3,
        label='\\alpha␣_4␣=␣\%f␣\\alpha␣_3␣^2
        ␣+␣\%f' \%(w[0],new_b))
    xlabel('\\alpha␣_3␣', fontsize='40')
    ylabel('\\alpha␣_4␣', fontsize='40')
    legend(fontsize='40')
    ylim(0,200)
    xlim(0,15)
    show()

if plotflag==2:
    #plot comparison and interpolation of
        skewness and std.
    plot(skew, var, 'bo', linewidth=3)
    hold('on')
    vlines(5, 0, 30, 'k', linewidth=3)
    plot(model_x, top_hat, 'r', linewidth=3,
        label='Top␣Hat␣Model')
    plot(svar_domain, var_linfit, 'y',
        linewidth=3, label='\\sigma␣^2␣/␣\\mu
        ␣^2␣=␣\%f␣\\alpha␣_3␣+␣\%f' \%(
        linearfit[0], linearfit[1]))
    plot(svar_domain, var_quadfit, 'm',
        linewidth=3, label='\\sigma␣^2␣/␣\\mu
        ␣^2␣=␣\%f␣\\alpha␣_3␣^2␣+␣\%f␣\\alpha
        ␣_3␣+␣\%f' \%(quadfit[0], quadfit[1],
         quadfit[2]))
    xlim(0,10)
    ylim(0,30)
    xlabel('\\alpha␣_3␣', fontsize='40')
    ylabel('\\sigma␣^2␣/␣\mu␣^2', fontsize='40
        ')
    legend(fontsize='40')
    show()
```

Listing A.7: A2_moments.py

```
#Plot <c^n>^(1/n) for n=1,2,3,4.

from math import *
from matplotlib.pyplot import *
from numpy import *
from madona_mods import *

c = store_data()
time = len(c)
dx = find_dx()
R = find_center_mass(c,time)
c_R = create_centered(c,time,dx, R)
miyake = miyake_scale(c_R, time)

query = ask_user()
if query == 0 :
    c_work = c*miyake
    x = linspace(0,511*dx, 512)
else:
    c_work = c_R*miyake
    x = linspace(-255*dx,256*dx, 512)

#Moments 1-4.
nu1 = zeros(512)
nu2 = zeros(512)
nu3 = zeros(512)
nu4 = zeros(512)
for i in range(time):
    for j in range(512):
        nu1[j] += c_work[i][j]
        nu2[j] += c_work[i][j]**2
        nu3[j] += c_work[i][j]**3
        nu4[j] += c_work[i][j]**4
nu1 = nu1/(1.*time)
nu2 = nu2/(1.*time)
nu3 = nu3/(1.*time)
nu4 = nu4/(1.*time)

#"normalize"
nu2 = nu2**(1./2)
nu3 = nu3**(1./3)
nu4 = nu4**(1./4)

#Plot
```

```
rc('xtick', labelsize=30)
rc('ytick', labelsize=30)
rc('text', usetex=True)
plot(x, nu1, label='<c(t)>')
hold('on')
plot(x,nu2, label='<c(t)^2>**(1/2)')
plot(x,nu3, label='<c(t)^3>**(1/3)')
plot(x,nu4, label='<c(t)^4>**(1/4)')
xlabel('x [m]', fontsize='30')
ylabel('c [ppm]', fontsize='30')
legend(prop={'size':30})
show()
```

A.3 Chapter 4

Listing A.8: A3_randwalk.py

```
#Simulate 1D unobstructed random walk. Compare
      with Gaussian analytical solution

import sys
from random import *
from math import *
from numpy import *
from matplotlib.pyplot import *

def gaussian(D, t, x):
    #D normed to 1.
    gauss = 1./(2*sqrt(D*pi*t))*exp(-(x**2)
        /(4.*D*t))
    return gauss

#Define parameters
c_0 = 20000 #walkers at x=0 at all times
iterations = 499 #actually iterations -1, i=0
     in the loop is not counted
loops_analytical = 1000
dt = 1
time = iterations*dt
D = 1./(24*dt)#diffusion constant for uniform
     distribution
l_0 = sqrt(2.*D*dt)
positions = 400
d = positions*l_0 #run at least d/l_0
     iterations
walkers = []
for i in range(c_0):
    walkers.append(0)
no_walkers = 0


#Rules: roll < 0 is left, roll <= 0 is right.
     If a walker is at x=-d+1, rolling
#< 0 detaches him from list.
#If a walker is at x=d-1, rolling >0 detaches
     him from list.
#The iterations were not run long enough for
     this to heavily
#affect the distribution.

for i in range(iterations):
    print i
    cap = len(walkers)
    walked = 0
    while walked < cap:
        roll = uniform(-.5,.5)
        if roll >= 0:#slight nonequality here
            roll = 1
        else:
            roll = 0

        if (d-l_0)==walkers[walked]: #if next
             to right sink
            if 1==roll:
                walkers.remove(d-l_0)
                cap += -1
            else:
                walkers[walked] += -l_0
        elif -(d-l_0)==walkers[walked]: #if
             next to left sink
            if 0==roll:
                walkers.remove(-d+l_0)
                cap += -1
            else:
                walkers[walked] += l_0
        else: #else a walker in homogeneous
             space
            if 1==roll:
                walkers[walked] += l_0
            else:
                walkers[walked] += -l_0
        walked += 1
x_mc = linspace(-d,d,positions)
y_mc = zeros(positions)
for j in range(len(x_mc)):
    for i in range(len(walkers)):
        if walkers[i]==x_mc[j]:
            y_mc[j] += 1./c_0

x_an = linspace(-d,d,positions+1)
y_an = zeros(positions+1)
for j in range(len(x_an)):
    y_an[j] = gaussian(D,time,x_an[j])

#plots
rc('xtick', labelsize=30)
rc('ytick', labelsize=30)

hist(walkers, bins=x_mc, normed=1)
hold('on')
plot(x_an, y_an, linewidth=3, label='Classical
     Diffusion')
vlines(0,0,1)
xlim(-20,20)
ylim(0,0.1)
legend()
show()
```

Listing A.9: A3_plumewidth.py

```
#FF plume width as 2 standard deviations

from madona_mods import *
from numpy import *
from matplotlib.pyplot import *
from math import *

c = store_data()
time = len(c)
dx = find_dx()
R = find_center_mass(c,time)
c_R = create_centered(c,time,dx, R)

miyake = miyake_scale(c_R, time)

c = c*miyake
c_R = c_R*miyake

#find <c(t)^m> at x = sum (c(t)^m at x)/
     totaltime
#center of mass frame
mean_1_cm = zeros(512)
for i in range(time):
    for j in range(512):
        mean_1_cm[j] = mean_1_cm[j] + c[i][j]
mean_1_cm = mean_1_cm/float(time)

#find mean position: sum (x*c(x)*dx)/sum(c(x))
     . dx cancels.
mean_pos = 0
sum_c = 0
x = linspace(0,511*dx,512)
for i in range(len(x)):
    mean_pos += mean_1_cm[i]*x[i]
    sum_c += mean_1_cm[i]
mean_pos = mean_pos /float(sum_c)

#find sigma
sigma = 0
for i in range(len(x)):
```

```
    sigma += (x[i]**2) * mean_1_cm[i]
sigma = (sigma/float(sum_c)) - (mean_pos**2)
sigma = sqrt(sigma)

#Plot
rc('xtick', labelsize=30)
rc('ytick', labelsize=30)
plot(x, mean_1_cm)
hold('on')
vlines(mean_pos, 0, max(mean_1_cm), 'r',
    linestyles='dashed', linewidth='3', label
    ='mean')
vlines(mean_pos + sigma, 0, max(mean_1_cm), 'g
    ', linestyles='dotted', linewidth='3',
    label = 'mean + sigma')
vlines(mean_pos - sigma, 0, max(mean_1_cm), 'g
    ', linestyles='dotted', linewidth='3',
    label= 'mean - sigma')
xlabel('x - R [m]', fontsize='30')
ylabel('c [ppm]', fontsize='30')
title('Standard deviation = \%f [m].' \% (
    sigma) , fontsize='30')
legend(fontsize='30')
show()
```

Listing A.10: A3_meandering.py

```
#Make an array of centers of mass, and create
    a PDF histogram based on the centers.

from madona_mods import *
from numpy import *

c = store_data()
time = find_total_time()
dx = find_dx()
R = find_center_mass(c,time)
R = R*dx
lower = 0
upper = max(R)
antall = 41
domain = linspace(lower,upper,num=antall)
pdf = histogram(R, bins=linspace(lower,upper,
    num=antall), normed=1)
R_pdf = zeros(len(pdf[0]))
p_c = zeros(len(pdf[0]) )#stores prob density
    points
for i in range(len(R_pdf)):
    R_pdf[i] = (pdf[1][i+1] + pdf[1][i])/2.
    p_c[i] = pdf[0][i]

#Do Gaussian smoothing
gausslet_points = (antall-1)*100 - 1
distance = R_pdf[1] - R_pdf[0] #since equally
    spaced, minimum 2 points for pdf
small_gauss = zeros((len(R_pdf),
    gausslet_points)) #takes all smoothing
    data
#loop over all points and create arrays
for i in range(len(p_c)):
    local_pd = p_c[i]
    small_gauss[i] = gausslet(upper, R_pdf[i],
        distance, local_pd, gausslet_points)
meshed_gauss = zeros(gausslet_points)
#mesh together
for j in range(gausslet_points): #all mesh
    points
    for i in range(len(p_c)): #no. poles to
        mesh "around"
        meshed_gauss[j] += small_gauss[i][j]
mesh_domain = linspace(-upper,2*upper,
    gausslet_points)

#Find the mean position based on pdf
R_mean = 0
delta_x = mesh_domain[1] - mesh_domain[0]
for i in range(len(mesh_domain)):
    R_mean += mesh_domain[i]*meshed_gauss[i]
R_mean = R_mean*delta_x

#Find sigma^2 = <x^2> - <x>^2
sigma = 0
for i in range(len(mesh_domain)):
    sigma += (mesh_domain[i]**2) *
        meshed_gauss[i]
sigma = (sigma * delta_x) - R_mean**2
sigma = sqrt(sigma)

#Plot
rc('xtick', labelsize=30)
rc('ytick', labelsize=30)
hist(R, bins=domain,color='w', normed=1)
hold('on')
plot(mesh_domain, meshed_gauss, 'b', linewidth
    =3, label='Gaussian smoothed PDF')
vlines(R_mean, 0, max(meshed_gauss), 'r',
    linestyles='dashed', linewidth='3', label
    ='mean')
vlines(R_mean - sigma, 0, max(meshed_gauss), '
    g', linestyles='dotted', linewidth='3',
    label='mean - sigma')
vlines(R_mean + sigma, 0, max(meshed_gauss), '
    g', linestyles='dotted', linewidth='3',
    label='mean + sigma')
xlabel('R [m]', fontsize='30')
ylabel('p(R=x) ', fontsize='30')
title('Standard deviation = \%f [m].' \% (
    sigma) , fontsize='30')
legend(fontsize='30')
show()
```

A.4 Chapter 5

Listing A.11: A4_interpolation.py

```
#Use linear interpolation to calculate \% t.
#Also checks error of Brute Force in \% t as
    opposed to interpolated.


from madona_mods import *
from math import *
from matplotlib.pyplot import *
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.ticker import LinearLocator,
    FormatStrFormatter
from numpy import *

c = store_data()
time = len(c)
dx = find_dx()
R = find_center_mass(c,time)
c_R = create_centered(c,time,dx, R)
miyake = miyake_scale(c_R, time)
offset = 255
query = ask_user()
if query == 0:
    c_work = c*miyake
else:
    c_work = c_R*miyake

samples = 500 #number of concentration
    thresholds
positions = len(c_work[0])
upper_threshold = 15.
lower_threshold = 0.

#Find time spent above threshold and count
    frequency of crossing
threshold = linspace(lower_threshold,
    upper_threshold, samples) # x-axis
time_spent = zeros(samples)
time_spent_wo = zeros(samples)
crossings = zeros(samples)
avg_time = zeros(samples)
position = ask_position() + offset

for k in range(samples):
    time_spent[k], crossings[k] = time_cross(
        c_work[:,position], threshold[k],
        time)
```

```
    time_spent_wo[k], crossings[k] =
        time_cross_wo(c_work[:,position],
        threshold[k], time)
    if crossings[k] != 0:
        avg_time[k] = time_spent[k]/crossings[
            k]
    else:
        avg_time[k] = 0

avg_time = avg_time * 3 #3s intervals between
    measurements
time_spent = 100*time_spent/float(time) #to
    get percentages
time_spent_wo = 100*time_spent_wo/float(time)
    #to get percentages

#Plot
rc('xtick', labelsize=30)
rc('ytick', labelsize=30)
rc('text', usetex=True)

# Section for plotting Interpolation vs Brute
    Force commented out below
#plot(threshold, time_spent, 'b', linewidth=3,
     label='Using Linear Interpolation')
#hold('on')
#plot(threshold, time_spent_wo, 'r', linewidth
    =3, label='Using Brute Force')
#xlabel('c_t [ppm]', fontsize='40')
#ylabel('t [\%]', fontsize='40')

#Plot error in Brute Force compared to
    interpolation.
plot(threshold, time_spent - time_spent_wo,
    linewidth=3, label='Linear Interpolation
    - Brute Force')
xlabel('c_t [ppm]', fontsize='40')
ylabel('\Delta t [\%]', fontsize='40')
legend(fontsize='40')
show()
```

Listing A.12: A4_thresholds.py

```
#Can create crossings, \%t and <T> against
#continuous c_t with fittings for <T>.
#currently set to plot <T> with fittings

from madona_mods import *
from math import *
from matplotlib.pyplot import *
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.ticker import LinearLocator,
    FormatStrFormatter
from numpy import *

def make_log(old_array):
    #takes 1 dim array and returns array with
        natural log of its elements
    new_array = zeros(len(old_array))
    for i in range(len(new_array)):
        if old_array[i] != 0:
            new_array[i] = log(old_array[i])
        else:
            new_array[i] = 0
    return new_array

c = store_data()
time = len(c)
dx = find_dx()
R = find_center_mass(c,time)
c_R = create_centered(c,time,dx, R)
miyake = miyake_scale(c_R, time)
offset = 256
query = ask_user()
if query == 0 :#scaled FF frame
    c_work = c*miyake
else: #scaled CM frame
    c_work = c_R*miyake

#Defines c_t
samples = 500 #number of sample concentration
    thresholds.
positions = len(c_work[0])
upper_threshold = 15.
lower_threshold = 0.
upper_model = exp(1.5)
lower_model = 1./exp(1)

#Find time spent above c_t and count one way
    crossings
threshold = linspace(lower_threshold,
    upper_threshold, samples)
threshold_model = linspace(lower_model,
    upper_model, samples)
time_spent = zeros(samples)
time_model = zeros(samples)
crossings = zeros(samples)
crossings_model = zeros(samples)
avg_time = zeros(samples)
avg_model = zeros(samples)
position = ask_position() + offset

for k in range(samples):
    time_spent[k], crossings[k] = time_cross(
        c_work[:,position], threshold[k],
        time)
    time_model[k], crossings_model[k] =
        time_cross(c_work[:,position],
        threshold_model[k], time)
    if crossings[k] != 0:
        avg_time[k] = time_spent[k]/crossings[
            k]
    else:
        avg_time[k] = 0
    if crossings_model[k] != 0:
        avg_model[k] = time_model[k]/
            crossings_model[k]
    else:
        avg_model[k] = 0

avg_time = avg_time * 3 #3s intervals between
    measurements
avg_model = avg_model * 3
time_spent = 100*time_spent/float(time) #to
    get percentages
time_model = 100*time_model/float(time)

#do loglog
new_thresh_model = make_log(threshold_model)
new_avg_model = make_log(avg_model)
#make a least-squares fit
A = array([new_thresh_model, ones(len(
    new_thresh_model))])
w = linalg.lstsq(A.T,new_avg_model)[0] #
    obtaining the parameters
line = w[0]*new_thresh_model + w[1]
model = exp(w[1])*threshold_model**(w[0])
pos = position - 256
long_model = exp(w[1])*threshold**(w[0])

#Plots. Currently plots only <T>
rc('xtick', labelsize=30)
rc('ytick', labelsize=30)
rc('text', usetex=True)
#Commented sections below fit <T>
#loglog(threshold, avg_time, 'bo', linewidth
    =3, label='loglog data')
#hold('on')
#plot(threshold_model, model, 'r', linewidth
    =3, label='linear model')
#legend(fontsize='30')
plot(threshold, avg_time, 'b', linewidth=3)
hold('on')
plot(threshold, long_model, 'r', linewidth=3,
    label='\langle T \\rangle \propto c_{t
    }^{%f}' %w[0])
xlabel('c_t [ppm]', fontsize='40')
ylabel('\langle T \\rangle [s]', fontsize='40'
    )
xlim(0,15)
ylim(0,300)
legend(fontsize='40')
show()
```

Listing A.13: A4_stack.py

```
#Purpose is to create stackplots of <T>, %T, #
    crossings vs c_t for a selection of x.

from madona_mods import *
from math import *
from matplotlib.pyplot import *
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.ticker import LinearLocator,
    FormatStrFormatter
from numpy import *

#setup
c = store_data()
time = len(c)
dx = find_dx()
R = find_center_mass(c,time)
c_R = create_centered(c,time,dx, R)
miyake = miyake_scale(c_R, time)
offset = 0
query = ask_user()
if query == 0 :#scaled FF frame
    c_work = c*miyake
    offset = 256
else: #scaled CM frame
    c_work = c_R*miyake
    offset = 256

#settings
samples = 250 #number of sample times.
positions = len(c_work[0])
upper_threshold = 15.
lower_threshold = 0.

# time above threshold and crossings
threshold = linspace(lower_threshold,
    upper_threshold, samples)
time_spent = zeros((positions,samples))
crossings = zeros((positions,samples))
avg_time = zeros((positions, samples))
for j in range(positions):
    for k in range(samples):
        time_spent[j][k], crossings[j][k] =
            time_cross(c_work[:,j], threshold
            [k],time)
        if crossings[j][k] != 0:
            avg_time[j][k] = time_spent[j][k]/
                crossings[j][k]
        else:
            avg_time[j][k] = 0

#<T>
avg_time = avg_time * 3 #3s intervals between
    measurements
time_spent = 100*time_spent/float(time) #to
    get percentages

#Create \%t, crossings, and <T> for stacked
    positions.
#Right now set left of center of measurement.
#Positions must be set manually.
plot_percentage = row_stack((time_spent[0+
    offset], time_spent[-20+offset],
    time_spent[-40+offset], time_spent[-60+
    offset], time_spent[-80+offset]))
#plot_crossings = row_stack((crossings[0+
    offset], crossings[20+offset], crossings
    [40+offset], crossings[60+offset],
    crossings[80+offset]))
#plot_avgtime = row_stack((avg_time[0+offset],
     avg_time[20+offset], avg_time[40+offset
    ],avg_time[60+offset],avg_time[80+offset
    ]))

#Plot
#Set to plot \%t. Commented lines below plot
    crossings and <T>.
rc('xtick', labelsize=30)
rc('ytick', labelsize=30)
fig, ax = subplots()

ax.stackplot(threshold,plot_percentage)
#ax.stackplot(threshold, plot_crossings)
#ax.stackplot(threshold, plot_avgtime)

xlabel('threshold [ppm]', fontsize='30')
ylabel('<T> [s]', fontsize='30')
xlim(0,15)
ylim(0,400)
show()
```

Listing A.14: A4_positions.py

```
#Plot \%t, crossings and <T> over each
    position in the original data

from madona_mods import *
from math import *
from matplotlib.pyplot import *
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.ticker import LinearLocator,
    FormatStrFormatter
from numpy import *

#list of thresholds to plot.
#[threshold value, flag]
#flag is 0 when c_t has not been crossed, and
    1 when crossed.
threshold_list = [[0.25,0],[0.75, 0], [5,0]]
size = len(threshold_list)

c = store_data()
time = len(c)
dx = find_dx()
R = find_center_mass(c,time)
c_R = create_centered(c,time,dx, R)
miyake = miyake_scale(c_R, time)
query = ask_user()
if query == 0 :#scaled FF frame
    c_work = c*miyake
    positions = linspace(0,511*dx,512)
else: #scaled CM frame
    c_work = c_R*miyake
    positions = linspace(-255*dx,256*dx,512)

#stores crossings and \%t
freqs = zeros((size, 512))
timespent = zeros((size,512))
#compute \%t and crossings using linear
    interpolation
for j in range(512):
    for k in range(size):
        timespent[k][j], freqs[k][j] =
            time_cross(c_work[:,j],
            threshold_list[k][0], time)

#<T> = 3*timespent/crossings [s]
exp_t = zeros((size,512))
for j in range(512):
    for k in range(size):
        if freqs[k][j] != 0:
            exp_t[k][j] = 3.*timespent[k][j]/
                freqs[k][j]
        else:
            exp_t[k][j] = 0

#plot: currently set to plot <T>. Set freqs or
     timespent to plot
#crossings or \%t
rc('xtick', labelsize=30)
rc('ytick', labelsize=30)
for i in range(size):
    plot(positions, exp_t[i], label='C_t = 
        \%.3f [ppm]' \%threshold_list[i][0])
xlabel('position from R [m]', fontsize = '30')
ylabel('<T> [s]', fontsize = '30')
legend(prop={'size':30})
show()
```

Listing A.15: A4_comparegaussian.py

```
#Compare \%t found by linear interpolation
    with CDF of Gaussian fit.
```

```
from madona_mods import *
from math import *
from matplotlib.pyplot import *
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.ticker import LinearLocator,
    FormatStrFormatter
from numpy import *

def make_log(old_array):
    #takes 1 dim array and returns array with
        natural log of its elements
    new_array = zeros(len(old_array))
    for i in range(len(new_array)):
        if old_array[i] != 0:
            new_array[i] = log(old_array[i])
        else:
            new_array[i] = 0
    return new_array

def get_mean(c, time):
    mean = 0
    for i in range(time):
        mean = mean + c[i]
    mean = mean/time
    return mean

def get_std(c, c_avg, time):
    std = 0
    for i in range(time):
        std = std + double((c[i] - c_avg))**2
    std = std/time
    std = sqrt(std)
    return std

c = store_data()
time = len(c)
dx = find_dx()
R = find_center_mass(c,time)
c_R = create_centered(c,time,dx, R)
miyake = miyake_scale(c_R, time)
offset = 256
query = ask_user()
if query == 0 :#scaled FF frame
    c_work = c*miyake
else: #scaled CM frame
    c_work = c_R*miyake

#Defines c_t
samples = 500 #number of sample concentrations
    .
positions = len(c_work[0])
upper_threshold = 15.
lower_threshold = 0.

#Find time spent above c_t and count one way
    crossings
threshold = linspace(lower_threshold,
    upper_threshold, samples)
time_spent = zeros(samples)
crossings = zeros(samples)
position = ask_position() + offset
for k in range(samples):
    time_spent[k], crossings[k] = time_cross(
        c_work[:,position], threshold[k],
        time)
time_spent = time_spent/float(time) #to get in
     fractions of 1

#Create CDF of gaussian fit
domain = linspace(0,20, 512)
mean = get_mean(c_work[:,position], time)
std = get_std(c_work[:,position], mean, time)
cdf = zeros(len(domain))
for i in range(512):
    cdf[i] = erfc((domain[i] - mean)/(std*sqrt
        (2)))
cdf = cdf/(2.)

#Plots. Currently plots only <T>
rc('xtick', labelsize=30)
rc('ytick', labelsize=30)
rc('text', usetex=True)
plot(threshold, time_spent, 'b', linewidth=3,
    label='By linear interpolation')
hold('on')
plot(domain, cdf, 'r', linewidth=3, label='
    Gaussian CDF fit')
xlabel('c_t [ppm]', fontsize='40')
ylabel('\langle \%t \\rangle [s]', fontsize='
    40')
legend(fontsize='40')
show()
```

Listing A.16: A4_crossings3D.py

```
#3D crossings vs thresholds and positions.
from madona_mods import *
from math import *
from matplotlib.pyplot import *
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.ticker import LinearLocator,
    FormatStrFormatter
from numpy import *

#list of thresholds. First of sublist is
    threshold value, second is flag.
#flag is 0 for not crossed, and 1 when crossed
    .

size = 25
threshold_list = zeros((size,2))
for i in range(size):
    threshold_list[i][0] = (i*5.)/(size-1.)
    threshold_list[i][1] = 0

c = store_data()
time = len(c)
dx = find_dx()
R = find_center_mass(c,time)
c_R = create_centered(c,time,dx, R)
miyake = miyake_scale(c_R, time)
offset = 256
query = ask_user()
if query == 0 :#scaled FF frame
    c_work = c*miyake
    positions = linspace(0,511*dx,512)
else: #scaled CM frame
    c_work = c_R*miyake
    positions = linspace(-255*dx,256*dx,512)

#Count crossings. Crossings do not need
    interpolation to count.
freqs = zeros((size,512))
for j in range(512):
    for i in range(time):
        for k in range(size):
            if threshold_list[k][1] == 0:
                if c_work[i][j] >
                    threshold_list[k][0]:
                    freqs[k][j] += 1
                    threshold_list[k][1] = 1
            else:
                if c_work[i][j] <=
                    threshold_list[k][0]:
                    threshold_list[k][1] = 0

#3D plot
c_t = threshold_list[:,0]
fig = figure()
ax = fig.gca(projection='3d')
X,Y = meshgrid(positions,c_t)
#Wire frame plot
surf = ax.plot_wireframe(X,Y,freqs,rstride=1,
    cstride=50, linewidth=3, color='b')
#Commented out below: color coded plot
#surf = ax.plot_surface(X, Y, freqs,rstride
    =10, cstride=10, linewidth=1, color='w')
#surf = ax.plot_surface(X, Y, freqs,rstride=1,
     cstride=25, cmap=cm.jet,
#  linewidth=0, antialiased=False)
#fig.colorbar(surf, shrink=1, aspect=1)
for item in ([ax.title, ax.xaxis.label, ax.
    yaxis.label, ax.zaxis.label]
```

```
    + ax.get_xticklabels() + ax.
        get_yticklabels() + ax.
        get_zticklabels()):
    item.set_fontsize(20)
ax.set_xlabel('x [m]')
ax.set_ylabel('c_t [ppm]')
ax.set_zlabel('# crossings')
show()
```

A.5 Chapter 6

Listing A.17: A5_histogram.py

```
#p(c) of a position in CM frame
#also plots the gaussian fit
#and analyticall computed gaussian + delta
    spike fit.

from madona_mods import *
from math import pi as PI
from math import erf
from matplotlib.pyplot import *
from numpy import *

def get_mean(c, time):
    #1 dim array version of the one in
        madona_mods
    mean = 0
    for i in range(time):
        mean = mean + c[i]
    mean = mean/time
    return mean

def get_std(c, c_avg, time):
    #1 dim array version of the one in
        madona_mods
    std = 0
    for i in range(time):
        std = std + double((c[i] - c_avg))**2
    std = std/time
    std = sqrt(std)
    return std

c = store_data()
time = len(c)
dx = find_dx()
R = find_center_mass(c,time)
c_R = create_centered(c,time,dx, R)
miyake = miyake_scale(c_R, time)
position = int(sys.argv[2]) +256
c_work = c_R[:,position]*miyake #specific
    position from command line

#Truncations and number of histogram bars.
lower = 0.
upper = 0.
antall =41 #number of bars + 1

#find upper truncation of c_t as largest c
for i in range(time):
    if c_work[i] > upper:
        upper = c_work[i]
#if we want to change upper truncation
    manually
#upper = 10.

#find p( c ) by histogram and center of each
    histogram
#I used this to try out some log and loglog
    fits that did not work out
domain = linspace(lower,upper,num=antall)
pdf = histogram(c_work, bins=linspace(lower,
    upper,num=antall), normed=1)
c_f = zeros(len(pdf[0]))
for i in range(len(c_f)):
        c_f[i] = (pdf[1][i+1]+pdf[1][i])/2.

#Find average and std of concentrations for
    Gaussian fit
size = 512
gauss_domain = linspace(lower, upper, size)
gauss_domainlong = linspace(-upper,upper,2*
    size)
c_mean = get_mean(c_work, time)
std = get_std(c_work, c_mean, time)
gauss = gaussian(c_mean, std, gauss_domainlong
    )
alt_mean = 4
alt_std = 4.25
gauss_alternate = gaussian(alt_mean, alt_std,
    gauss_domain)

#Delta Bin for recalculated Gaussian fit
A = .5*(1-erf(alt_mean/(sqrt(2)*alt_std)))

#Plotting
rc('xtick', labelsize=30)
rc('ytick', labelsize=30)
hist(c_work, bins=domain, color='b', normed=1)
hold('on')
plot(gauss_domainlong, gauss, 'r',linewidth=3,
    label='Gaussian fit')
vlines(c_mean, 0, max(gauss), 'r', linewidth='
    3', linestyles='dashed', label = 'mean')
vlines(c_mean + std, 0, max(gauss), 'r',
    linestyles='dotted', linewidth='3', label
    ='1 standard deviation from mean')
vlines(alt_mean, 0, max(gauss_alternate), 'g',
    linewidth='3', linestyles='dashed',
    label = 'mean for reworked fit')
vlines(alt_mean + alt_std, 0, max(gauss), 'g',
    linestyles='dotted', linewidth='3',
    label='1 reworked st.d. from reworked
    mean')
vlines(0, 0, A, 'g', linewidth='6')
plot(gauss_domain, gauss_alternate, 'g',
    linewidth=3, label='Reworked version')
legend()
xlabel('c [ppm]', fontsize='30')
ylabel('p(c)', fontsize='30')
legend(fontsize='30')
xlim(-5,25)
ylim(0,0.2)
show()
```

Listing A.18: A5_gaussian_smoothing.py

```
#Gaussian smoothing of the p(c) of position in
    CM given in command line.

from madona_mods import *
from math import pi as PI
from math import erf, exp
from matplotlib.pyplot import *
from numpy import *

def get_mean(c, time):
    mean = 0
    for i in range(time):
        mean = mean + c[i]
    mean = mean/time
    return mean

def get_std(c, c_avg, time):
    std = 0
    for i in range(time):
        std = std + double((c[i] - c_avg))**2
    std = std/time
    std = sqrt(std)
    return std

c = store_data()
time = len(c)
dx = find_dx()
R = find_center_mass(c,time)
c_R = create_centered(c,time,dx, R)
miyake = miyake_scale(c_R, time)
position = int(sys.argv[2]) +255
c_work = c_R[:,position]*miyake

#bins
```

```
lower = 0.
upper = 0.
antall =41

#find upper truncation
for i in range(time):
    if c_work[i] > upper:
        upper = c_work[i]
#or change it manually
#upper = 10.

domain = linspace(lower,upper,num=antall)
pdf = histogram(c_work, bins=linspace(lower,
    upper,num=antall), normed=1)
c_f = zeros(len(pdf[0]) ) #stores
    concentration points
p_c = zeros(len(pdf[0]) )#stores prob density
    points
for i in range(len(c_f)):
    c_f[i] = (pdf[1][i+1]+pdf[1][i])/2.
    p_c[i] = pdf[0][i]

#gausslets are the Gaussians of each bin
gausslet_points = (antall-1)*100 - 1
distance = c_f[1] - c_f[0] #since bin domain
    has constant spacing
small_gauss = zeros((len(c_f),gausslet_points)
    ) #the gausslets

#loop over all points and create gausslets
for i in range(len(p_c)):
    local_pd = p_c[i]
    small_gauss[i] = gausslet(upper, c_f[i],
        distance, local_pd, gausslet_points)
meshed_gauss = zeros(gausslet_points)
#mesh together in a fixed domain
for j in range(gausslet_points): #all mesh
    points
    for i in range(len(p_c)): #no. poles
        meshed_gauss[j] += small_gauss[i][j]
mesh_domain = linspace(-upper,2*upper,
    gausslet_points)

#Find average and std of concentrations
size = 512
gauss_domain = linspace(lower, upper, size)
c_mean = get_mean(c_work, time)
std = get_std(c_work, c_mean, time)
gauss = gaussian(c_mean, std, gauss_domain)

#Plot
rc('xtick', labelsize=30)
rc('ytick', labelsize=30)
hist(c_work, bins=domain, color='w', normed=1)
hold('on')
#plot(c_f, p_c, 'gx')
plot(mesh_domain, meshed_gauss, 'b', linewidth
    =3)
#plot(gauss_domain, gauss, 'g', label='
    Gaussian fit for same mean and standard
    deviation')
#vlines(c_mean, 0, max(gauss), 'r', label = '
    mean')
#vlines(c_mean + std, 0, max(gauss), 'y',
    label='1 standard deviation from mean')
#legend()
xlabel('c␣[ppm]', fontsize='30')
ylabel('p(c)', fontsize='30')
#title('PDF at x=\%f' \%(position-256))
#legend(fontsize='30')
show()
```

Listing A.19: A5_histogram3D.py

```
#Plot 3D histogram in either FF or CM frame.

from mpl_toolkits.mplot3d import Axes3D
from madona_mods import *
from math import pi as PI
from math import erf
from matplotlib.pyplot import *
from numpy import *

c = store_data()
time = len(c)
dx = find_dx()
R = find_center_mass(c,time)
c_R = create_centered(c,time,dx, R) #center of
    mass frame concentrations.
miyake = miyake_scale(c_R, time) #scaling
    always done against 21 points around x=R.
offset = 256

x_samples = 10 #samples along x to minimize
    computational expense
query = ask_user()
if query == 0 :#FF
    c_work = c*miyake
    domain_x = linspace((-80+255)*dx, (80+255)
        *dx, x_samples)
    large_x = linspace(0,511*dx,512)
else: #CM
    c_work = c_R*miyake
    domain_x = linspace(-80*dx,80*dx,x_samples
        )
    large_x = linspace(-255*dx, 256*dx, 512)

#truncations and bins
lower = 0.
upper = 10.
antall =41
x_separation = 161./x_samples
domain_c = linspace(lower,upper,num=antall)
distance = domain_x[1] - domain_x[0]
c_dist = domain_c[1] - domain_c[0]
hist = zeros((x_samples,antall-1))
for i in range(x_samples):
    hist[i] = histogram(c_work[:,int(256-80 +
        i*x_separation)], bins=domain_c,
        normed=1)[0]

#cap at p=0.5. This is for easier
    visualization
for i in range(x_samples):
    for j in range(antall-1):
        if hist[i][j] > 0.5:
            hist[i][j] = 0.5

#compute histograms
temp_pdf = histogram(c_work[:0], bins=domain_c
    , normed=1)
c_plot = zeros(len(temp_pdf[0]))
for i in range(len(c_plot)):
    c_plot[i] = (temp_pdf[1][i+1]+temp_pdf[1][
        i])/2.
pdf = zeros((512, antall-1))
for j in range(512):
    pdf[j] = histogram(c_work[:,j],bins=
        domain_c, normed=1)[0]


#cap at 0.5
cap = 0.5
for j in range(512):
    for i in range(antall-1):
        if pdf[j][i] > 0.5:
            pdf[j][i] = 0.5

#Plot 3D bins
fig = figure()
ax = fig.add_subplot(111, projection='3d')
elements = (len(domain_c) - 1) * (len(domain_x
    ) - 1)
xpos, ypos = meshgrid(domain_c[:-1]+0.25,
    domain_x[:-1]+0.25)
xpos = xpos.flatten()
ypos = ypos.flatten()
zpos = np.zeros(elements)
dx = c_dist * ones_like(zpos)
dy = distance/2. * ones_like(zpos)
dz = hist.flatten()
ax.bar3d(xpos, ypos, zpos, dx, dy, dz, color='
    b', zsort='average', )
ax.set_zlim(0,.5)
ax.set_xlabel('c␣[ppm]',fontsize='30')
```

```
ax.set_ylabel('x-R␣[m]',fontsize='30')
ax.set_zlabel('PDF',fontsize='30')
for item in ([ax.title, ax.xaxis.label, ax.
    yaxis.label, ax.zaxis.label]
        + ax.get_xticklabels() + ax.
            get_yticklabels() + ax.
            get_zticklabels()):
        item.set_fontsize(30)
show()
```

Listing A.20: A5_histogram3Dgaussian.py

```
#3D plot of Gaussian envelope p(c) for several
    positions x.
from mpl_toolkits.mplot3d import Axes3D
from madona_mods import *
from matplotlib.pyplot import *
from numpy import *
from math import *
from math import pi as PI
from math import erf, exp

c = store_data()
time = len(c)
dx = find_dx()
R = find_center_mass(c,time)
c_R = create_centered(c,time,dx, R)
miyake = miyake_scale(c_R, time)
offset = 256
x_samples = 10
query = ask_user()
if query == 0 :#scaled FF frame
    c_work = c*miyake
    domain_x = linspace((-80+255)*dx, (80+255)
        *dx, x_samples)
else: #scaled CM frame
    c_work = c_R*miyake
    domain_x = linspace(-80*dx,80*dx,x_samples
        )

#setup
lower = 0.
upper = 10.
antall =41
x_separation = 161./x_samples
gausslet_points = (antall-1)*100 - 1
domain_c = linspace(-upper,2*upper,
    gausslet_points)
x_dist = domain_x[1] - domain_x[0]
c_dist = domain_c[1] - domain_c[0]
pdf_domain = linspace(lower,upper,num=antall)

#create space and p(c,x) for Gaussian envelope
pdf_env = zeros((x_samples,gausslet_points))
hist = zeros((x_samples,antall-1))
get_cf = histogram(c_work[:,int(256-80)], bins
    =linspace(lower,upper,num=antall), normed
    =1)[1]
for i in range(x_samples):
    hist[i] = histogram(c_work[:,int(256-80 +
        i*x_separation)], bins=linspace(lower
        ,upper,num=antall),normed=1)[0]
c_f = zeros(len(hist[0])) #stores
    concentration points
for i in range(len(c_f)):
    c_f[i] = (get_cf[i+1]+get_cf[i])/2.

#perform Gaussian smoothing
distance = c_f[1] - c_f[0]
gausslet_points = (antall-1)*100 - 1
for k in range(x_samples):
    small_gauss = zeros((len(c_f),
        gausslet_points))
    for i in range(len(hist[k])):
        local_pd = hist[k][i]
        small_gauss[i] = gausslet(upper, c_f[i
            ], 2*distance, local_pd,
            gausslet_points)
    #mesh together
    for j in range(gausslet_points):
        for i in range(len(hist[k])): #no.
            poles
            pdf_env[k][j] += small_gauss[i][j]

#cutoff below p= 0.5 and for c<0 and c>0.
cut_low = 0
cut_high = 0
for i in range(len(domain_c)):
    if domain_c[i] > 0 and cut_low == 0:
        #smallest point over 0
        cut_low = i
    if domain_c[i] > 10 and cut_high == 0:
        #smallest point over 10
        cut_high = i
cut_diff = cut_high - cut_low
plot_pdf = zeros((x_samples, cut_diff))
plot_c = zeros(cut_diff)
for j in range(cut_diff):
    plot_c[j] = domain_c[j+cut_low]
    for i in range(len(domain_x)):
        if pdf_env[i][j+cut_low] > 0.5:
            plot_pdf[i][j] = 0.5
        else:
            plot_pdf[i][j] = pdf_env[i][j+
                cut_low]

#3D plot
fig = figure()
ax = fig.gca(projection='3d')
X,Y = meshgrid(plot_c,domain_x)
surf = ax.plot_wireframe(X,Y,plot_pdf,rstride
    =1, cstride=10, linewidth=1, color='b')
for item in ([ax.title, ax.xaxis.label, ax.
    yaxis.label, ax.zaxis.label]
    + ax.get_xticklabels() + ax.
        get_yticklabels() + ax.
        get_zticklabels()):
    item.set_fontsize(20)
ax.set_xlabel('c␣[ppm]')
ax.set_ylabel('x-R␣[m]')
ax.set_zlabel('PDF')
ax.set_zlim(0,.5)
ax.set_xlim(0,10)
show()
```

Listing A.21: A5_jpdf.py

```
#Create JPDF p(c,dc/dt) by linear
    interpolation for dc/dt.
#Presently set to use the average of the two
    methods to find dc/dt.
#Can be set to "fold" the JPDF by sampling
    only abs(dcdt).

from mpl_toolkits.mplot3d import Axes3D
from madona_mods import *
from math import *
from matplotlib.pyplot import *
from numpy import *
from math import *
from math import pi as PI
from math import erf, exp

c = store_data()
time = len(c)
dx = find_dx()
R = find_center_mass(c,time)
c_R = create_centered(c,time,dx, R)
miyake = miyake_scale(c_R, time)
offset = 255

frame = ask_user()
x_0 = ask_position() + offset
c_work = zeros(time)
if 0 == frame :#scaled FF frame
    for i in range(time):
        c_work[i] = c[i][x_0]*miyake

else: #scaled CM frame
    for i in range(time):
        c_work[i] = c_R[i][x_0]*miyake

#find dc/dt
dcdt_left = zeros(time)
```

```
dcdt_right = zeros(time)
t = linspace(0,3*time,time)
#assign gradient to concentration point on the
    left
for i in range(time-1):
    dcdt_left[i] = lin_diff(c_work[i+1],
        c_work[i], t[i+1], t[i])
#assign gradient to concentration point on the
    left
for i in range(1,time):
    dcdt_right[i] = lin_diff(c_work[i], c_work
        [i-1], t[i], t[i-1])

#define grid
dcdt_low = -2
dcdt_high = 2.
dcdt_borders = 21
c_low = 0.
c_high = 10.
c_borders = 21

#area of a single rectangle in the grid for
    normalization
area = ((c_high - c_low)/(c_borders - 1.))*((
    dcdt_high - dcdt_low)/(dcdt_borders - 1.)
    )#area of each grid square (or rectangle)

#p(c) and associated domain
c_hist = histogram(c_work, bins=linspace(c_low
    ,c_high,c_borders), normed=0)
p_c = c_hist[0]
c_domain = zeros(len(p_c))
for i in range(len(c_domain)):
    c_domain[i] = (c_hist[1][i+1]+c_hist[1][i
        ])/2.

#p(dc/dt) and associated domain
dcdt_left_hist = histogram(dcdt_left, bins=
    linspace(dcdt_low, dcdt_high,
    dcdt_borders), normed=0)
dcdt_right_hist = histogram(dcdt_right, bins=
    linspace(dcdt_low, dcdt_high,
    dcdt_borders), normed=0)
p_dcdt_left = dcdt_left_hist[0]
p_dcdt_right = dcdt_right_hist[0]
dcdt_left_domain = zeros(len(p_dcdt_left))
dcdt_right_domain = zeros(len(p_dcdt_right))
for i in range(len(dcdt_left_domain)):
    dcdt_left_domain[i] = (dcdt_left_hist[1][i
        +1] + dcdt_left_hist[1][i])/2.
    dcdt_right_domain[i] = (dcdt_right_hist
        [1][i+1] + dcdt_right_hist[1][i])/2.

#Create the synthetic independent JPDF p(c)*p(
    dc/dt)
ipdf_left = zeros((c_borders-1, dcdt_borders
    -1))
ipdf_left_size = 0.
ipdf_right = zeros((c_borders-1, dcdt_borders
    -1))
ipdf_right_size = 0.
for k in range(c_borders-1):
    for i in range(dcdt_borders -1):
        ipdf_left[k][i] = p_dcdt_left[i]*p_c[k
            ]
        ipdf_left_size += ipdf_left[k][i]
        ipdf_right[k][i] = p_dcdt_right[i]*p_c
            [k]
        ipdf_right_size += ipdf_right[k][i]
#norm by number*grid size.
ipdf_left = ipdf_left/(ipdf_left_size*area)
ipdf_right = ipdf_right/(ipdf_right_size*area)

#Create JPDF p(dc/dt, c)
jpdf_left = zeros((c_borders-1, dcdt_borders
    -1))
jpdf_left_size = 0.
jpdf_right = zeros((c_borders-1, dcdt_borders
    -1))
jpdf_right_size = 0.
for k in range(c_borders-1):
    for i in range(dcdt_borders -1):
        for j in range(time):
            #set abs dcdt below if we "fold"
                along dc/dt = 0
            if (c_work[j] >= c_hist[1][k] and
                c_work[j] < c_hist[1][k+1] \
                #abs here
            and dcdt_left[j] >= (
                dcdt_left_hist[1][i]) and
                dcdt_left[j] < (
                dcdt_left_hist[1][i+1])):
                jpdf_left[k][i] += 1
                jpdf_left_size += 1
            if (c_work[j] >= c_hist[1][k] and
                c_work[j] < c_hist[1][k+1] \
                #abs here
            and dcdt_right[j] >= (
                dcdt_right_hist[1][i]) and
                dcdt_right[j] < (
                dcdt_right_hist[1][i+1])):
                jpdf_right[k][i] += 1
                jpdf_right_size += 1
#jpdf_size = jpdf_size*(c_high - c_low)/(
    c_borders - 1) * (dcdt_high - dcdt_low)/(
    dcdt_borders -1)
#norm
jpdf_left = jpdf_left/(jpdf_left_size*area)
jpdf_right = jpdf_right/(jpdf_right_size*area)


#jpdf_left and jpdf_right found by the two
    linear methods in sec.6.5.
#jpdf is set as the mean between them.
#indep checks for independence between p(c)
    and p(dc/dt).
jpdf = (jpdf_left + jpdf_right)/2.
ipdf = (ipdf_left + ipdf_right)/2.
indep = abs(jpdf - ipdf)

#create cutoffs for visualization
for i in range(c_borders-1):
    for j in range(dcdt_borders -1):
        if jpdf[i][j] > 0.2:
            jpdf[i][j] = 0.2
        if indep[i][j] > 0.2:
            indep[i][j] = 0.2


#Plot
fig = figure()
ax = fig.gca(projection='3d')
X,Y = meshgrid(dcdt_left_domain,c_domain)
surf = ax.plot_surface(X, Y, jpdf,rstride=1,
    cstride=1, cmap=cm.jet,
  linewidth=0, antialiased=False)
fig.colorbar(surf, shrink=1, aspect=10)
for item in ([ax.title, ax.xaxis.label, ax.
    yaxis.label, ax.zaxis.label]
    + ax.get_xticklabels() + ax.
        get_yticklabels() + ax.
        get_zticklabels()):
    item.set_fontsize(20)
ax.set_xlabel('dc/dt [ppm/s]')
ax.set_ylabel('c [ppm]')
ax.w_zaxis.line.set_lw(0.)
ax.set_zticks([])
show()
```

Listing A.22: A5_jpdf4.py

```
#Same as A5_jpdf.py, but for 4 data sets that
    I manually input when prompted.

from mpl_toolkits.mplot3d import Axes3D
from madona_mods import *
from matplotlib.pyplot import *
from numpy import *

query = ask_user()
time = 0
dx = 0
x_0 = ask_position()
c_list = [] #contains scaled c at sample
    position for 4 data sets
for k in range(4):
```

```
    c, dx = store_data_prompt()
    time = len(c)
    R = find_center_mass(c,time)
    c_R = create_centered(c,time,dx, R)
    miyake = miyake_scale(c_R, time)
    offset = 256
    if query == 0 :#scaled FF frame
        for i in range(time):
            c_list.append(miyake*c[i][x_0 +
                offset])
    else: #scaled CM frame
        for i in range(time):
            c_list.append(miyake*c_R[i][x_0 +
                offset])
c_work = asarray(c_list)
time = len(c_work)

#find dc/dt
dcdt_left = zeros(time)
dcdt_right = zeros(time)
t = linspace(0,3*time,time)
for i in range(time-1):
    dcdt_left[i] = lin_diff(c_work[i+1],
        c_work[i], t[i+1], t[i])
for i in range(1,time):
    dcdt_right[i] = lin_diff(c_work[i], c_work
        [i-1], t[i], t[i-1])

#define grid
dcdt_low = -2
dcdt_high = 2.
dcdt_borders = 41
c_low = 0.
c_high = 10.
c_borders = 41

#define area of each square for normalization
area = ((c_high - c_low)/(c_borders - 1.))*((
    dcdt_high - dcdt_low)/(dcdt_borders - 1.)
    )#area of each grid square (or rectangle)

#p(c)
c_hist = histogram(c_work, bins=linspace(c_low
    ,c_high,c_borders), normed=0)
p_c = c_hist[0]
c_domain = zeros(len(p_c))
for i in range(len(c_domain)):
    c_domain[i] = (c_hist[1][i+1]+c_hist[1][i
        ])/2.

#p(dc/dt)
dcdt_left_hist = histogram(dcdt_left, bins=
    linspace(dcdt_low, dcdt_high,
    dcdt_borders), normed=0)
dcdt_right_hist = histogram(dcdt_right, bins=
    linspace(dcdt_low, dcdt_high,
    dcdt_borders), normed=0)
p_dcdt_left = dcdt_left_hist[0]
p_dcdt_right = dcdt_right_hist[0]
dcdt_left_domain = zeros(len(p_dcdt_left))
dcdt_right_domain = zeros(len(p_dcdt_right))
for i in range(len(dcdt_left_domain)):
    dcdt_left_domain[i] = (dcdt_left_hist[1][i
        +1] + dcdt_left_hist[1][i])/2.
    dcdt_right_domain[i] = (dcdt_right_hist
        [1][i+1] + dcdt_right_hist[1][i])/2.

#p(dc/dt)*p(c)
ipdf_left = zeros((c_borders-1, dcdt_borders
    -1))
ipdf_left_size = 0.
ipdf_right = zeros((c_borders-1, dcdt_borders
    -1))
ipdf_right_size = 0.
for k in range(c_borders-1):
    for i in range(dcdt_borders -1):
        ipdf_left[k][i] = p_dcdt_left[i]*p_c[k
            ]
        ipdf_left_size += ipdf_left[k][i]
        ipdf_right[k][i] = p_dcdt_right[i]*p_c
            [k]
        ipdf_right_size += ipdf_right[k][i]
#norm
#norm by number*grid size.
ipdf_left = ipdf_left/(ipdf_left_size*area)
ipdf_right = ipdf_right/(ipdf_right_size*area)

#JPDF
jpdf_left = zeros((c_borders-1, dcdt_borders
    -1))
jpdf_left_size = 0.
jpdf_right = zeros((c_borders-1, dcdt_borders
    -1))
jpdf_right_size = 0.
for k in range(c_borders-1):
    for i in range(dcdt_borders -1):
        for j in range(time):
            #set abs dcdt below if we calp the
                sides
            if (c_work[j] >= c_hist[1][k] and
                c_work[j] < c_hist[1][k+1] \
                #abs here
            and dcdt_left[j] >= (
                dcdt_left_hist[1][i]) and
                dcdt_left[j] < (
                dcdt_left_hist[1][i+1])):
                jpdf_left[k][i] += 1
                jpdf_left_size += 1
            if (c_work[j] >= c_hist[1][k] and
                c_work[j] < c_hist[1][k+1] \
                #abs here
            and dcdt_right[j] >= (
                dcdt_right_hist[1][i]) and
                dcdt_right[j] < (
                dcdt_right_hist[1][i+1])):
                jpdf_right[k][i] += 1
                jpdf_right_size += 1
#jpdf_size = jpdf_size*(c_high - c_low)/(
    c_borders - 1) * (dcdt_high - dcdt_low)/(
    dcdt_borders -1)
#norm
jpdf_left = jpdf_left/(jpdf_left_size*area)
jpdf_right = jpdf_right/(jpdf_right_size*area)

#Check independence
jpdf = (jpdf_left + jpdf_right)/2.
ipdf = (ipdf_left + ipdf_right)/2.
indep = abs(jpdf - ipdf)

#create cutoffs
for i in range(c_borders-1):
    for j in range(dcdt_borders -1):
        if jpdf[i][j] > 0.2:
            jpdf[i][j] = 0.2
        if indep[i][j] > 0.2:
            indep[i][j] = 0.2

#3D plot
fig = figure()
ax = fig.gca(projection='3d')
X,Y = meshgrid(dcdt_left_domain,c_domain)
surf = ax.plot_surface(X, Y, jpdf,rstride=1,
    cstride=1, cmap=cm.jet,
  linewidth=0, antialiased=False)
fig.colorbar(surf, shrink=1, aspect=10)
for item in ([ax.title, ax.xaxis.label, ax.
    yaxis.label, ax.zaxis.label]
    + ax.get_xticklabels() + ax.
        get_yticklabels() + ax.
        get_zticklabels()):
    item.set_fontsize(20)
ax.set_xlabel('dc/dt [ppm/s]')
ax.set_ylabel('c [ppm]')
ax.w_zaxis.line.set_lw(0.)
ax.set_zticks([])
show()
```

Listing A.23: A5_jpdf_gaussian.py

```
#create Gaussian Smoothed coating for JPDF

from mpl_toolkits.mplot3d import Axes3D
from madona_mods import *
from matplotlib.pyplot import *
from numpy import *
```

```
from math import *
from math import pi as PI
from math import erf, exp
import sys

#Setup
c = store_data()
time = len(c)
dx = find_dx()
R = find_center_mass(c,time)
c_R = create_centered(c,time,dx, R)
miyake = miyake_scale(c_R, time)
offset = 255
frame = ask_user()
x_0 = ask_position() + offset
c_work = zeros(time)
if 0 == frame :#scaled FF frame
    for i in range(time):
        c_work[i] = c[i][x_0]*miyake

else: #scaled CM frame
    for i in range(time):
        c_work[i] = c_R[i][x_0]*miyake

#dc/dt
dcdt_left = zeros(time)
dcdt_right = zeros(time)
t = linspace(0,3*time,time)
for i in range(time-1):
    dcdt_left[i] = lin_diff(c_work[i+1],
        c_work[i], t[i+1], t[i])
for i in range(1,time):
    dcdt_right[i] = lin_diff(c_work[i], c_work
        [i-1], t[i], t[i-1])

#Define grid
dcdt_low = -2
dcdt_high = 2.
dcdt_borders = 21
c_low = 0.
c_high = 10.
c_borders = 21
area = ((c_high - c_low)/(c_borders - 1.))*((
    dcdt_high - dcdt_low)/(dcdt_borders - 1.)
    )#area of each grid square (or rectangle)

#p(c)
c_hist = histogram(c_work, bins=linspace(c_low
    ,c_high,c_borders), normed=0)
p_c = c_hist[0]
c_domain = zeros(len(p_c))
for i in range(len(c_domain)):
    c_domain[i] = (c_hist[1][i+1]+c_hist[1][i
        ])/2.

#p(dc/dt)
dcdt_left_hist = histogram(dcdt_left, bins=
    linspace(dcdt_low, dcdt_high,
    dcdt_borders), normed=0)
dcdt_right_hist = histogram(dcdt_right, bins=
    linspace(dcdt_low, dcdt_high,
    dcdt_borders), normed=0)
p_dcdt_left = dcdt_left_hist[0]
p_dcdt_right = dcdt_right_hist[0]
dcdt_left_domain = zeros(len(p_dcdt_left))
dcdt_right_domain = zeros(len(p_dcdt_right))
for i in range(len(dcdt_left_domain)):
    dcdt_left_domain[i] = (dcdt_left_hist[1][i
        +1] + dcdt_left_hist[1][i])/2.
    dcdt_right_domain[i] = (dcdt_right_hist
        [1][i+1] + dcdt_right_hist[1][i])/2.

#p(dc/dt,c)
jpdf_left = zeros((c_borders-1, dcdt_borders
    -1))
jpdf_left_size = 0.
jpdf_right = zeros((c_borders-1, dcdt_borders
    -1))
jpdf_right_size = 0.
for k in range(c_borders-1):
    for i in range(dcdt_borders -1):
        for j in range(time):
            #set abs dcdt below if we calp the
                sides
            if (c_work[j] >= c_hist[1][k] and
                c_work[j] < c_hist[1][k+1] \
                #abs here
            and dcdt_left[j] >= (
                dcdt_left_hist[1][i]) and
                dcdt_left[j] < (
                dcdt_left_hist[1][i+1])):
                jpdf_left[k][i] += 1
                jpdf_left_size += 1
            if (c_work[j] >= c_hist[1][k] and
                c_work[j] < c_hist[1][k+1] \
                #abs here
            and dcdt_right[j] >= (
                dcdt_right_hist[1][i]) and
                dcdt_right[j] < (
                dcdt_right_hist[1][i+1])):
                jpdf_right[k][i] += 1
                jpdf_right_size += 1

#norm
jpdf_left = jpdf_left/(jpdf_left_size*area)
jpdf_right = jpdf_right/(jpdf_right_size*area)
jpdf = (jpdf_left + jpdf_right)/2.

#Smooth along c
gausslet_points = (c_borders-1)*100 - 1
domain_c = linspace(-c_high,2*c_high,
    gausslet_points)
c_dist = domain_c[1] - domain_c[0]
pdf_domain = linspace(c_low,c_high,num=
    c_borders)
pdf_env = zeros((gausslet_points,dcdt_borders
    -1))
distance = c_domain[1] - c_domain[0]
for k in range(dcdt_borders-1):
    small_gauss = zeros((len(c_domain),
        gausslet_points))
    for i in range(c_borders-1):
        local_pd = jpdf[i][k]
        small_gauss[i] = gausslet(c_high,
            c_domain[i], 2*distance, local_pd
            , gausslet_points)
    #mesh together
    for j in range(gausslet_points):
        for i in range(c_borders-1): #no.
            poles
            pdf_env[j][k] += small_gauss[i][j]

#cutoff below p= 0.5 and for c<0 and c>0.
cut_low = 0
cut_high = 0
for i in range(len(domain_c)):
    if domain_c[i] > 0 and cut_low == 0:
        #smallest point over 0
        cut_low = i
    if domain_c[i] > 10 and cut_high == 0:
        #smallest point over 10
        cut_high = i
cut_diff = cut_high - cut_low
plot_pdf = zeros((cut_diff, dcdt_borders-1))
plot_c = zeros(cut_diff)
for j in range(cut_diff):
    plot_c[j] = domain_c[j+cut_low]
    for i in range(dcdt_borders-1):
        if pdf_env[j+cut_low][i] > 0.5:
            plot_pdf[j][i] = 0.5
        else:
            plot_pdf[j][i] = pdf_env[j+cut_low
                ][i]

#3D plot
fig = figure()
ax = fig.gca(projection='3d')
X,Y = meshgrid(dcdt_left_domain,plot_c)
surf = ax.plot_wireframe(X,Y,plot_pdf,rstride
    =10, cstride=1, linewidth=1, color='b')
#Commented out below: if we want surface color
    plots instead.
#surf = ax.plot_surface(X, Y, plot_pdf,rstride
    =1, cstride=1, cmap=cm.jet,
#   linewidth=0, antialiased=False)
#fig.colorbar(surf, shrink=1, aspect=10)
```

```
for item in ([ax.title, ax.xaxis.label, ax.
    yaxis.label, ax.zaxis.label]
        + ax.get_xticklabels() + ax.
            get_yticklabels() + ax.
            get_zticklabels()):
        item.set_fontsize(20)
ax.set_xlabel('dc/dt␣[ppm/s]')
ax.set_ylabel('c␣[ppm]')
ax.w_zaxis.line.set_lw(0.)
ax.set_zticks([])
show()
```

Listing A.24: A5_compare.py

```
#Compare <T> and f_cross by sampling raw data
    and against using JPDF and PDF.

from madona_mods import *
from math import *
from matplotlib.pyplot import *
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.ticker import LinearLocator,
    FormatStrFormatter
from numpy import *

c = store_data()
time = len(c)
dx = find_dx()
R = find_center_mass(c,time)
c_R = create_centered(c,time,dx, R)
miyake = miyake_scale(c_R, time)
offset = 255

query = ask_user()
x0 = ask_position() + offset
if query == 0 :#scaled FF frame
    c_work = c[:,x0]*miyake
else: #scaled CM frame
    c_work = c_R[:,x0]*miyake

positions = len(c_work)
samples = 500 #number of sample concentration
    thresholds
upper_threshold = 20.
lower_threshold = 0.

#\% t and crossings by interpolating data
threshold = linspace(lower_threshold,
    upper_threshold, samples) # x-axis
time_spent = zeros(samples)
crossings = zeros(samples)
position = ask_position() + offset
for k in range(samples):
    time_spent[k], crossings[k] = time_cross(
        c_work, threshold[k], time)

#norm
time_spent = time_spent/float(time) #to get in
     units of 1
crossings = crossings/(3*time) #in terms of
    units of 1

#dc/dt
dcdt_left = zeros(time)
dcdt_right = zeros(time)
t = linspace(0,3*time,time)
for i in range(time-1):
    dcdt_left[i] = lin_diff(c_work[i+1],
        c_work[i], t[i+1], t[i])
for i in range(1,time):
    dcdt_right[i] = lin_diff(c_work[i], c_work
        [i-1], t[i], t[i-1])

#grid
dcdt_low = 0
dcdt_high = 3.
dcdt_borders = 41
c_low = 0.
c_high = 20.
c_borders = 41
area = ((c_high - c_low)/(c_borders - 1.))*((
    dcdt_high - dcdt_low)/(dcdt_borders - 1.)
    )#area of each grid square (or rectangle)

#p(c)
c_hist = histogram(c_work, bins=linspace(c_low
    ,c_high,c_borders), normed=0)
p_c = c_hist[0]
c_domain = zeros(len(p_c))
for i in range(len(c_domain)):
    c_domain[i] = (c_hist[1][i+1]+c_hist[1][i
        ])/2.

#p(dc/dt)
dcdt_left_hist = histogram(dcdt_left, bins=
    linspace(dcdt_low, dcdt_high,
    dcdt_borders), normed=0)
dcdt_right_hist = histogram(dcdt_right, bins=
    linspace(dcdt_low, dcdt_high,
    dcdt_borders), normed=0)
p_dcdt_left = dcdt_left_hist[0]
p_dcdt_right = dcdt_right_hist[0]
dcdt_left_domain = zeros(len(p_dcdt_left))
dcdt_right_domain = zeros(len(p_dcdt_right))
for i in range(len(dcdt_left_domain)):
    dcdt_left_domain[i] = (dcdt_left_hist[1][i
        +1] + dcdt_left_hist[1][i])/2.
    dcdt_right_domain[i] = (dcdt_right_hist
        [1][i+1] + dcdt_right_hist[1][i])/2.
p_dcdt = (p_dcdt_left + p_dcdt_right)/2.

#p(c,dc/dt)
jpdf_left = zeros((c_borders-1, dcdt_borders
    -1))
jpdf_left_size = 0.
jpdf_right = zeros((c_borders-1, dcdt_borders
    -1))
jpdf_right_size = 0.
for k in range(c_borders-1):
    for i in range(dcdt_borders -1):
        for j in range(time):
            #set abs dcdt below if we calp the
                sides
            if (c_work[j] >= c_hist[1][k] and
                c_work[j] < c_hist[1][k+1] \
                #abs here
            and dcdt_left[j] >= abs(
                dcdt_left_hist[1][i]) and
                dcdt_left[j] < abs(
                dcdt_left_hist[1][i+1])):
                jpdf_left[k][i] += 1
                jpdf_left_size += 1
            if (c_work[j] >= c_hist[1][k] and
                c_work[j] < c_hist[1][k+1] \
                #abs here
            and dcdt_right[j] >= abs(
                dcdt_right_hist[1][i]) and
                dcdt_right[j] < abs(
                dcdt_right_hist[1][i+1])):
                jpdf_right[k][i] += 1
                jpdf_right_size += 1

jpdf_left = jpdf_left/(jpdf_left_size*area)
jpdf_right = jpdf_right/(jpdf_right_size*area)
jpdf = (jpdf_left + jpdf_right)/2.

#calculate pdf_cross, which is normalized
    crossings frequency
#found from the JPDF
c_hist = histogram(c_work, bins=linspace(c_low
    ,c_high,c_borders), normed=1)
p_c = c_hist[0]
pdf_cross = zeros(len(p_c))
dcdt_left_hist = histogram(dcdt_left, bins=
    linspace(dcdt_low, dcdt_high,
    dcdt_borders), normed=1)
dcdt_right_hist = histogram(dcdt_right, bins=
    linspace(dcdt_low, dcdt_high,
    dcdt_borders), normed=1)
p_dcdt_left = dcdt_left_hist[0]
p_dcdt_right = dcdt_right_hist[0]
dcdt_left_domain = zeros(len(p_dcdt_left))
dcdt_right_domain = zeros(len(p_dcdt_right))
for i in range(len(dcdt_left_domain)):
    dcdt_left_domain[i] = (dcdt_left_hist[1][i
```

```
        +1] + dcdt_left_hist[1][i])/2.
    dcdt_right_domain[i] = (dcdt_right_hist
        [1][i+1] + dcdt_right_hist[1][i])/2.
p_dcdt = (p_dcdt_left + p_dcdt_right)/2.

dc = (c_high - c_low)/(c_borders - 1)
dcc = (dcdt_high - c_low)/(c_borders -1)
dcdt = dcdt_left_hist[1][:-1]

#t_ex takes normalized %t calculated from pdf.
for i in range(len(p_c)):
    for j in range(len(p_dcdt)):
        pdf_cross[i] += dcdt[j]*jpdf[i][j]
t_ex = zeros(len(p_c))
#"integrate" from largest to c'_i
for i in range(len(p_c)): #loop c_t from small
     to high
    for j in range(len(p_c) - i):
        t_ex[i] += p_c[len(p_c) -j-1]
t_ex = t_ex*dc
pdf_cross = pdf_cross*area
t_exp = t_ex/pdf_cross
c_domain = c_hist[1][:-1]

#Plot currently set to crossings.
#Change to t_exp and time_spent for %t
rc('xtick', labelsize=30)
rc('ytick', labelsize=30)
rc('text', usetex=True)
plot(threshold, crossings, 'b', linewidth=3,
    label='Using Linear Interpolation')
hold('on')
plot(c_domain, pdf_cross, 'r', linewidth=3,
    label='Using JPDF')
legend(fontsize='40')
show()
```

A.6 General Functions

Listing A.25: madona_mods.py

```
#contains all general functions called during
    data processing

import sys
from numpy import *
from matplotlib.pyplot import *
from math import pi as PI

#functions below for reading mad data files.
#these are the raw data files Bjorn Lybekk
    helped me get over to plain text format.
#functions with _prompt only read when
    prompted.
#these are for codes that await input of
    several data files in succession.
def find_total_time():
    #"records" were the number of samples in
        time in original data sets
    g = open(sys.argv[1],'r')
    for line in g:
        line.strip()
        words = line.split()
        if str("records") in words:
            total = int(words[3])
            g.close()
            return total

def find_dx():
    h = open(sys.argv[1],'r')
    for line in h:
        line.strip()
        words = line.split()
        #"bereich" gave a code for dx
        #in the original data files
        if str("bereich::") in words:
            characters = list(words[1])
            if int(characters[0]) == 3: #
                bereich 300
                scaling = 0.6
            elif int(characters[0]) == 7: #
                bereich 750
                scaling = 1.5
            else:
                print 'Something went wrong, 
                    cant read bereich.'
            break
    h.close()
    return scaling

def find_total_time_prompt(address):
    g = open(address,'r')
    for line in g:
        line.strip()
        words = line.split()
        if str("records") in words:
            total = int(words[3])
            g.close()
            return total

def find_dx_prompt(address):
    #Reads 'bereich' from file
    h = open(address,'r')
    for line in h:
        line.strip()
        words = line.split()
        if str("bereich::") in words:
            characters = list(words[1])
            if int(characters[0]) == 3: #
                bereich 300
                scaling = 0.6
            elif int(characters[0]) == 7: #
                bereich 750
                scaling = 1.5
            else:
                print 'Something went wrong, 
                    cant read bereich.'
            break
    h.close()
    return scaling

def store_data():
    #reads data into a two-dimensional array
        of c[time][position]
    f = open(sys.argv[1],'r')
    total_time = find_total_time()
    c = zeros((total_time,512))

    #TEST will be set to the value of the
        first number of each sentence
    #(if it is a number)
    #only the first time-series of data have '
        sentences' starting
    #with 0, and this is the condition for
        starting to read
    test = 1
    #FLAG activates once TEST has done its job
        , and makes sure
    #we continue reading without checking for
        sentences starting with 0.
    flag = 0
    position_counter = 0
    time_counter = 0
    empty_arrays = 0 #keeps count of no. empty
        subarrays
    for line in f:
        line.strip()
        numbers = line.split()
        if flag == 0:
            try:
                test = int(numbers[0])
                if test == 0:
                    #start processing
                    flag = 1
                    c[0][0] = float(numbers
                        [2])
                    position_counter += 1
            except:
                flag = 0
        if flag == 1:
            #continue processing
            c[time_counter][
```

```
                    position_counter] = float
                    (numbers[2])
                position_counter+=1
            if position_counter==512:
                #increment time. check for
                    empty data set.
                test_sum = 0
                position_counter= 0
                for j in range(512):
                    test_sum += c[time_counter
                        ][j]
                if test_sum != 0: #not empty
                    data set
                    time_counter+=1
                else:
                    #erase data set and keep
                        count
                    empty_arrays += 1

                #condition for ending read
                if time_counter == total_time:
                    break
        f.close()
        #clean away empty subarrays
        if empty_arrays != 0:
            for i in range(empty_arrays):
                c = delete(c, (total_time-i-1),
                    axis=0)
        return c

def store_data_prompt():
        #reads data into a two-dimensional
            array of c[time][position]
        #this version prompts the reader to
            give the location of the data
        address = raw_input("Enter location
            data is stored in: ")
        f = open(address,'r')
        total_time = find_total_time_prompt(
            address)
        c = zeros((total_time,512))
        dx = find_dx_prompt(address)
        test = 1
        flag = 0
        position_counter = 0
        time_counter = 0
        empty_arrays = 0 #keeps count of no.
            empty subarrays
        for line in f:
            line.strip()
            numbers = line.split()
            if flag == 0:
                try:
                    test = int(numbers[0])
                    if test == 0:
                        #start processing
                        flag = 1
                        c[0][0] = float(
                            numbers[2])
                        position_counter += 1
                except:
                    flag = 0
            if flag == 1:
                #continue processing
                c[time_counter][
                    position_counter] = float
                    (numbers[2])
                position_counter+=1
            if position_counter==512:
                #increment time. check for
                    empty data set.
                test_sum = 0
                position_counter= 0
                for j in range(512):
                    test_sum += c[time_counter
                        ][j]
                if test_sum != 0: #not empty
                    data set
                    time_counter+=1
                else:
                    #erase data set and keep
                        count
                    empty_arrays += 1

                #condition for ending read
                if time_counter == total_time:
                    break
        f.close()
        #clean away empty subarrays
        if empty_arrays != 0:
            for i in range(empty_arrays):
                c = delete(c, (total_time-i-1),
                    axis=0)
        return c, dx

#ask about reference frame
def ask_user():
    f = open(sys.argv[1],'r')
    n = int(raw_input("Type 0 fixed frame.
        Type 1 for CM frame: "))
    if n==0 or n==1:
        f.close()
        return n
    else:
        print "Something else than 0 or 1
            written."
        exit()

#below are functions related to CM
def find_center_mass(c, time):
    #Will find the center of mass R(t).
    uR = 0 #will store unnormalized center of
        mass
    newtime = len(c)
    R = zeros(newtime) #will store center of
        mass of each time-spot
    sum_c = 0 #total concentration
    for dt in range(0,newtime):
        for dx in range(0,512):
            uR += c[dt][dx]*dx
            sum_c += c[dt][dx]
        R[dt] = uR/float(sum_c)
        uR = 0
        sum_c = 0
        try:
            R[dt] = int(R[dt])
        except ValueError:
            R[dt] = 0
    return R

def create_centered(c, time, dx, R):
    #Will center arrays around the center of
        mass.
    newtime = len(c)
    conc_com = zeros((newtime,512))
    shift = 0
    for i in range(time):
        shift = 256 - R[i]
        for j in range(512):
            try:
                conc_com[i][j] = c[i][j-shift]
            except:
                #make 0s away from center of
                    mass
                conc_com[i][j] = 0
    return conc_com

#below function scales concentration to Miyake
    values
def miyake_scale(c_R, time):
    c_mean_center = 0 #this will hold the mean
        concentration of the center of mass
    for j in range(21):
        for i in range(time):
            c_mean_center += c_R[i][256-10+j]
    c_mean_center = c_mean_center/(time*21.)
    measured_maximal = 4.27 #or 17.25, these
        are 5 min avgs.
    scaling_ratio = 4.27/c_mean_center #
        measured/data
    return scaling_ratio

#asks about array position from CM or center
    of measurement
def ask_position():
    n = int(raw_input("How far in array units
```

```
        from center of array? "))
    return n

#below are functions for finding moments their
     related properties
def get_mean(c, time):
    mu = zeros(512)
    for j in range(512):
        for i in range(time):
            mu[j] += c[i][j]
    mu = mu/(1.*time)
    return mu

def get_sigma(c, time, mu):
    sigma = zeros(512)
    for j in range(512):
        for i in range(time):
            sigma[j] += c[i][j]**2
        sigma[j] = sigma[j]/(1.*time)
    for i in range(len(sigma)):
        sigma[i] += - mu[i]**2
        sigma[i] = sqrt(sigma[i])
    return sigma

def get_skewness(c, time, mu, sigma):
    skew = zeros(512)
    for j in range(512):
        for i in range(time):
            skew[j] += c[i][j]**3
    skew = skew/time
    skew += - mu**3
    skew = skew/(sigma**3)
    skew += - 3*mu/sigma
    return skew


def get_kurtosis(c, time, mu, sigma):
    kurt = zeros(512)
    for j in range(512):
        for i in range(time):
            kurt[j] += (c[i][j] - mu[j])**4
    kurt = kurt/time
    kurt = kurt/(sigma**4)
    return kurt

def get_moment(c, time, mean, power):
    moment = zeros(512)
    for j in range(512):
        for i in range(time):
            moment[j] = moment[j] + (c[i][j] -
                mean[j])**power
    moment = moment/time
    return moment

def find_max(c):
    maximum = 0
    for i in range(512):
        if (maximum < c[i]):
            maximum = c[i]
    return maximum

#linear interpolation calc of crossings and
     time above
def time_cross(signal, limit, time):
    #using linear interpolation
    #takes a 1-dim array 'signal', finds total
         time spent above thresholds
    #and number of crossings
    total_time = 0.
    total_cross = 0
    flag_time = 0
    temp = 0
    for i in range(time):
        if flag_time == 0:
            if signal[i] > limit:
                flag_time = 1
                total_cross += 1
                temp = 1- interpol_thresh(
                    float(signal[i]), float(
                    signal[i-1]), limit)
                total_time += temp
        else:
            if signal[i] <= limit:
                flag_time = 0
                temp = interpol_thresh(float(
                    signal[i]), float(signal[
                    i-1]), limit)
                total_time += temp
            else:
                total_time += 1
    return total_time, total_cross #in terms
        of 3s units

def interpol_thresh(c_f, c_0, c_t):
    #Returns fraction of 1 unit of time
        according to two-point linear
        interpolation of crossing.
    time_crossed = (c_t - c_0)/(c_f - c_0)
    return time_crossed


#brute force calc. of crossings and time above
def time_cross_wo(signal, limit, time):
    #without using linear interpolation
    #takes a 1-dim array 'signal', finds total
         time spent above thresholds
    #and number of crossings
    total_time = 0.
    total_cross = 0
    flag_time = 0
    for i in range(time):
        if flag_time == 0:
            if signal[i] > limit:
                flag_time = 1
                total_cross += 1
                total_time += 1
        else:
            if signal[i] <= limit:
                flag_time = 0
            else:
                total_time += 1
    return total_time, total_cross #in terms
        of 3s units

#functions below are needed for Gaussian
     Smoothing
def gaussian(c_mean, sigma, domain):
    f = zeros(len(domain))
    for i in range(len(f)):
        f[i] = exp(-(double(domain[i] - c_mean
            ))**2/(2*sigma**2))
    f = f*(1./(sqrt(2.*PI)*sigma))
    return  f

def gausslet(max_c, mean, d, height,
     gausslet_points):
    gauss_array = zeros(gausslet_points)
    #domain is always from 1 pdf domain to the
         left to 1 pdf domain
    #to the right of the actual pdf domain
    x = linspace(-max_c, 2*max_c,
        gausslet_points)
    sigma = d
    for i in range(gausslet_points):
        gauss_array[i] = exp(-((x[i]-mean)**2)
            /(2.*sigma**2))
    gauss_array = gauss_array*height*d/(sigma*
        sqrt(2*PI))
    return gauss_array

#linear gradient as used for finding dc/dt
def lin_diff(c_R, c_L, t_R, t_L):
    temp=(c_R - c_L)/(t_R - t_L)
    return temp
```