Delft University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science

PROBABILISTIC INVERSION METHODS IN AN INVERSE GAS DISPERSION PROBLEM FOR LOCATING HYDROCARBON RESERVOIRS

Master's thesis

Lizyayev, Andrey M.

Delft, the Netherlands
2005

Table of contents

Abstract

A new oil and gas prospecting technique based on measurements of sub-part-per-billion ethane concentration in the atmosphere and local wind data was reported in the LightTouch project between Shell Global Solutions BV and the University of Glasgow. An ultra-sensitive gas sensor measures the atmospheric concentrations of ethane dispersing from the naturally occurring ethane seepages that accompany all hydrocarbon reservoirs. Typical seepages are detectable from a range of several km downwind. The goal of this MSc project at TU Delft was to propose alternatives to the binary search optimisation scheme previously employed to locate and quantify the seepage sources from the concentration and wind data, used in conjunction with a gas dispersion model. Several approaches were considered and 2 Classical Probabilistic Inversion methods – Iterative Proportional Fitting, IPF, and Parametric Fitting for Uncertain Models, PARFUM – were selected for implementation, evaluation and development. Subsequently, a new Probabilistic Inversion method was developed, offering significant operational advantages: faster convergence and more efficient use of memory. The new method retained key attributes of the Probabilistic Inversion approach, namely: confidence estimates for the results and freedom from arbitrary cost functions and regularisation issues.

KEYWORDS: Gaussian plume, inverse problems, gas dispersion modelling, Probabilistic Inversion, IPF, PARFUM.

Introduction

Background

It is recognised that oil and gas reservoirs gradually leak some of their contents to the surface [*Horvitz*, 1985; *Schumacher,* 1999] and for the first 100 years, much prospecting centred around finding surface signs of hydrocarbons. Initially such signs were: pools of oil or tar, burning gas on hillsides and gas escaping through cracks in the ground. As these discoveries were depleted, attention shifted towards finding the geological structures capable of trapping hydrocarbons rather than the hydrocarbons themselves.

Several direct hydrocarbon detection methods, that exploit advances in sensor technologies, are in use today. Offshore, these include searching satellite and airborne images for subtle signs of oil on the sea-surface [*MacDonald et al.*, 1996] and the stimulation of fluorescence in these oil films by airborne lasers [*Martin and Cawley*, 1991]. Onshore techniques range from the direct sampling of gases in the soil measured subsequently by laboratory-based gas chromatography [*Jones et al.*, 2000] to the accumulation of hydrocarbons on buried adsorbent pellets [*El-Bishlawy et al.*, 2001]. These point sampling techniques require extensive physical coverage of the area, making them unsuited to large surveys. Also, for some methods the results are not available for several months.

A new tool for the measurement of hydrocarbon gases seeping from reservoirs was recently reported by Shell Global Solutions in cooperation with the Optics group of the University of Glasgow within the LightTouch project. Methane and ethane are both indicators of reservoirs and detecting either could be significant. However, methane arises from numerous organic processes – and consequently has an atmospheric background concentration of ~1.7 parts per million (ppm) [*Houghton et al.*, 1996]. Against this relatively high concentration only very large and nearby sources would be discernable; furthermore, they could still correspond to bacterial action rather than an oil or gas reservoir. In contrast, there are no significant biogenic sources of ethane and the atmospheric background concentration is approximately 1 part per billion (ppb), making ethane an ideal indicator of oil or gas reservoirs [*Jones et al.*, 2000].

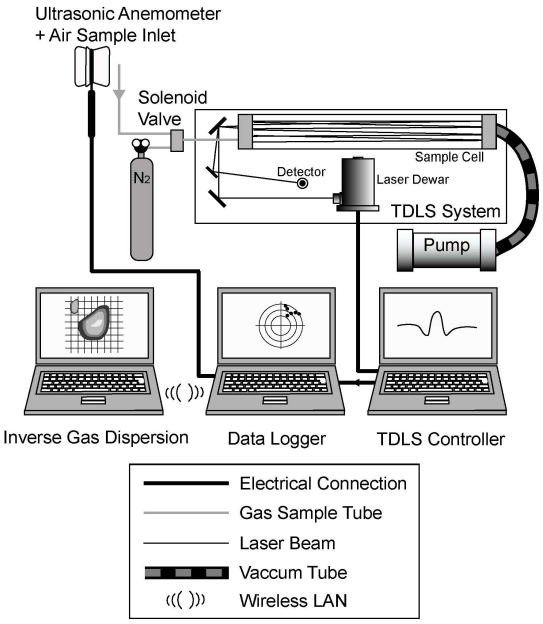


Figure 1. Configuration of the Tuneable Diode Laser Spectroscopy system (TDLS), wind velocity data logger and inverse gas dispersion modelling. Air is drawn into the multiple-pass sample cell (an astigmatic Herriott cell), which has an effective optical path length of 208 metres. The solenoid valve selects between sampled air and pure nitrogen, providing automated zero checks. The data logger combines wind data from the anemometer with the corresponding ethane concentrations. The resulting data files are downloaded to a computer in the support vehicle via a wireless LAN, allowing ethane flux maps to be calculated during the survey.

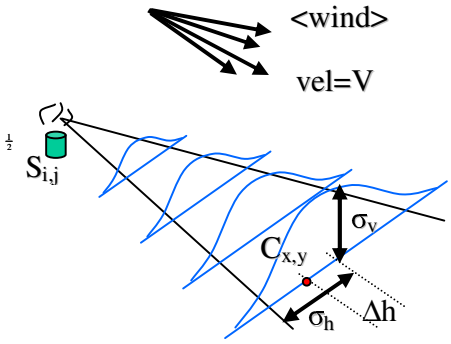
The direct approach to prospecting by ethane detection would be to traverse the survey area searching for emissions. This would be time consuming, require extensive access to difficult locations and would be susceptible to inconsistencies in sampling due to the wind as well as disturbance by the survey itself. More attractive is to use the wind driven gas dispersion to gain information over an extensive upwind area. Recording gas concentrations, using an extremely sensitive detector, and the corresponding wind data allows large areas to be rapidly screened for sources and for the surface emission flux to be determined by inverting the gas dispersion process. The measurement of flux will contribute to ongoing research into the mechanisms responsible for seepages [*Klusman and Saeed*, 1996].

Figure 2. The laser diode spectrometer, control hardware, gas handling system, data logger and liquid nitrogen supply are mounted in an off-road vehicle. Although the instrument is operated continuously, best noise performance is obtained with the vehicle stationary. Typically 15 minutes of data are recorded at each measurement position. A telescopic mast places the sample inlet and anemometer 5m above the ground.

Problem formulation

Calculating gas dispersion requires knowledge of the entire wind field over the area of interest but this is impractical for our application. Instead one can generalise the wind field from a single location near the gas measurement point. An ultrasonic anemometer was mounted on the vehicle mast to measure both horizontal and vertical components of the wind speed with an accuracy of 0.1m/s.

Many groups have considered the modelling of gas dispersion and providing the data is averaged over a comparable time to the time of flight from the source to the measurement position, a simple Gaussian plume is a good approximation [*Gifford*, 1976]. The survey area is represented by an array of point sources arranged on a uniform grid. The predicted atmospheric concentration at any position $C(x, y)$ is related to the magnitude of the point sources $S_{i,j}$, the wind speed V, the offset from the plume centre Δ, the width σ_h and height σ_v of the Gaussian plume by the equation



$$C(x, y) = \frac{S_{i,j}}{\pi V \sigma_h \sigma_v} \left(\exp - \frac{1}{2} \left(\Delta / \sigma_h \right)^2 \right) \qquad (1)$$

The concentration given by (1) is a mass concentration. To get the concentration by volume divide (1) by the gas density. Further in the text the term “concentration” will refer to the mass concentration, unless otherwise specified.

The width and height of the Gaussian plume are determined from the variability in wind direction over the averaging time. The validity of this gas plume model is

substantially reduced if convection is significant and hence a lower bound of 2m/s was set on the wind speed (i.e. data with a lower value of wind speed were not recorded).

Equation (1) relates a single measurement of gas concentration and corresponding wind data to the magnitude of every potential source within the survey grid, giving an equation with many unknowns corresponding to each grid box. Assuming that none of the sources is time varying, multiple measurements from different locations and/or under different wind conditions yield a set of simultaneous equations in the following way.

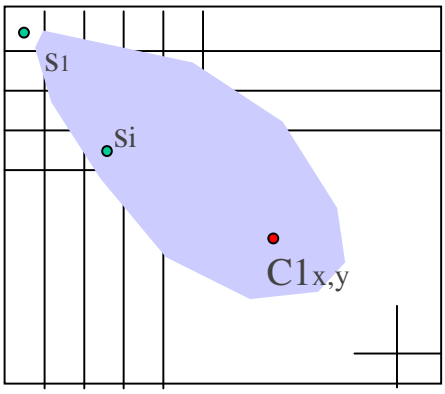
Divide the survey area into a grid of n cells, each containing an emission source at its centre. The concentration Y(x,y) that would result from a given source pattern S = (s_1, …, s_n) at the point (x,y), given wind characteristics would then be

$$Y(x,y) = a_1 s_1 + \ldots + a_n s_n \qquad (2)$$

with a_i determined by (1).

So, if C_1, …,C_m are the actual measured concentrations (not necessarily all in different locations), then the system of equations becomes

$$\begin{cases} C_1 = a_{11}s_1 + a_{12}s_2 + \ldots + a_{1n}s_n \\ C_j = a_{j1}s_1 + a_{j2}s_2 + \ldots + a_{jn}s_n \\ \ldots \\ C_m = a_{m1}s_1 + a_{m2}s_2 + \ldots + a_{mn}s_n \end{cases}$$



or, in matrix notations

$$C = A*S \qquad (3)$$

In general, such a system of equations is over determined (more equations than unknowns), sparse (only a few of the potential sources are related to each concentration measurement) and ill-conditioned (many of our equations have similar coefficients). The problem is to find a source vector S whose resulting concentrations are in the best possible agreement with the concentrations C actually measured, given the Gaussian Plume model coefficients A.

Previously developed approach

The Optics Group of Glasgow University had developed a binary search technique that iteratively optimised a spatial source distribution to find one minimising a specified "cost function". The main term in this cost function corresponds to seeking the best agreement between the measured and predicted concentrations, usually defined by the total square error (often referred to as "chi-squared" by the Group). This optimization approach was implemented by the Group in LabView and offers a variety of different cost functions. Of these, a cost function provided by the product of the total square error and a factor related to the total flux of the proposed source distribution, is believed to be robust.

$$\text{cost} = \frac{\chi^2 \times (totalflux\ estimate + total\ \ flux)}{totalflux\ \ estimate}$$

In essence this finds the optimum source distribution for the minimum total flux.

Starting from an initial random distribution, the algorithm selects a grid position at random and sets its source to a random positive value. If this change reduces the value of the cost function, then it is kept, else it is discarded. Repeating this process leads to a stable distribution of sources. Despite the random nature of the initial distribution and each iteration, multiple executions of the programme give the same final source distribution.

While this multiplicative cost function has the advantage of avoiding the dilemma of how to trade off total square error against flux; it has the undesirable effect of introducing bias in the results. Since it aims to minimise total flux, there is a preference for placing sources close to the measurement location – as smaller sources will then "explain" large concentrations. This is undesirable, and a variety of different cost functions have – and are still being – evaluated by the Glasgow group.

Shell Global Solutions approached the Statistics group of the Delft University of Technology with the following project goals:

- A review of the mathematical and statistical approaches that might be applicable to the LightTouch gas dispersion inversion problem.
- A proposal for a fundamentally different approach to the binary search technique already developed.
- Suggestions of ways to adapt or improve the existing binary search methodology:
    - Better-suited cost functions or other regularisation approaches.
    - Ways of characterising confidence in individual sources within the resulting flux source distributions.
    - Different search/minimisation strategies for locating the optimum source distribution consistent with our data.
    - Ways of determining the optimum no of cells to divide the source field into from a consideration of the absolute error and number of measurements available.

The Alternative Approaches Considered

The main goal of the MSc project was to propose a fundamentally different approach to the binary search technique already developed.

The following approaches were considered for the project:

- Optimisation (Interior point) methods
- QR Factorisation
- Kalman Filtering
- Variational Methods.
- Probabilistic Inversion.

Of these the Probabilistic Inversion approach was considered most likely to provide a successful and distinctively different approach that could be implemented, tested and developed within the time-scale of the proposed project.

The alternative approaches will be described in a later section.

.

Probabilistic inversion

The Probabilistic Inversion approach was chosen because it offered several advantages:

- it does not involve ad-hoc cost functions and regularisation.
- It offers an objective confidence measure for the sources identified.
- It has been previously applied in several other projects and gave good results
- A basic version of the method can be implemented within the given time frame

Probabilistic inversion can be applied to inverting the gas dispersion process as follows.

The survey area is divided into a grid of n cells, each containing an emission source at its centre. Then, randomly generate N candidate source patterns for the disposition of the n sources $S_j = (s1, s2 \ldots sn)_j$, j = 1..N. N should be as large a number as is practical. The resulting Nxn individual source strengths must lie between 0 and a defined value MaxEmis. The distribution of the number of sources versus source strength is governed by a Beta distribution – described below. The sources are assumed to be independent of each other and time invariant.

A forward gas dispersion model is then applied to each of these N candidate source patterns, resulting in M (the number of actual gas concentration measurements) values of predicted gas concentration for each of the N patterns. This can be presented as an MxN matrix of predicted gas concentrations Y, which is defined in general case as:

$$Y_{ij} = F_i (S_j) = F_i ((s1, s2 \ldots sn)_j), i = 1..M, j = 1..N \quad (4)$$

where $F_i(x1,\ldots,xn)$ – is the result of applying the forward model for the i-th measurement. In the current stage of the project the forward model is a simple Gaussian Plume model, defined by (1) – (3), which incorporates the combined effects of measurement location, windspeed and direction, standard deviations of vertical and

horizontal wind components and gas density. This Gaussian Plume model is linear with respect to the source vector. As in (3), let us denote the Mxn Gaussian Plume coefficient matrix as A, its j-th row relates a source vector with the concentrations resulting for j-th measurement. The concentrations by volume C lie between 0 and 1.

The following data are available a priori:

A – Mxn matrix of the Gaussian Plume coefficients, computed with the LabView programme, written by the Optics Group; this incorporates the effects of area sources, measurement and source heights, different averaging times, background concentration offsets, and other physically significant factors.

C – Mx1 vector of the actual concentrations (by volume) measured.

MaxEmis – the maximum source strength, in [kg/(s)].

Sigma –Mx1 vector of standard deviations of each concentration measurement (assessed by experts as a combination of measurement sensitivity and the statistics of the measured concentrations at each location).

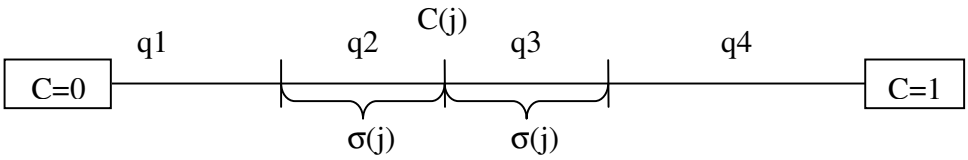
Probabilistic Inversion Algorithm

The probabilistic inversion method comprises the following steps:

1. Generate N candidate source patterns $S_j = (s1, s2 \ldots sn)_j$, j = 1..N. This will result in an Nxn matrix S.
2. Compute the predicted concentrations $Y = A*S^T$
3. Based on the values Y, each of the N source patterns S are then weighted such that the resulting modified distribution of Y's agrees with a specified distribution.

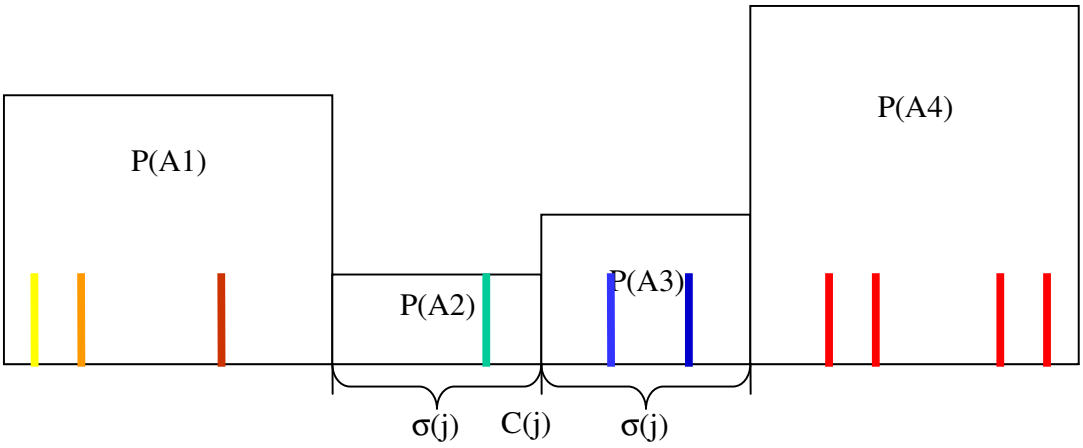
For the LightTouch application the following considerations are sensible:

- It's reasonable to use Sigma, the standard deviation of concentration measurements given by experts, to specify the following distribution of the N predicted concentrations Y_i, i=1 to N, in relation to the actual concentration measurement C(j).

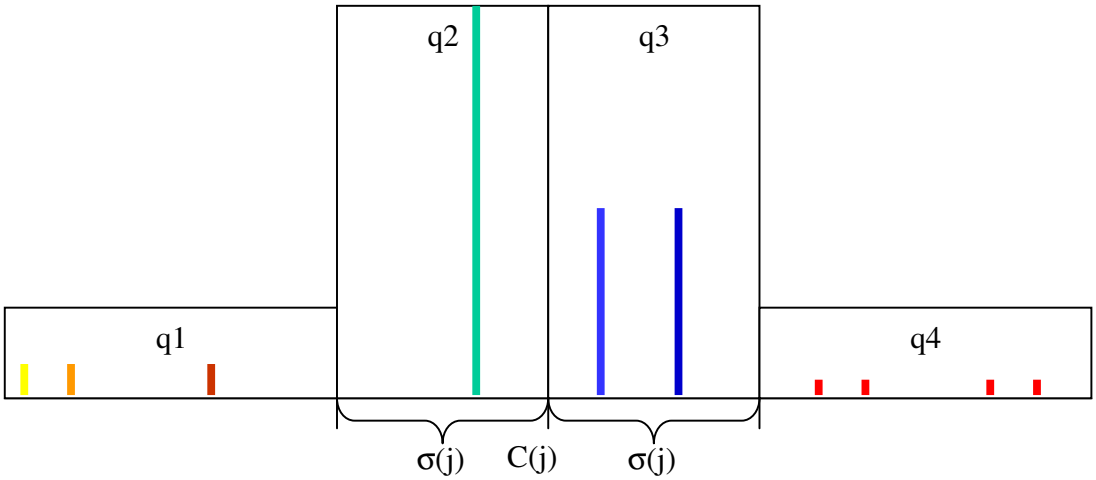


- The quantiles q1, q2, q3 and q4 are specified in advance (all between 0 and 1, summing to 1; where these quantiles are the relative frequencies of predicted concentrations lying in the specified interval). In other words, we require that the relative frequency of the reweighted predicted concentrations falling into the j-th interval should be equal to q(j). That is, the number of reweighted predicted concentrations which are smaller than Cj – Sig(j), divided by N, should be close to q1; for those between Cj – Sig(j) and Cj – close to q2; etc. The algorithm reweights the N initial source patterns such that the predicted concentrations correspond to the distribution defined above for all M concentration measurements.

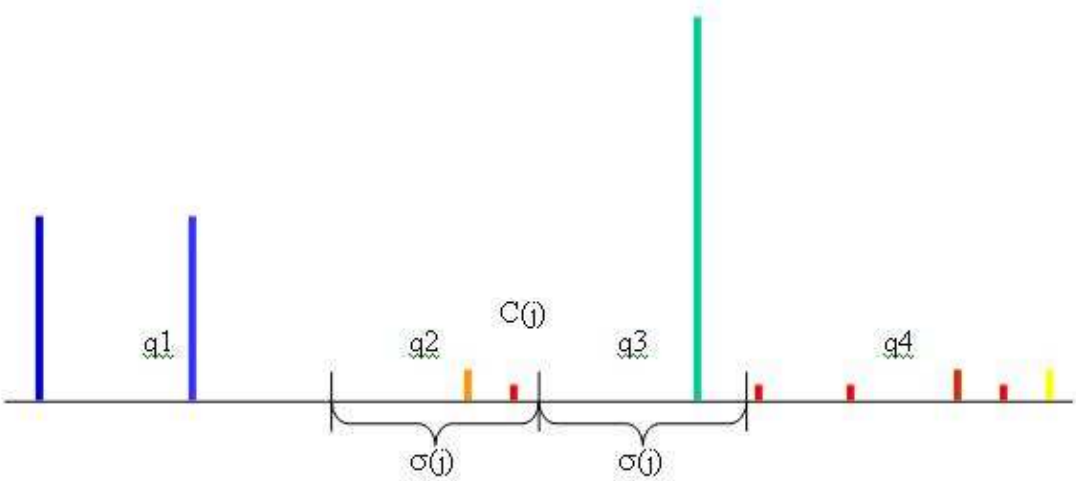
For example, if 10 patterns (initially having equal weights) are distributed around the first measured concentration in the following way:



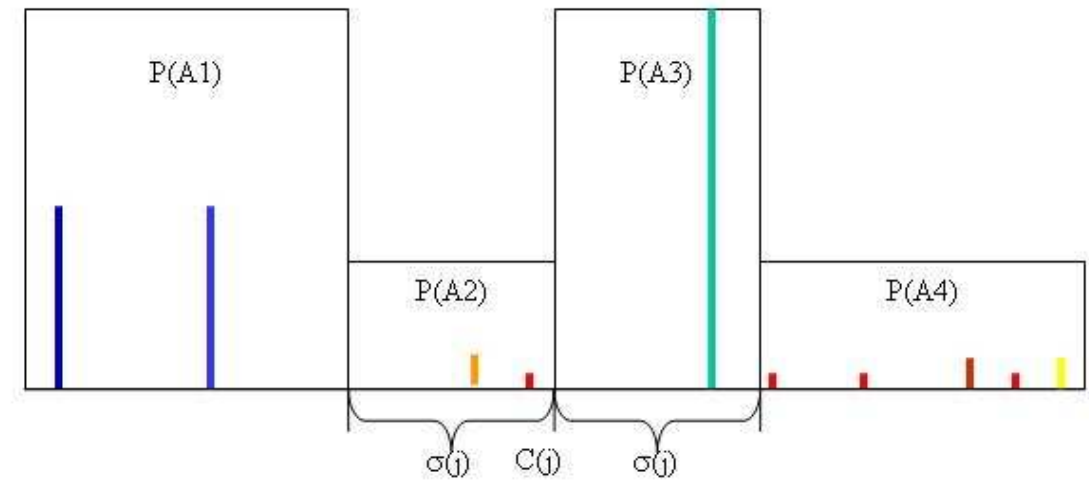
After re-weighting it will look like:



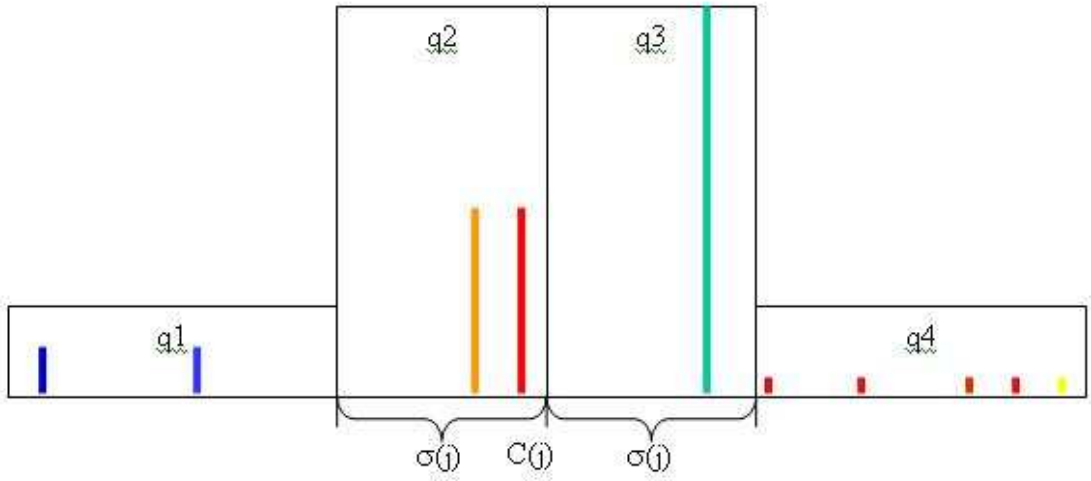
Now let's plot the predicted concentrations of those same 10 patterns for the second measurement with the weights determined above:



So, the distribution for the second measurement becomes:



And after re-weighting gives required distribution for the second measurement.



Of course, taking those weights for the first measurement again will disfigure the required distribution. One can immediately suggest two iterative approaches: consecutive fitting through all the measurements (Iterative Proportional Fitting, IPF)

or averaging weights obtained from all the measurements (Parametric Fitting for Uncertain Models, PARFUM). These two methods will be formally described later.

- The individual source strengths should be generated on [0..MaxEmis] according to some distribution, such that:
    - There is at least one predicted concentration within each interval (corresponding to q1, …, q4) for each actual concentration measurement. Let A(i,j) stand for the sub-set of N source patterns for which Y_i falls into inter-quantile interval j. (This A(i,j) is not to be confused with the Gaussian Plume matrix which will not be referred to further.)
    - The N candidate source patterns can be generated in numerous ways. Better sampling procedures will be those resulting in more values of predicted concentrations within inter-quantile intervals 2 and 3 around C(j) (corresponding to q2 and q3) for each concentration measurement.

IPF and PARFUM

Two approaches to implementing step 3 of the Algorithm are:

- Iterative Proportional Fitting (IPF)
- Parameter Fitting for Uncertain Models (PARFUM).

The IPF algorithm adapts an initially uniform probability vector p to all partitions iteratively. The iterative update of current vector p is

$p' = (\ldots(p^{A1})^{A2}\ldots)^{A(M)}$,

where $p_i^A = \sum_{j=1}^{4} I_{\{i \in A_j\}} \frac{p_i q_j}{p(A_j)} = \frac{p_i q_{j(i)}}{p(A_{j(i)})}$

is the adaptation of p to the partition $\{A_j\}$, j=1..4; $q_{j(i)}$ is defined by context.

The PARFUM algorithm, on the contrary, adapts a starting measure to each partition and then averages these adaptations to obtain the next iterate:

$$p' = \frac{1}{M}\sum_{m=1}^{M} p^{Am}$$

$$p_i' = p_i\left(\frac{1}{M}\right)\sum_{m=1}^{M}\frac{q_{k(i)}}{p(A^m_{k(i)})}$$

It can be shown (see [1]) that the solution (a stable point) P* that PARFUM converges upon for the measurement constraints given, is also the solution for the minimum information problem, given those same constraints. More precisely, P* minimizes the relative information:

$\sum_{i=1}^{N} I(P_i, P)$ where:

$$I(P_i, P) = \sum_X P_i(X)\ln\frac{P_i(X)}{P(X)},$$

Subject to additional constraints imposed by measurements.

A stable point for the IPF approach is related to a modified minimum information problem. While it is known that the PARFUM algorithm always converges, IPF may not (for example when the corresponding minimum information problem is non-feasible). Computer experiments show that if IPF does not converge, it oscillates. For more links on theoretical results refer to [1].

After the algorithm has stabilized its evaluation of p=(p1, p2,…pN) to within a given tolerance limit (the Tolerance), one has the probabilistic distribution for each source. With this distribution one can compute the corresponding final Source pattern S_f as the weighted average of S_j's:

$$S_f = \sum_{i=1}^{N} p_i S_i ,$$

the variance and the standard deviation of each of the n individual sources that comprise S_f

$$Var\, S_f = E\left(S^2\right) - \left(ES\right)^2 = \left(\sum_{i=1}^{N} p_i S_i^2\right) - S_f^2$$

Std S_f = √(Var S_f)

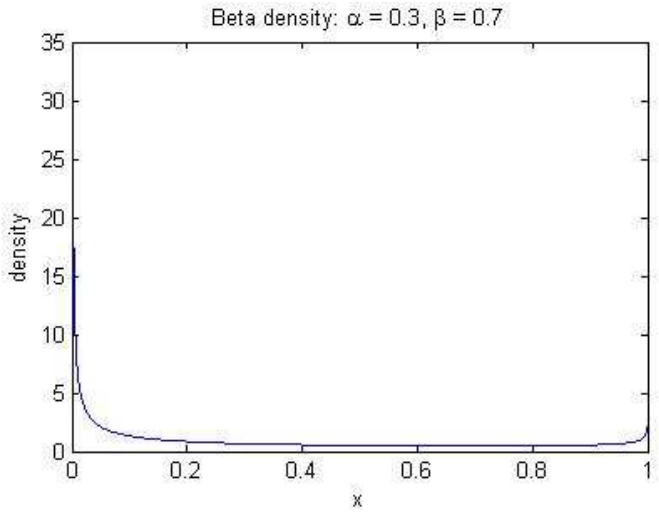
along with 5%- and 95%- percentiles, which can serve as confidence bounds for each of the n individual sources that comprise the answer S_f.

Implementation.

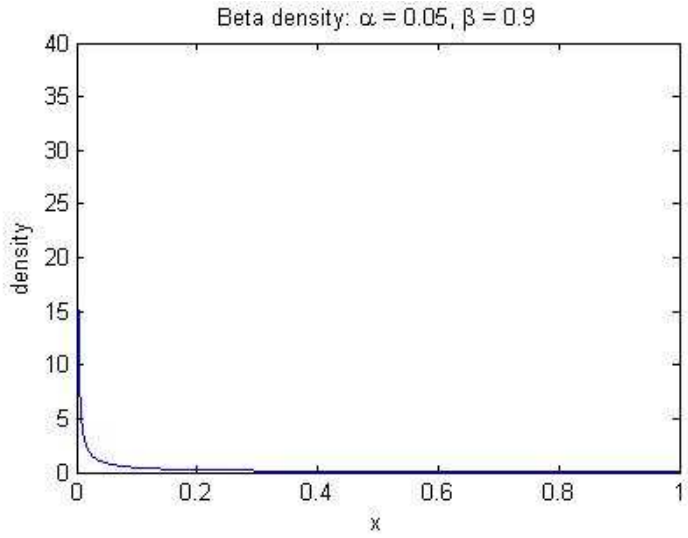
Both algorithms were implemented in Matlab and tested for a Cylinder release dataset in which we have M = 80 measurements, n = 256 (16*16) grid cells and the value of MaxEmis = 5 kg/hr. The quantiles were chosen to be q = [0.05 0.45 0.45 0.05]. The sources were assumed to be independent. First, they were generated uniformly between 0 and MaxEmis, but this did not give a good result: for some measurements there were quantile intervals without any predicted concentrations. Much better results were obtained using a Beta distribution. In general the probability density of a Beta distribution with parameters (α, β) is given by:

$$\frac{(1-x)^{\beta-1}x^{\alpha-1}}{B(\alpha,\beta)}$$

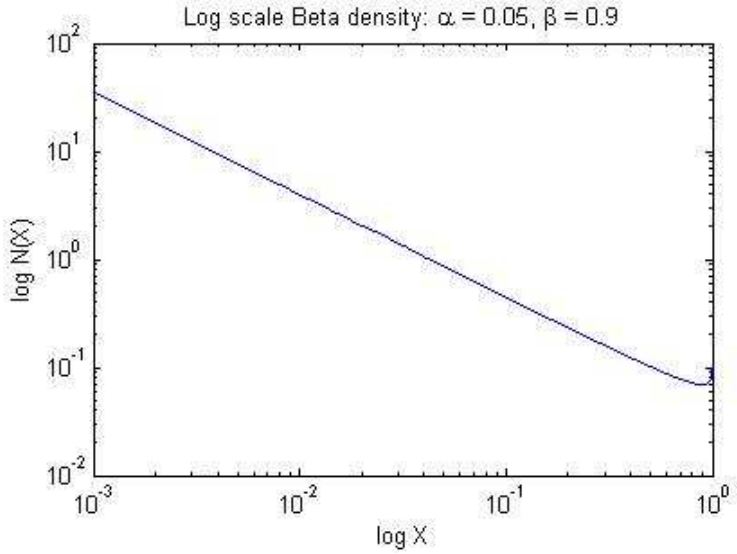
Here is a typical shape of the Beta distribution:



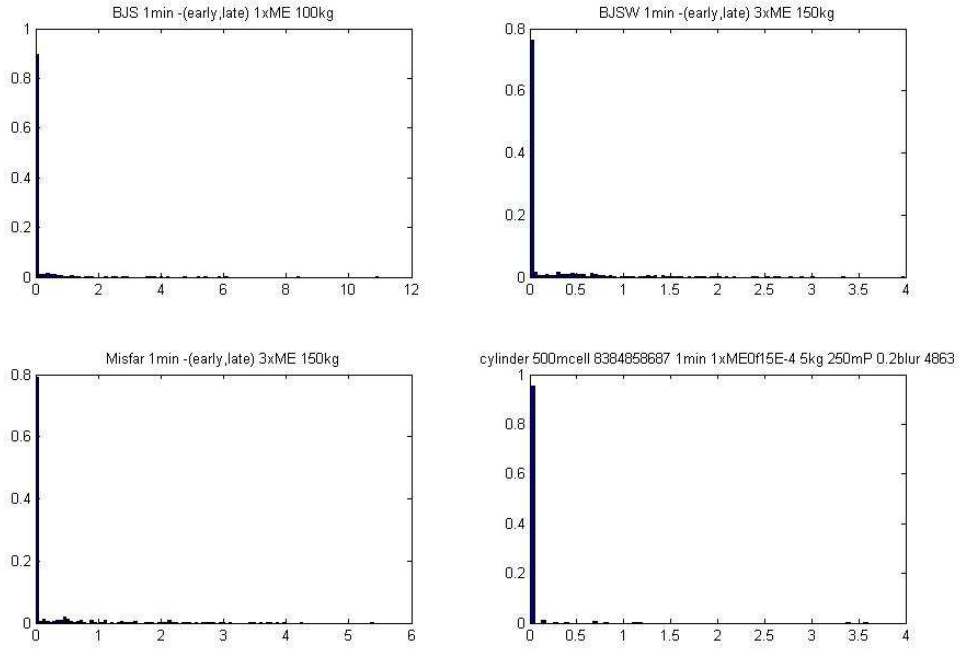
Experiments revealed that a suitable Beta distribution was given by $\alpha = 0.05$ and $\beta = 0.9$; this yields the probability distribution shown below:



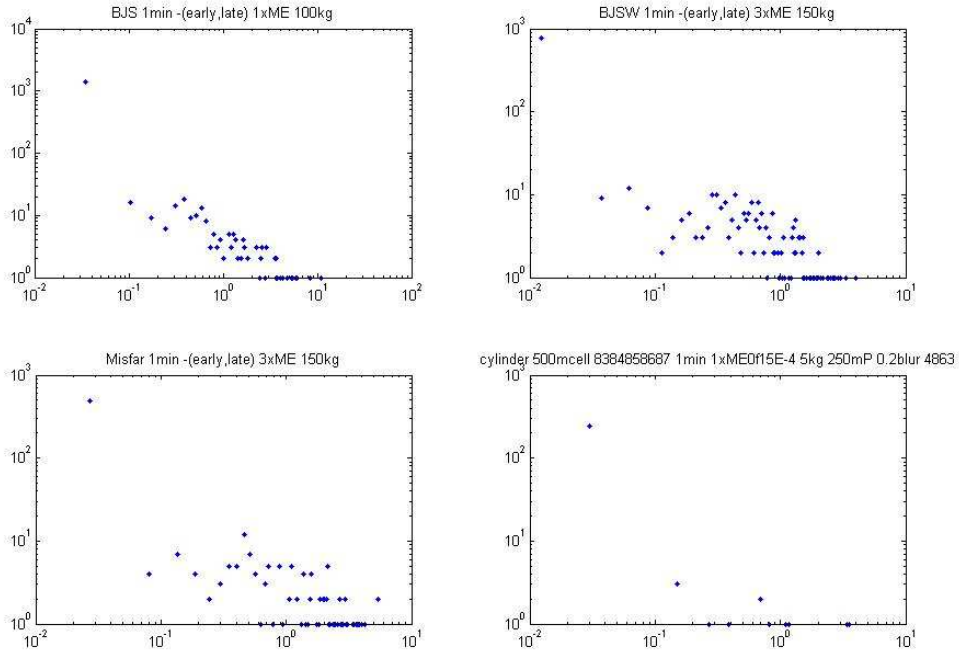
As the significant emission sources occur with relatively low probability densities, it is more informative to examine a plot of LogN(X) versus Log(X).



We then examined the results from several real surveys, analysed using AvRecon, to see how the source distributions compared to our empirically chosen Beta distribution – which had been selected as appropriate for the cylinder release data:



The same data sets plotted in log-log scale:



As the source strengths are directly dependent on migration pathways determined by fracturing of rock, one might suspect that the sources would have a distribution

comparable to that characterising the fracture network. These are often described using the concepts of fractals and power laws; the most famous of which is the Gutenberg-Richter Law. This states that the frequency of earthquakes of magnitude M or greater, N(M) is given by

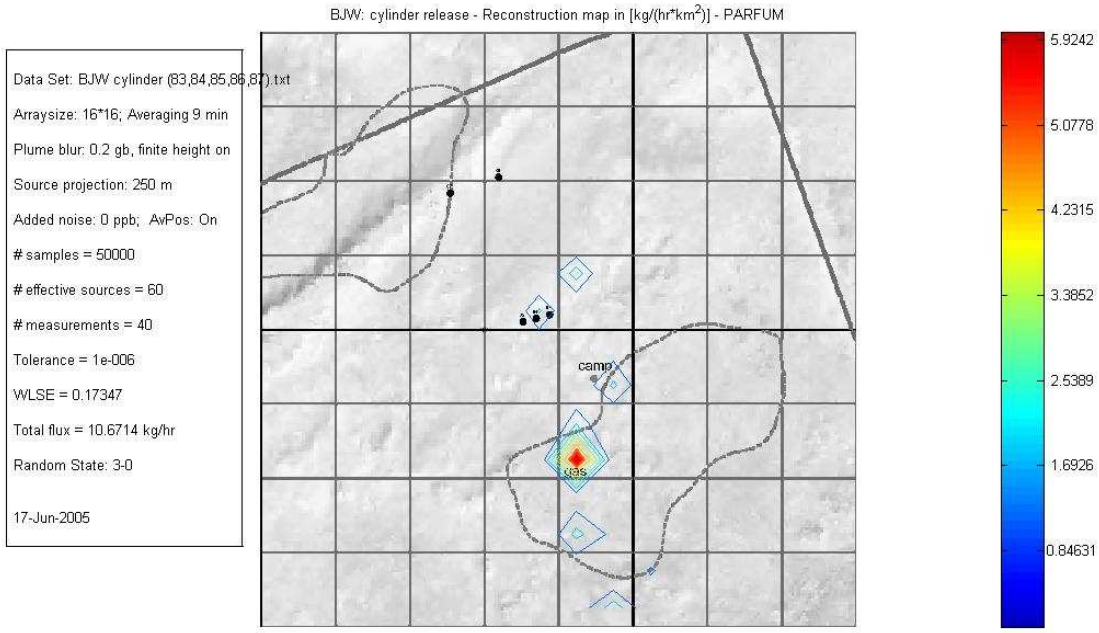
$\log N(M) = a - bM$

for some constants a, b.

In Matlab the sampler of Beta-distributed random variables is determined through the uniform and normal generators. In particular, if we fix the initial random seed for those two generators, the generated Beta-paths will be uniquely determined. For that reason, the initial random state is fixed in the program by two integers representing uniform and normal random states. Thus by changing the seed values we are just taking a different random sequence, there is no other significance to the actual seed numbers used.

Testing

The IPF and PARFUM algorithms were implemented in Matlab and tested on the cylinder release dataset with 9 min. averaging (thus, M = 40 measurements), when they both converge. The PARFUM method resulted in the following stable fir for N = 50000 samples (the reconstruction below was performed on 16x16 grid at 8x8 km area of Bahja west survey):

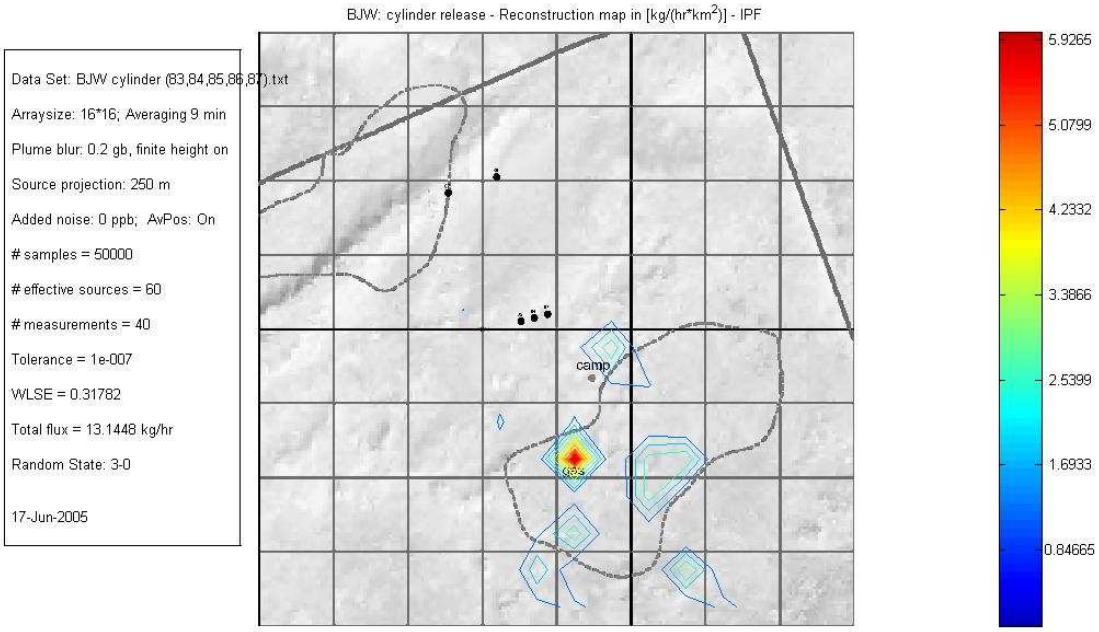


With the following distribution for each measurement:

Meas-t	P(q1)	P(q2)	P(q3)	P(q4)
1	0.04504	0.35166	0.51210	0.09120
2	0.05509	0.44051	0.42708	0.07733
3	0.05370	0.42407	0.44615	0.07608
4	0.06254	0.39175	0.45837	0.08734
5	0.05516	0.42845	0.45547	0.06092
6	0.05446	0.48303	0.40905	0.05346
7	0.07427	0.49715	0.37666	0.05191
8	0.05276	0.45757	0.43681	0.05286
9	0.30221	0.46983	0.19712	0.03083
10	0.18875	0.56505	0.22290	0.02330
11	0.11151	0.61166	0.25192	0.02491
12	0.05203	0.41064	0.48255	0.05478
13	0.04749	0.37973	0.49783	0.07495
14	0.03954	0.37177	0.51155	0.07713
15	0.04047	0.46847	0.44135	0.04971
16	0.04294	0.48636	0.42106	0.04964
17	0.03752	0.68863	0.24014	0.03371

18	0.03610	0.52808	0.38748	0.04834
19	0.04061	0.30030	0.37760	0.28150
20	0.04234	0.45016	0.41340	0.09411
21	0.04935	0.44136	0.39714	0.11214
22	0.05933	0.48680	0.36285	0.09102
23	0.07543	0.37622	0.44881	0.09954
24	0.03934	0.28783	0.44205	0.23078
25	0.24859	0.45802	0.26159	0.03180
26	0.39553	0.39510	0.18725	0.02212
27	0.19424	0.41785	0.34611	0.04180
28	0.18939	0.40141	0.27628	0.13292
29	0.08582	0.40548	0.37574	0.13296
30	0.03928	0.34780	0.38800	0.22492
31	0.02398	0.20892	0.42031	0.34680
32	0.01899	0.16713	0.36584	0.44804
33	0.02692	0.29354	0.60093	0.07861
34	0.03867	0.46414	0.44077	0.05642
35	0.05829	0.39361	0.39981	0.14828
36	0.07296	0.39928	0.38649	0.14128
37	0.07775	0.50597	0.36932	0.04696
38	0.06080	0.44319	0.46243	0.03358
39	0.07765	0.48274	0.38870	0.05090
40	0.08904	0.51154	0.35274	0.04667

The IPF results in a similar fit with the same parameters:



Meas-t	P(q1)	P(q2)	P(q3)	P(q4)
1	0.07400	0.42982	0.42581	0.07037
2	0.08709	0.30425	0.50712	0.10155
3	0.10013	0.29247	0.50585	0.10155
4	0.10634	0.25712	0.50188	0.13466
5	0.10607	0.34342	0.44897	0.10155
6	0.10198	0.35196	0.46350	0.08257
7	0.12876	0.49229	0.31596	0.06299
8	0.10019	0.51515	0.32167	0.06299
9	0.32362	0.55712	0.11210	0.00716
10	0.24573	0.68626	0.06085	0.00716

11	0.23801	0.69398	0.06085	0.00716
12	0.15639	0.73528	0.10115	0.00718
13	0.13722	0.76698	0.08745	0.00835
14	0.13039	0.69929	0.16197	0.00835
15	0.12332	0.78531	0.08421	0.00716
16	0.13039	0.78083	0.08161	0.00716
17	0.10637	0.82336	0.06053	0.00975
18	0.09713	0.77724	0.11588	0.00975
19	0.10637	0.50371	0.31965	0.07027
20	0.08141	0.55038	0.29445	0.07375
21	0.09031	0.52046	0.31496	0.07427
22	0.09581	0.51496	0.31496	0.07427
23	0.07727	0.49261	0.32953	0.10060
24	0.08141	0.45263	0.38638	0.07957
25	0.62132	0.34857	0.02982	0.00029
26	0.66095	0.32556	0.01323	0.00026
27	0.67680	0.29748	0.02543	0.00029
28	0.66866	0.30490	0.02565	0.00080
29	0.36324	0.57697	0.05729	0.00250
30	0.14197	0.51960	0.32373	0.01470
31	0.05916	0.59394	0.31679	0.03011
32	0.05916	0.55147	0.32885	0.06052
33	0.03057	0.28767	0.65206	0.02970
34	0.04202	0.30610	0.62218	0.02970
35	0.03841	0.28263	0.62176	0.05720
36	0.03841	0.31710	0.60723	0.03725
37	0.04600	0.38493	0.53937	0.02970
38	0.04030	0.37819	0.55181	0.02970
39	0.05348	0.48393	0.43290	0.02970
40	0.05000	0.45000	0.45000	0.05000

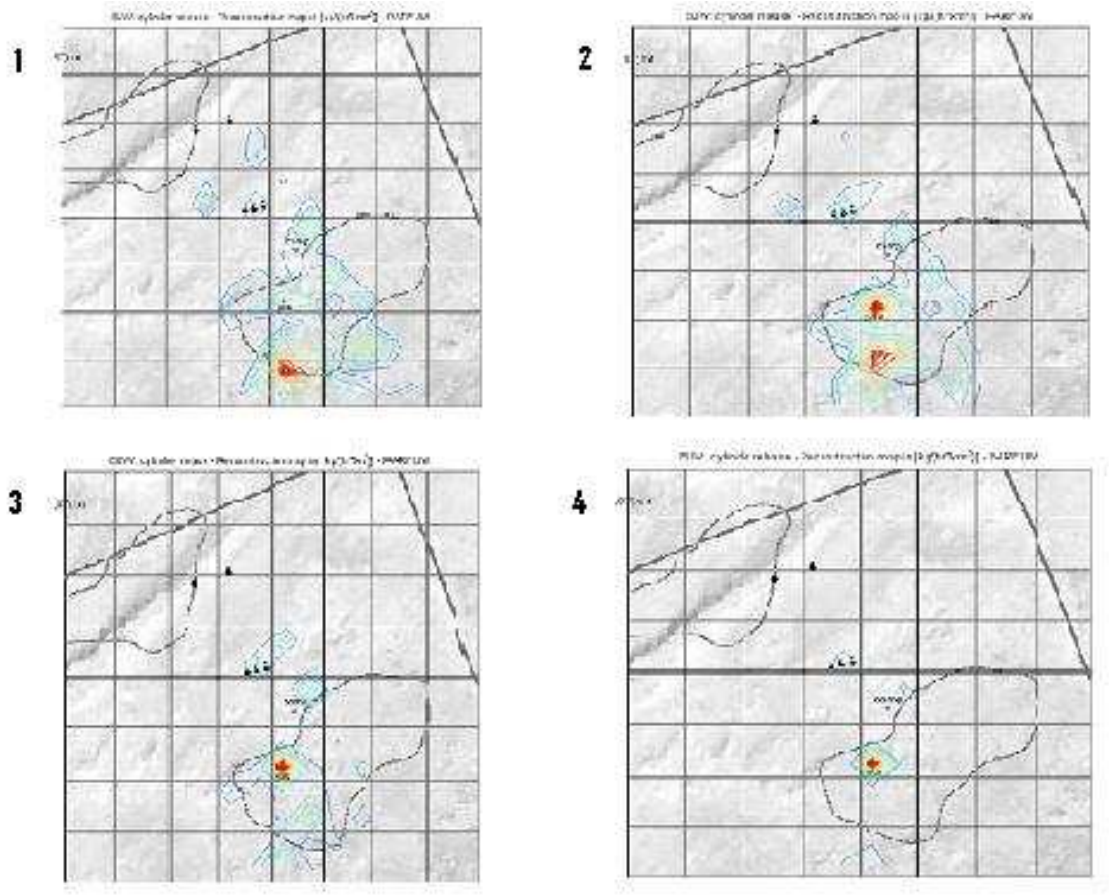
Modified methods

Genetic approach:

The 2 Probabilistic Inversion methods described above have been studied theoretically extensively and applied in several projects. The advantage of the PARFUM algorithm is that it always converges, whereas IPF may not. In any case, both of them require a large number of patterns, which is not always feasible within given memory constraints. The following idea was employed in order to reduce the number of samples needed for a reasonable fit.

After we run PARFUM with a relatively small number of patterns, we can take the most significant patterns and, based on them, generate other patterns and make another run. Implementing this idea, after each run the most significant patterns, whose total weight exceeded 0.9, remained, the others were discarded. Quite a few techniques were used for generating new patterns out of those remaining. The best performing approach proved to be a cell-wise random selection among the remaining patterns, that is when each cell of a new pattern is a (uniform) random selection of the same cell among

remaining patterns. To illustrate the improvement, here is the successive reconstruction maps for N = 2000. Note, that the first run does not give a reasonable reconstruction. However after a few runs the method converges to a stable answer.



Modified distribution - ALPINE

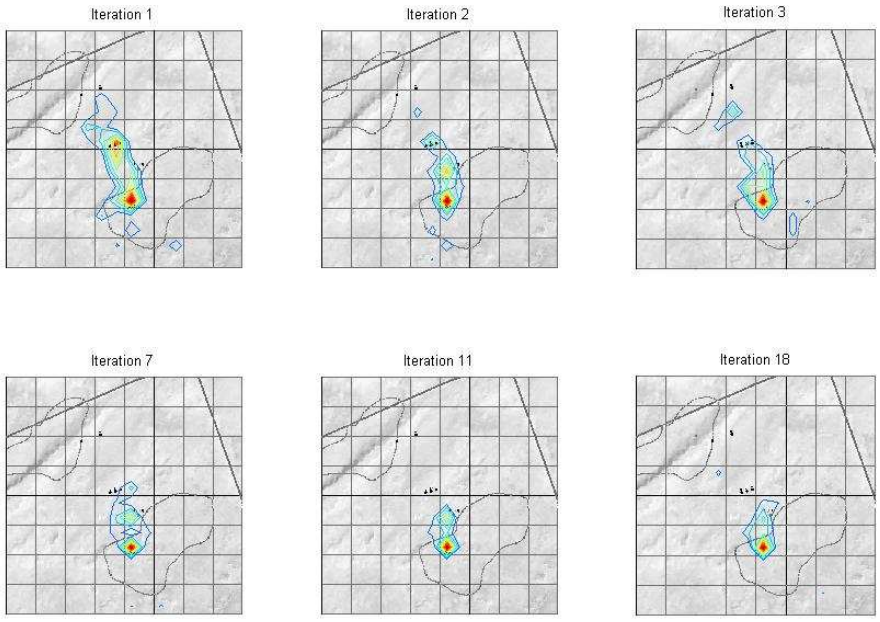
The previous modification permits computation with fewer patterns (and therefore reduces memory requirements – or increases range of patterns that can be employed for a fixed memory specification). Nevertheless it still requires multiple runs of the whole algorithm until a stable answer is obtained. For that reason the following modification was developed. Instead of running the whole algorithm until the probability vector stabilizes, we can just perform the first iteration and retain the most significant patterns

only. Further, for each iteration there is no need to discretize the distribution function. Instead, we can assign weights to each pattern based of the difference between the predicted and measured concentrations. The best result was achieved using weights proportional to the inverse square distance (using notations (2)-(3)):

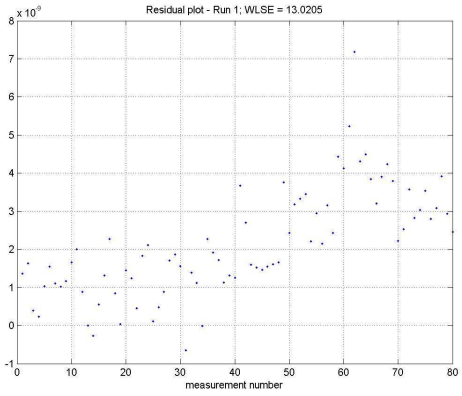
$$p_i \propto \left\langle \frac{1}{(Y_{ij} - C_j)^2} \right\rangle_j$$

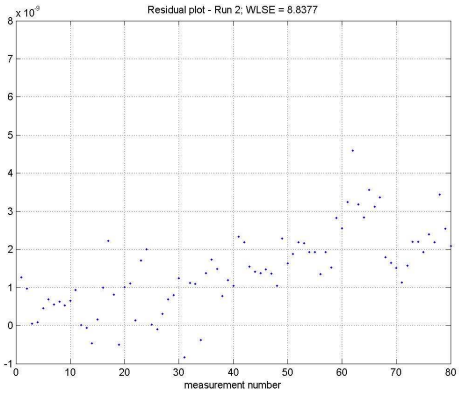
$$i = 1 \ldots N, j = 1 \ldots m.$$

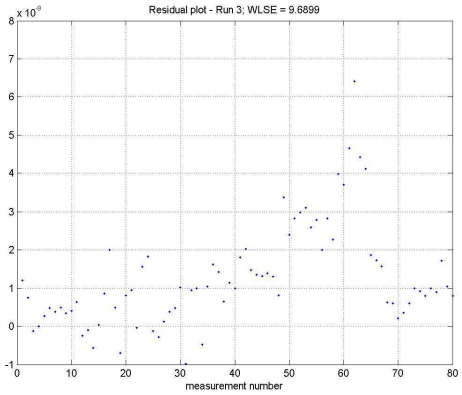
Here is the iterative reconstruction map for the method with N = 50000 patterns:

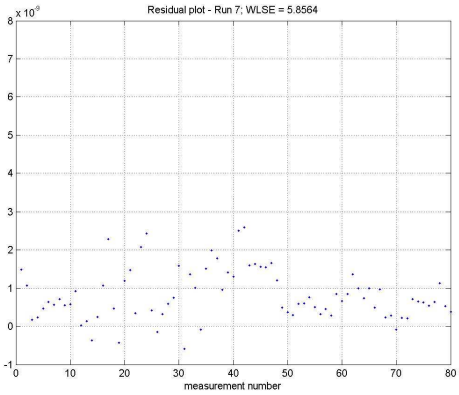


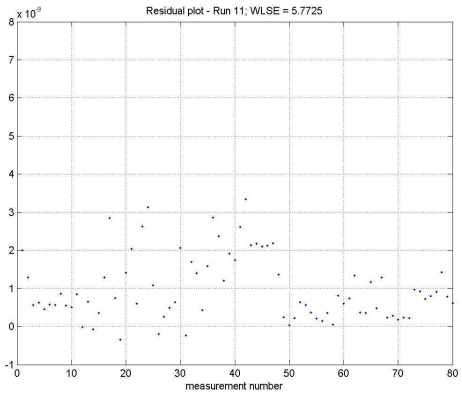
with the following concentration residuals structure:

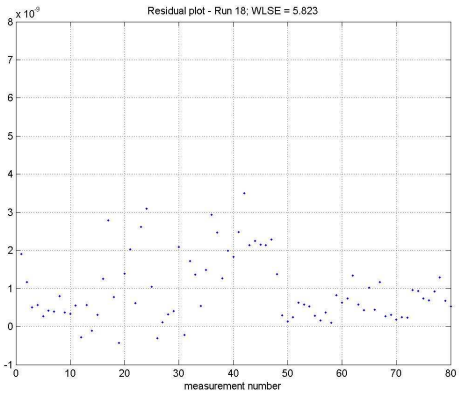












Note, that one step of the modified algorithm corresponds to just one iteration of PARFUM (not the whole run!) which reduces the computational effort drastically. This modified method, combining genetic approach, modified distribution and relaxation of the classical (PARFUM) scheme, will further be referred to as ALPINE. The modifications mentioned provide the following advantages:

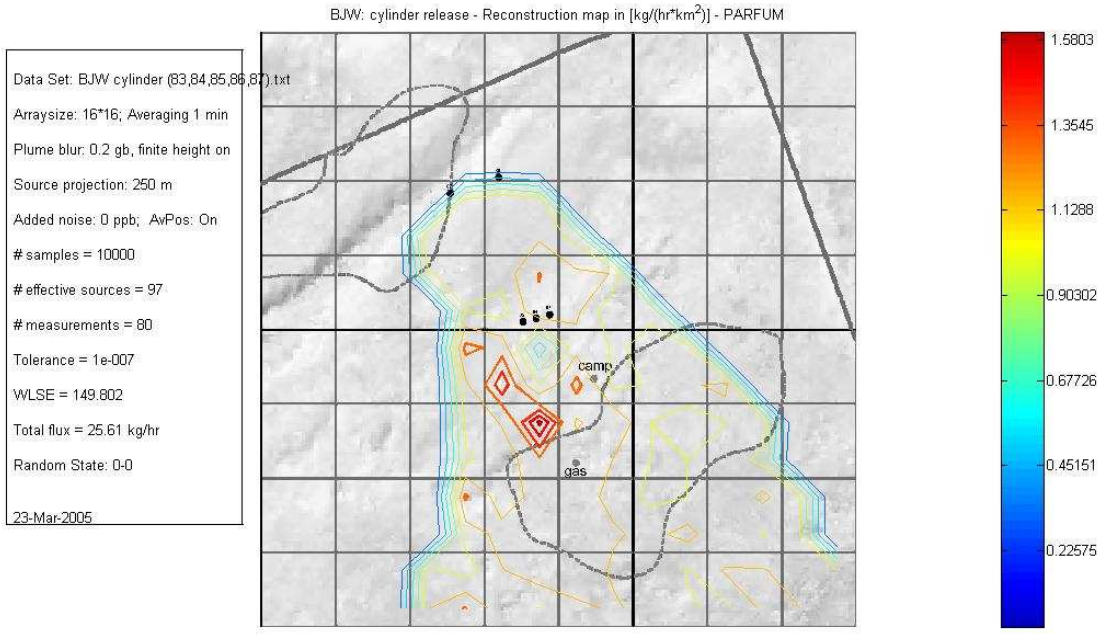
- Faster convergence (performing 1 iteration instead of the whole run)

- With the modified distribution the need to avoid "empty inter-quantile intervals" is removed. Therefore, a broader range of source strength distributions can be tested.
- More efficient use of memory. In particular, a reasonable reconstruction with ALPINE requires fewer patterns than a similar one with PARFUM.

Results

IPF vs PARFUM

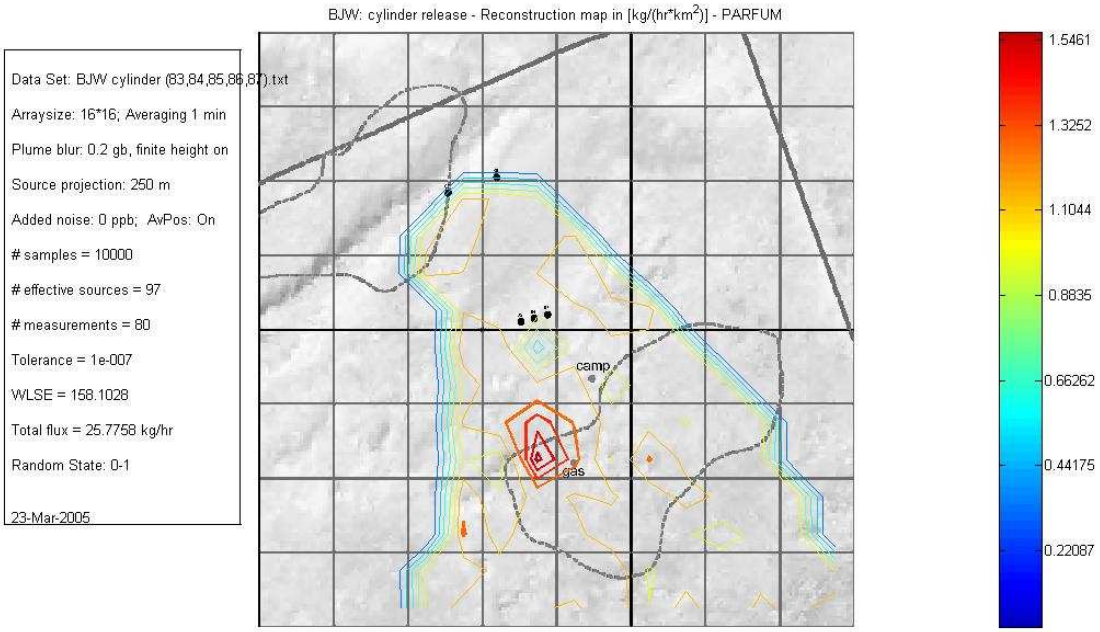
If we take all available measurements (80 one-minute measurements) of the cylinder release dataset, the IPF algorithm does not converge (it oscillates). Whereas PARFUM results in the following reconstruction:



Meas-t	P(q1)	P(q2)	P(q3)	P(q4)
1	0.31573	0.04728	0.04769	0.58930
2	0.23900	0.05988	0.06572	0.63541
3	0.51561	0.04466	0.04573	0.39400
4	0.60457	0.05379	0.04881	0.29284
5	0.48417	0.05779	0.06148	0.39656
...				
75	0.11416	0.04794	0.04623	0.79167
76	0.26517	0.04828	0.04578	0.64077
77	0.18759	0.04831	0.04703	0.71707

78	0.09459	0.04774	0.04631	0.81136
79	0.25618	0.04777	0.04574	0.65031
80	0.31548	0.04855	0.04647	0.58950

But if we change the initial random state to 0-1, we get the following reconstruction:

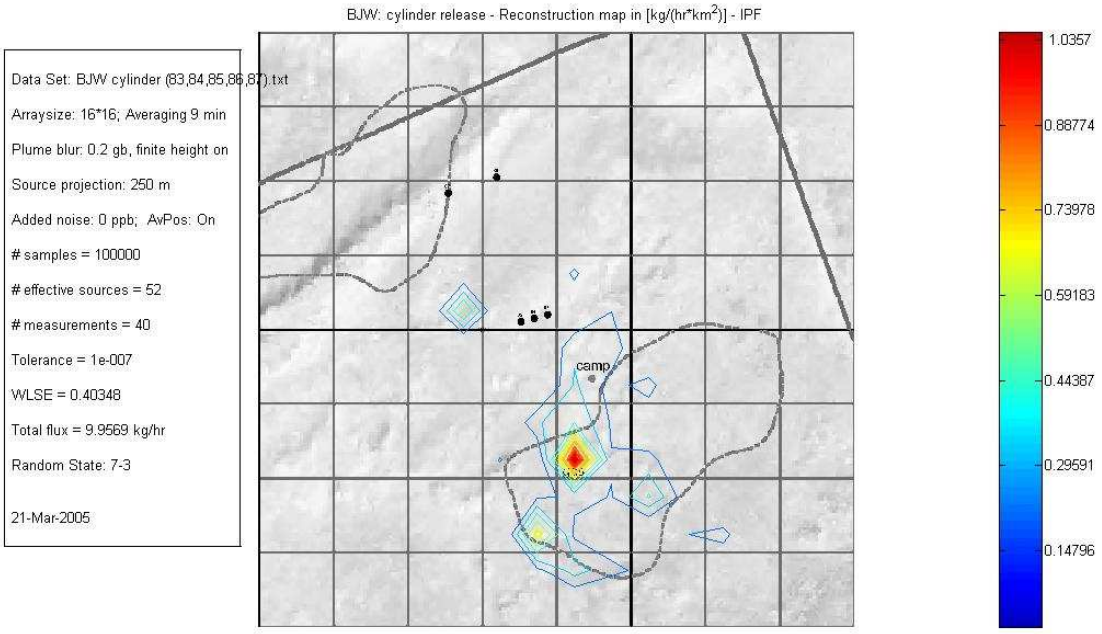


The difference in results obtained with differing seed values indicates the result has not yet fully converged.

However, clearly what is of more interest is to examine the results obtained using all the measurement data available. It is likely that IPF was prevented from converging when used with all 80 of the 1-minute measurements, by the degree of conflict within the data. Consequently we used the front-end averaging facility within AvRecon to provide data averaged over longer times: hoping to reduce the degree of data conflict. Averaging over 9 minutes provided 5 measurement positions with a total of 40 concentration measurements, and for these data IPF did converge to provide the following fit. (Note; N here is 100,000 and the Tolerance 1E-07)

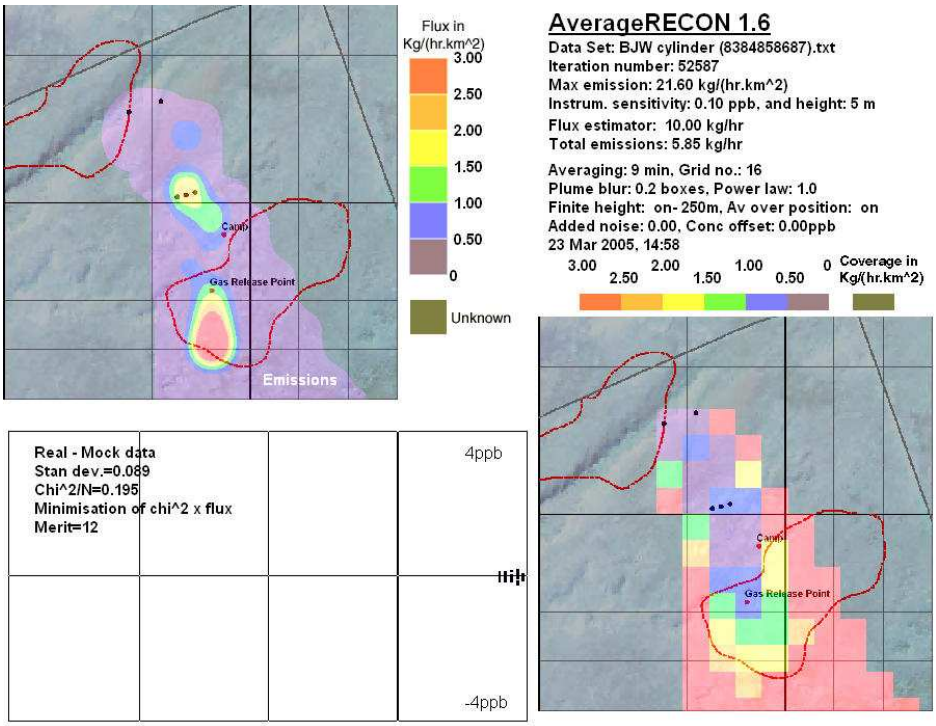
(Here we see the value of a fully objective measure of confidence – the degree of improvement delivered by eradicating systematic errors or shortcomings in the forward

modelling can be assessed. Other suspected shortcomings in the forward model can similarly be tested and improved on the basis of the algorithms confidence measure.)



The above is a perfect result – to within the limitations inherent in the experiment (presence of atmospheric background concentration, and possible minor additional real emission sources as seen in the full survey of this area.)

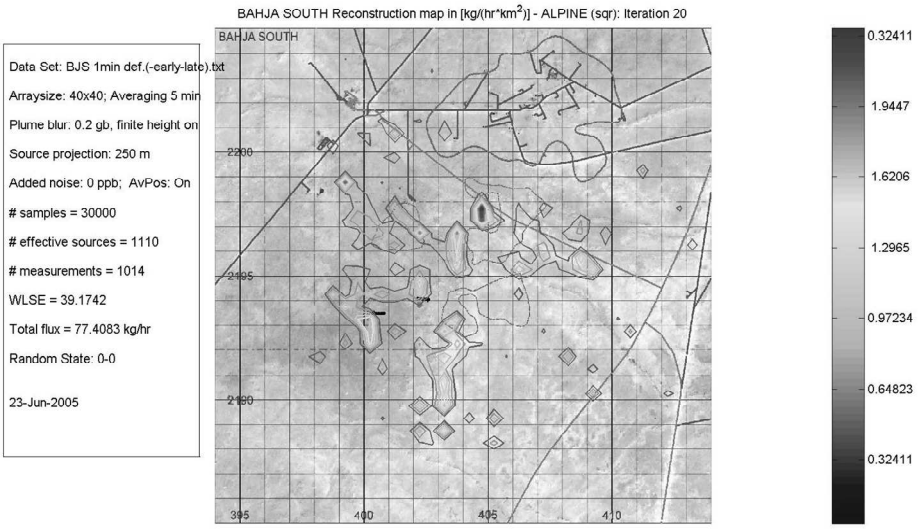
The AvRecon result with the same data and the same parameters:



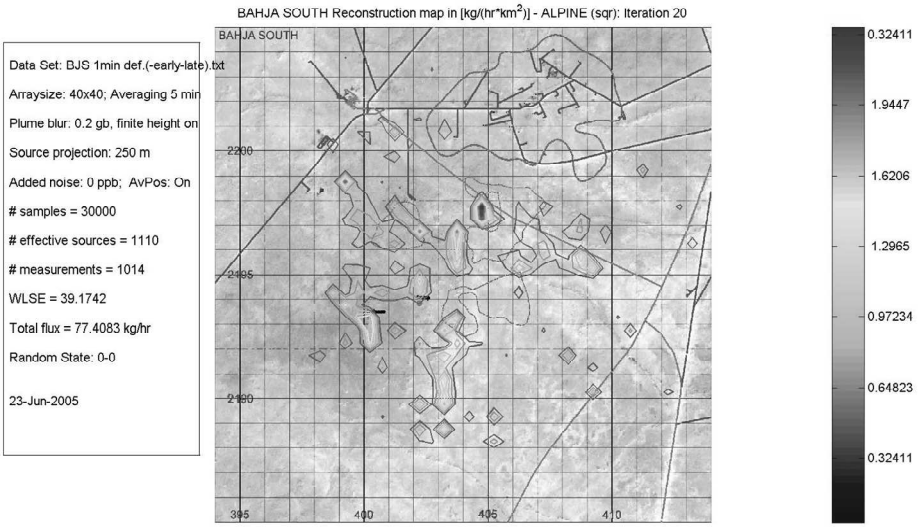
Bahja South Survey: real survey full data

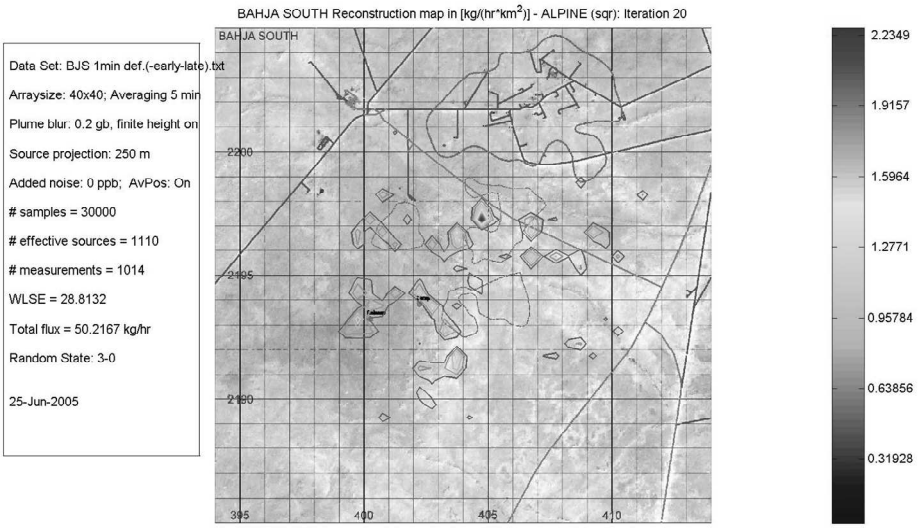
As seen above, the modified algorithm (ALPINE) gives a very good result for the cylinder release data set. The methods were then tested on the real survey data set of Bahja South with 1440 1-minute measurements averaged over 5 minutes and 1600 (40x40) grid cells.

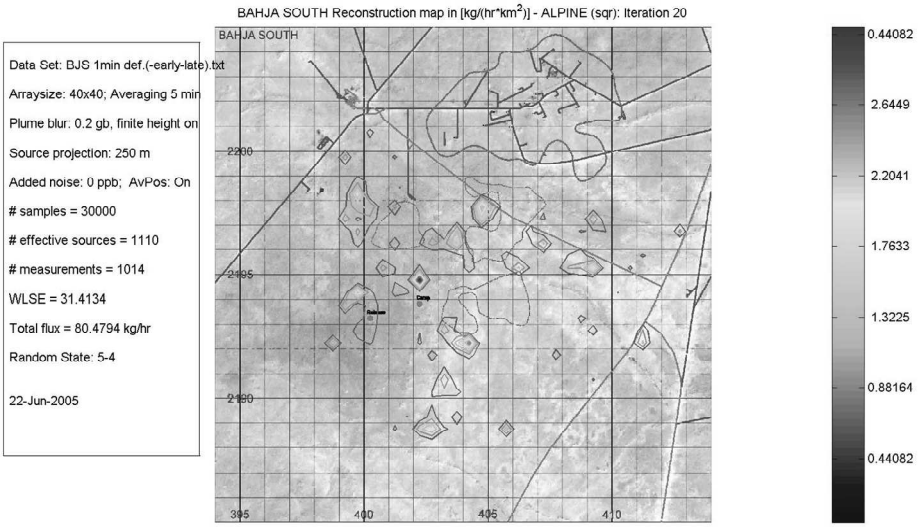
The IPF and PARFUM could not cope with high dimensionality of the full data set from Bahja South. Because of a large number of measurements, and fine grid size, the two methods could not find a stable fit within given memory constraints. However the modified method, ALPINE, combining the genetic approach with the modified re-weighting distribution, does give a stable result for the whole data set.



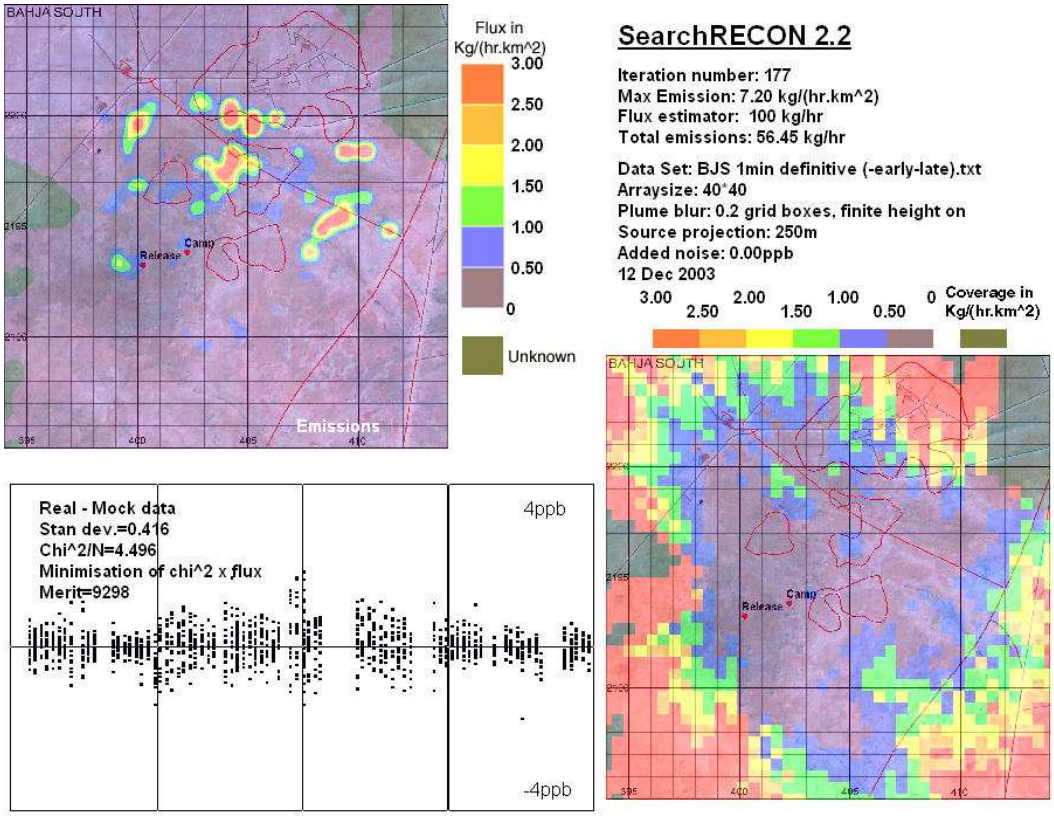
Below are shown three reconstructions with different initial random seeds for the same data set.







The results are clearly correlated, and exhibit features present in the result obtained by SearchRecon with the same parameters:

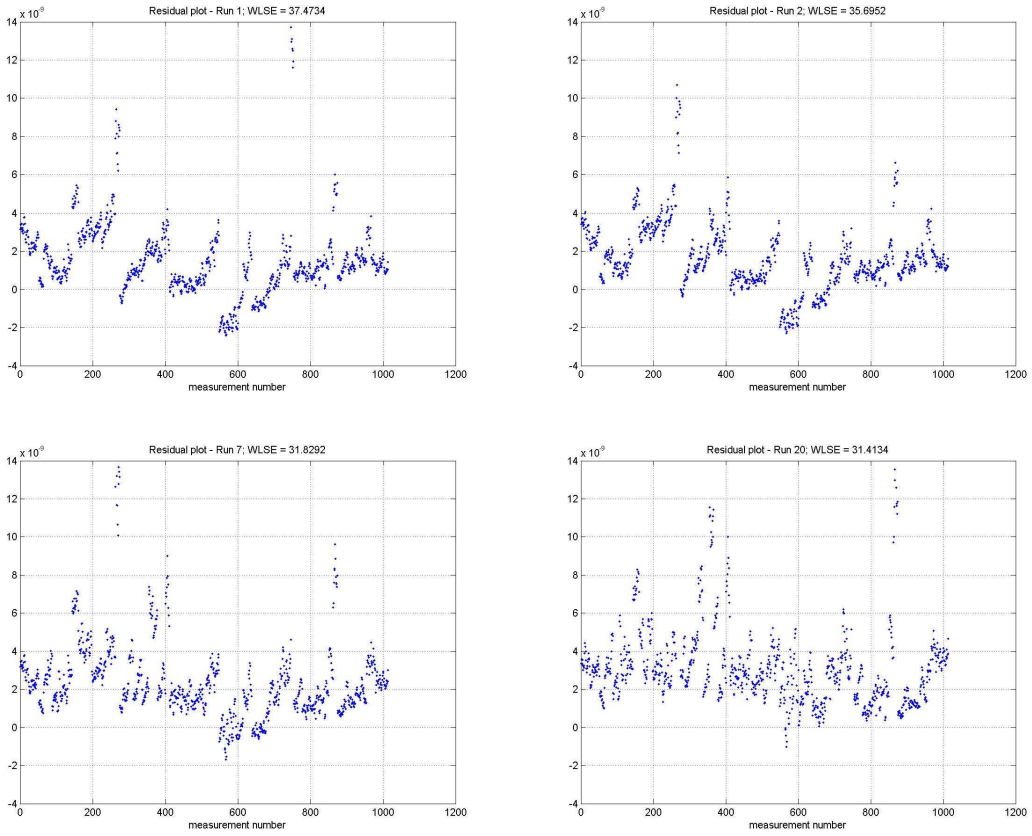


Unfortunately we do not know the "true answer" to the full data set problem, and can only try to compare the results of running different methods. Although Shell will be evaluating this new method against data sets for which a degree of "ground-truthing" is available via a separate geochemical method run in tandem with LightTouch in blind trials. For now, we have an option to run ALPINE with different random seeds and see how correlated the results are. A close correlation between the results for different random seeds would be an indicator of a stable result, due to the data rather than being constrained by the particular set of patterns generated for that run. Similarly the absence of any structure within the residuals would suggest full convergence.

Another feature of probabilistic inversion methods worth mentioning is that they do not seek to minimize the total square error. Any specified distribution for penalizing residuals always permits some deviations from the concentrations measured. Therefore the total square error can increase at a later iteration, and can only serve as a secondary "goodness of fit" measure. In particular, the probabilistic inversion methods did not reconstruct a "self-fulfilling" data set (a data set resulting from a known source pattern,

via the given forward model), the solution of which would be the “perfect” least squares fit.

The residuals behaviour resulted from ALPINE algorithm for the Bahja South data set has the following structure:



One can see that the ALPINE smoothes the overall residuals picture (reducing “outliers”) which can introduce a bias into residuals.

Other methods considered

Optimizational methods

QR factorization

Currently the forward model used in the project is the Gaussian Plume model (2), which leads to a linear inverse problem (3). There exist some well-known matrix factorization techniques to solve the system (3). One of these is the QR-decomposition. Orthogonal matrix triangularization (*QR* decomposition) reduces a real (*m*,*n*) matrix *A* with m≥n and full rank to a much simpler form. It guarantees numerical stability by minimizing errors influenced by machine roundoffs. A suitably chosen orthogonal matrix *Q* will triangularize the given matrix:

$$A = Q \begin{bmatrix} R \\ 0 \end{bmatrix} ,$$

with the (*n*,*n*) upper triangular matrix *R*. One only has then to solve the triangular system *Rx* = *Pb*, where *P* consists of the first *n* rows of *Q*.

The least squares problem Ax ≈ b is easy to solve with *A* = *QR* and $Q^TQ = I$. The solution

$$x = (A^T A)^{-1} A^T b$$

becomes

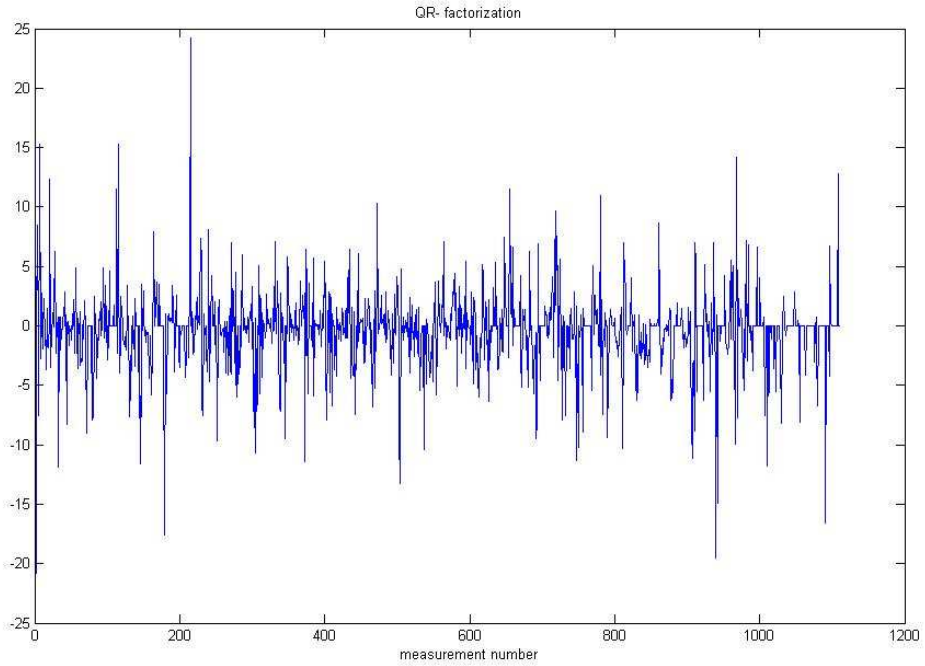
$$x = (R^T Q^T Q R)^{-1} R^T Q^T b = (R^T R)^{-1} R^T Q^T b = R^{-1} Q^T b .$$

This is a matrix-vector multiplication Q^Tb, followed by the solution of the triangular system $Rx = Q^Tb$ by back-substitution.

This method was applied to solve the system (3):

Ax ≈ C

and gave the following answer:



which has 447 negative components out of 1110. Therefore, this method gives an unphysical solution, which has to be iteratively modified in order for it to satisfy lower and upper boundary constraints, which will imply using one of the quadratic optimization methods.

Interior point method

To impose nonnegativity and an upper boundary to the solution, one has to formulate the problem explicitly:

Minimize $(Ax - C)^2$ subject to

$x \geq 0$

$x \leq$ MaxEmis, where MaxEmis – is a defined maximal possible level of emission rate.

This is a convex quadratic programming problem. There exist several methods for solving it, one of the most effective group of methods is Interior Point methods. The Logarithmic Barrier Interior Point method can be summarized as follows.

maximize f(y), subject to

fi(y) ≤ 0 (i=1..n),
y € R^m,
where: -f(y), fi(y) are convex, twice cont. differentiable

Logarithmic Barrier Algorithm

Input:
ϵ is the accuracy parameter;
τ is the proximity parameter;
θ is the reduction parameter, $0 < \theta < 1$;
μ_0 is the initial barrier value;
y^0 is a given interior feasible point such that $\|p(y^0, \mu_0)\|_{H(y^0,\mu_0)} \leq \tau$;

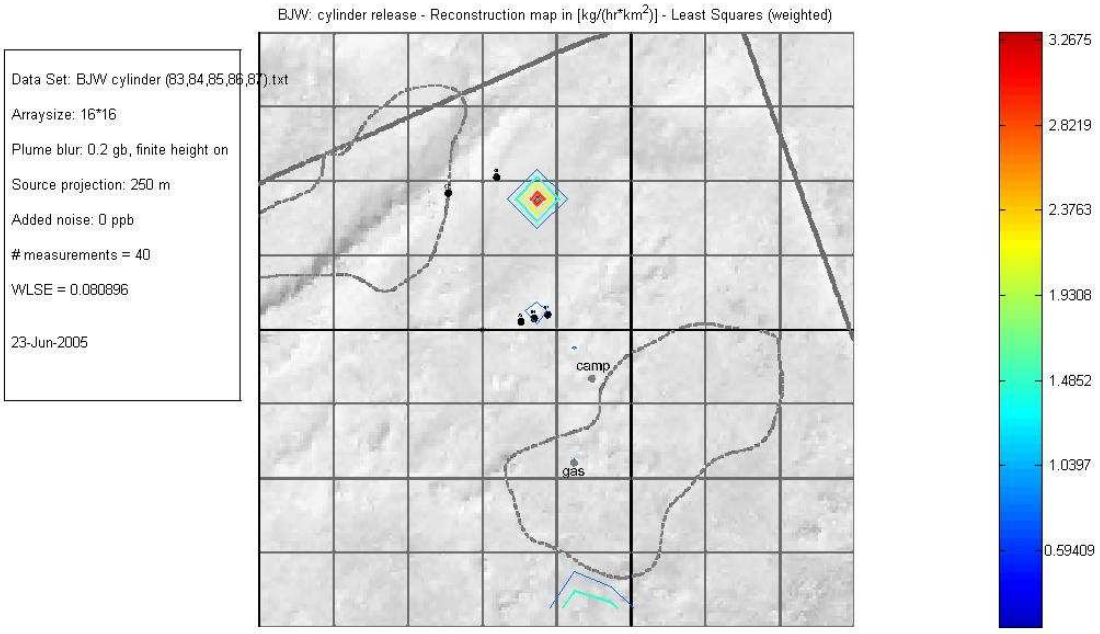
begin
  $y := y_0$; $\mu := \mu_0$;
  while $\mu > \frac{\epsilon}{4n}$ **do**
  begin (outer step)
    $\mu := (1 - \theta)\mu$;
    while $\|p\|_H \geq \tau$ **do**
    begin (inner step)
      $\tilde{\alpha} := \arg\min_{\alpha>0} \{\phi_B(y + \alpha p, \mu) : y + \alpha p \in \mathcal{F}^0\}$
      $y := y + \tilde{\alpha} p$
    end (inner step)
  end (outer step)
end.

where $\phi_B(y,\mu) = -\frac{f(y)}{\mu} - \sum_{i=1}^{n} \ln(-f_i(y))$

is the logarithmic barrier function.

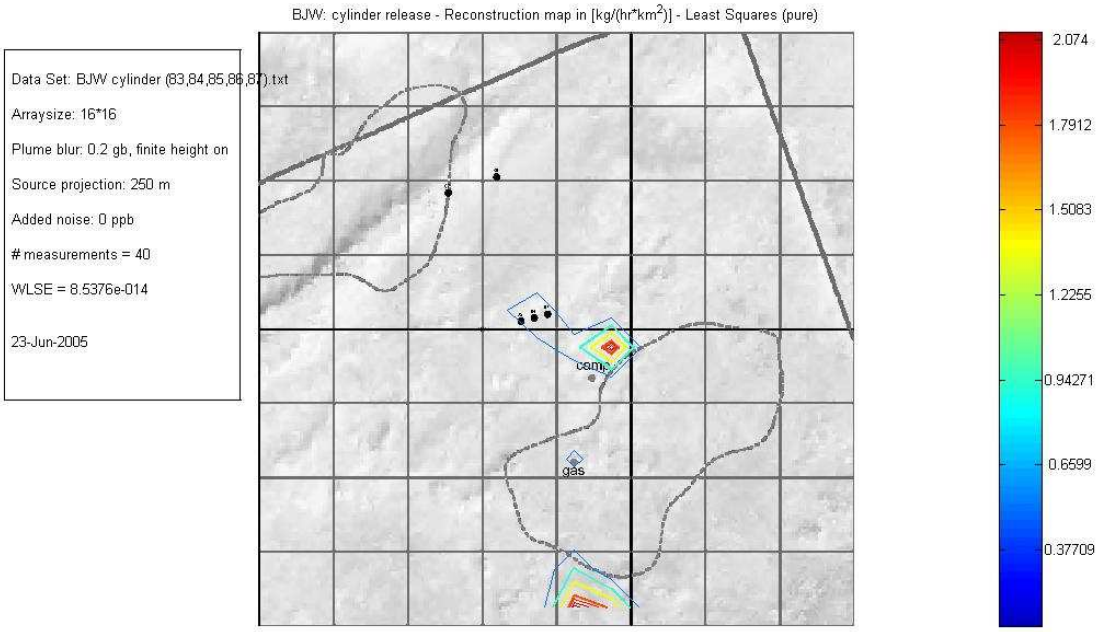
This algorithm, along with other optimization methods, finds the solution of the quadratic programming problem satisfying boundary constraints relatively fast. However, all of them depend on the cost function f(y), that is, as soon as one has to change the cost function, the whole algorithm has to be re-programmed. The obvious cost function (total

square error) does not always give a physical solution. For example, the following solution was obtained by built-in Matlab quadratic solver for weighted least squares cost function: $f(y) = (Ay - C)^2/\text{Sigma}$:

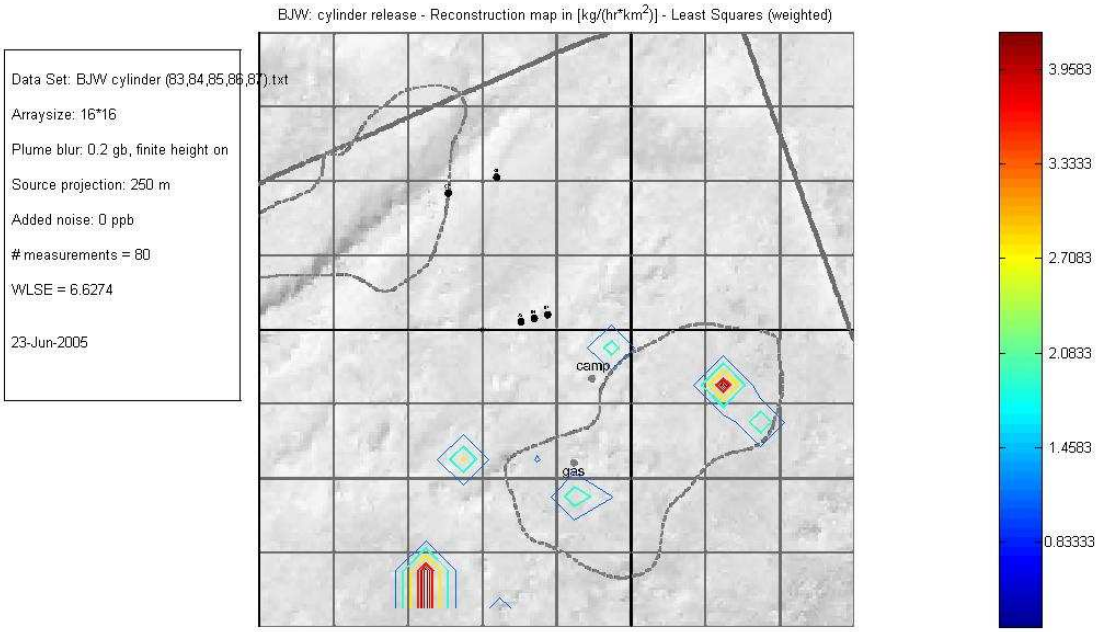


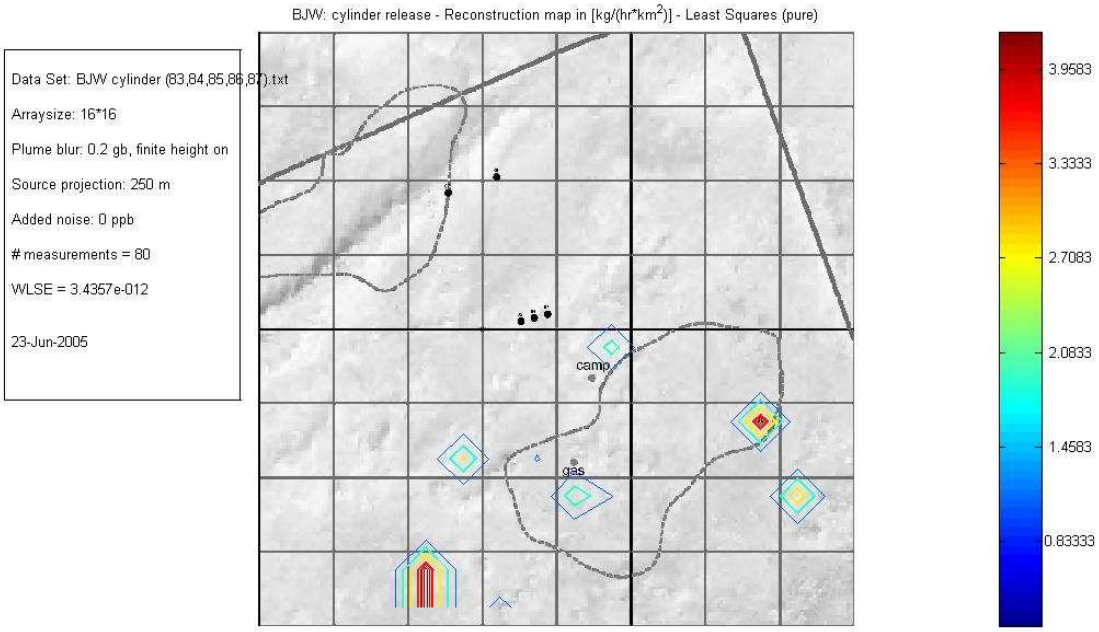
The solution above was obtained for the cylinder release dataset with 9 minutes averaging time.

The same solver gives the following "pure" least square fit
(where $f(y) = (Ay - C)^2$):



If we take 1 min averaging, the quadratic programming gives the following solution:





Kalman filtering

In 1960, R.E. Kalman published his famous paper describing a recursive solution to the discrete data linear filtering problem [Kalman60]. Since that time, due in large part to advances in digital computing, the Kalman filter has been the subject of extensive research and application.

The Kalman filter addresses the general problem of trying to estimate the state $x \in \Re^n$ of a discrete-time controlled process that is governed by the linear stochastic difference equation

$$x_k = F_k x_{k-1} + B_k u_{k-1} + w_{k-1}$$

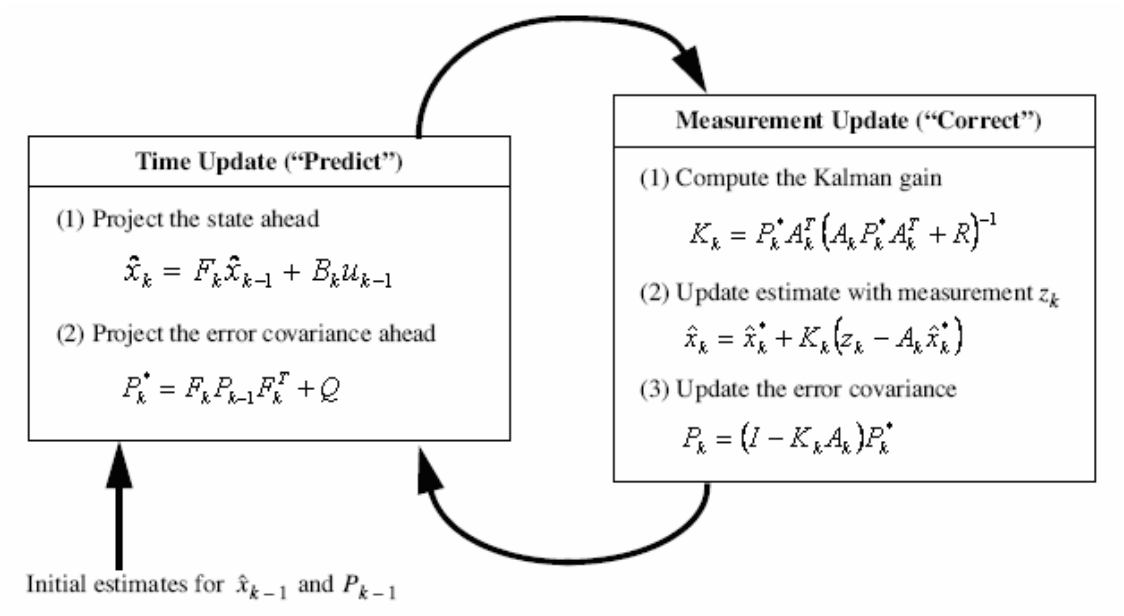
with a measurement $z \in \Re^m$ that is

$$z_k = A_k x_k + v_k$$

The random variables w_k and v_k represent the process and measurement noise (respectively). They are assumed to be independent (of each other), white, and with normal probability distributions

$$p(w) \sim N(0, Q),$$
$$p(v) \sim N(0, R).$$

In practice, the process noise covariance Q and measurement noise covariance R matrices might change with each time step or measurement, however here we assume they are constant. The Kalman filtering assumes the following iterative update of the system state:



The idea of the algorithm is to find an equation that computes an a posteriori state estimate as a linear combination of an a priori estimate and a weighted difference between an actual measurement and a measurement prediction.

In terms of the LightTouch project, the system state x_k will stand for the vector of source strengths, estimated for time k; F_k would be the transition matrix from state x_{k-1} to x_k, given wind characteristics and the time interval between states k and k-1 (a gas dispersion model). B_k is an optional parameter, that can serve to tune the dispersion model in use, or be set to zero. A_k will be the Gaussian plume matrix defined in (3), that relates predicted measurements with the sources.

The scheme above is meant for a prediction of the system state at the next time frame. However, we can also reconstruct the initial system state x_0 by modifying the system equation:

$$\begin{cases} \begin{pmatrix} x \\ x_0 \end{pmatrix}_k = \begin{bmatrix} F_k & 0 \\ 0 & I \end{bmatrix}_k \begin{pmatrix} x \\ x_0 \end{pmatrix}_{k-1} + \begin{bmatrix} B_k \\ 0 \end{bmatrix} u_{k-1} + \begin{bmatrix} I \\ 0 \end{bmatrix} w_{k-1} \\ z_k = A_k x_k + v_k \end{cases}$$

Then, applying the same scheme to the new state $(x\ x_0)_k$ will improve the estimate of x_0 at each iteration.

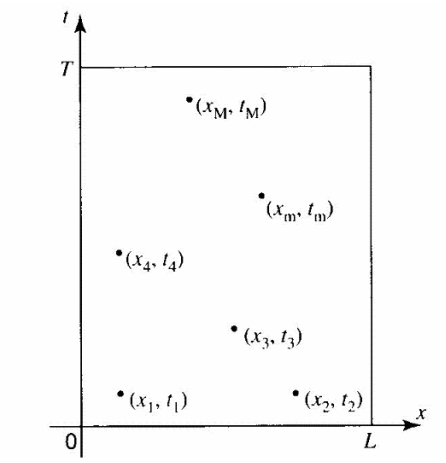
The Kalman filtering approach was initially considered as the way to continue the project. It has a clear advantage over other methods: its iterative predictive-corrective structure allows "on-line" updates of the system state as new measurements become available. One of the major drawbacks of the method (with respect to the LightTouch project) would be the requirement of specifying the gas dynamics matrix F_k at each time frame k and therefore implementing the forward model anew. Another minuspoint is that the system state doubles for the modified problem – if we use 40x40 grid size, the problem dimension would be 3200, instead of 1600. After much thinking and assessment the priority was given to probabilistic methods, thereby rejecting the Kalman filtering approach.

Variational methods

Variational methods can also be applied to reconstruction of gas or water dynamics in the following way. Suppose we need to find the function u(x,t) (which can represent, for example, the gas concentration at the point x at time t) that satisfies the Euler-Lagrange equation

$$\frac{\partial u_F}{\partial t} + c\frac{\partial u_F}{\partial x} = F$$

Suppose also that we have measurements $d_1, \ldots, d_m$ such that



$$d_m = u(x_m, t_m) + \epsilon_m,$$

where u(x,t) is the real concentration at point x at time t, and ε is the measurement error.

Then, if we introduce the representers functions for the Euler-Lagrange equation, as it shown in [6], this will lead to the following forward (Fm) and backward (Bm) systems

$$(\mathrm{B}_m)\begin{cases} -\dfrac{\partial \alpha_m}{\partial t} - c\dfrac{\partial \alpha_m}{\partial x} = \delta(x - x_m)\delta(t - t_m) \\ \alpha_m(x, T) = 0 \\ \alpha_m(L, t) = 0. \end{cases} \qquad (\mathrm{F}_m)\begin{cases} \dfrac{\partial r_m}{\partial t} + c\dfrac{\partial r_m}{\partial x} = W_f^{-1}\alpha_m \\ r_m(x, 0) = W_i^{-1}\alpha_m(x, 0) \\ r_m(0, t) = cW_b^{-1}\alpha_m(0, t). \end{cases}$$

If we now look for the approximation of u(x,t) in the form

$$\hat{u}(x, t) = u_F(x, t) + \sum_{m=1}^{M} \beta_m r_m(x, t),$$

This will lead (see [6]) to the following solution

$$\hat{u}(x, t) = u_F(x, t) + (\mathbf{d} - \mathbf{u}_F)^{\mathrm{T}}(\mathbf{R} + \mathbf{w}^{-1})^{-1}\mathbf{r}(x, t).$$

This method can be interesting for reconstruction of a gas profile in time. However, its major drawback is that the time dependence is involved. Therefore, one needs to have enough measurement data to cope with the time dimension.

Conclusions, Recommendations

As we see from the cylinder release data set (for which we know the correct result), the modified method ALPINE gives a very good result. As the next steps for improving the method, the author would recommend the following:

- Try to use different genetic combinations of the sources left after the 90% - selection. For the moment, a cell-wise uniform selection among all those sources performs best. However, one may find better ways of combining the significant patterns to generate new ones in place of those discarded
- An objective way of convergence assessment for the modified algorithm is crucial. Since the modified algorithm performs only one PARFUM iteration instead of the whole run, it does not use the value of Tolerance. Therefore, there's a need for an objective "goodness of fit", based on which we could decide whether to stop the algorithm or continue running. Consider a residuals measure or progressive modification of the parameters defining the Beta distribution.
- A different distribution for generating the source strengths may help. Up until now, the Beta distribution gives reasonable results and performs a lot better than uniform or cut normal model. In addition, some dependence among neighbouring cells might be included. This will introduce a bias into the size and strength of reconstructed sources, which might be motivated by some physical statements, such as the Gutenberg-Richter law.
- The forward model has only an a priori effect on the methods. So, improvements to the forward model would be another area to investigate, as a better forward model will lead to a better input data for the probabilistic inversion methods, which in its turn will give more accurate and stable reconstruction.
- Some new ideas, not yet incorporated into the probabilistic inversion directly, may help. For example, one may try an iterative grid refinement. That is, starting from a coarser grid, perform a (fast) reconstruction on it and then, according to some criteria, decide whether to refine the grid further. For that, a penalizing function on the number of cells should be introduced. There is no common

consideration for taking such a function, so the problem is to choose the one that gives the best possible result. Such an approach might be amenable to the concepts of simulated annealing and should allow more extensive searches of the full parameter space.

References

1. Kurowicka D, Cooke R., Uncertainty Analysis: mathematical foundations and applications, DIAM, TU Delft, 2004
2. Bill Hirst, Steve Gillespie, Ian Archibald, Olaf Podlah, Graham Gibson, Ken Skeldon, Johannes Courtial, Steve Monk & Miles Padgett "Oil and gas prospecting by ultra-sensitive optical gas detection with inverse gas dispersion modelling" Geophysical Research Letters: Vol 31, L2115, 2004, The Background section presented in this thesis is take from this paper with the permission of the authors.
3. The Optics group website: http://www.physics.gla.ac.uk/Optics/
4. Welch G., Bishop G., "An Introduction to the Kalman Filter". University of North Carolina, Department of Computer Science, TR 95-041, 1995.
5. Hertog, Dick den "Interior Point Approach to Linear, Quadratic and Convex Programming" PhD dissertation, TU Delft, 1992.
6. Bennett, Andrew F., "Inverse modelling the ocean and atmosphere" Cambridge, UK : Cambridge University Press, 2002

Appendix

All programs enclosed here were written and tested in Matlab 7.0 and additionally require the Statistics toolbox.

ALPINE.M – Modified algorithm - genetic approach + modified distribution

```
clear all

N = 25000;              % number of samples
MaxEmis = 5;            % Max. emission [kg/s]
format short g
rus = 6; rns = 2;       % Initial random state
avtime = 9;             % Averaging time, minutes
rand('state',rus);
randn('state',rns);
GS = 16;                % grid size

% ind = [1:1014];       % optional - indices of measurements to be used
% mind = [136 1:16];    % optional - cell indices required to have zero source strength
mind = [];              % optional: measurement locations, as above
A = single(load('A_pr9.txt')');         % A matrix gives dispersion coefficients
% A = A(ind,:);         % active if <ind = [1:1014];> is active
ns = size(A,2);
ind = 1:size(A,1);      % should not be active if <ind = [1:1014];> is active
eff = setdiff(find(sum(A)>0),mind);
A = A(:,eff);
C = load('C_pr9.txt')'*3600;            % C - file of measured conc-s [kg/m3] - confusion with
ChoiceRecon units
C = C(ind);
sig = load('sig_pr9.txt')*3600;         % sigma concentrations [kg/m3]
sig = sig(ind);
[m n] = size(A);                        % m - # measurements, n - # effective sources

S = single((betarnd(0.01,0.9,N,n))*MaxEmis);    % S -the generated N random source patterns

format short e;

p = ones(N,1)/N;        % vector of weights for the N patterns

for NIter = 1:20        % number of iterations to be performed before stopping

   [i j] = sort(p,'descend');
   pind = find(cumsum(p(j))<=0.9,1,'last')      % chooses those patterns that comprise 90% of
weight
                                                % from the previous iteration
   OS = [S(j(1:pind),:); p'*S; betarnd(0.02,0.9,50,n)];
   for j = 1:n          % generating new patterns to replace those abandoned
      S(:,j) = OS(round(rand(N,1)*(pind+50))+1,j);
   end;

   Y = single(S*A');    % Nxm matrix of predicted concentrations

   Prime = single(zeros(N,m));
   for meas = 1:m       % computing weights for each measurement
```

```
    eq = find(abs(Y(:,meas)-C(meas))<sig(1)/1000);          % zero-threshhold is sig(1)/1000: eq
stores indices of those predicted concentration close to actual concentration (can be empty)
    if length(eq)>0
       Prime(eq,meas)=0.9/length(eq);
       Prime(setdiff(1:N,eq),meas)=0.1/(N-length(eq));
    else
       p = 1./(abs(Y(:,meas)-C(meas))).^2;                  % weights are set to be ~ 1/d^2 where d
is the residual concentration
       Prime(:,meas) = p/sum(p);
    end;

 end;

 p = mean(Prime,2);
 clear Prime;

 WLSE = sum(((A*S'*p-C)./sig').^2)/m
              % weighted least square error, same as 'Chi^2' in
              % AvRecon

 % transferring to AvRecon representation so as to generate compatible output files:
 clear Y;
 SS = single(zeros(N, ns));
 SS(:,eff) = 4*S;

 Es = SS'*p;              % calculate mean source values
 % Vs = S'.^2*p - Es.^2;
               % calculate variance of each source

 % saving as same Excel spreadsheet format as used by AvRecon:
 sources = zeros(GS,GS);
 for i=1:GS
    sources(GS+1-i,:)=Es(i*GS-GS+1:i*GS);
 end;
 XLSWRITE('sources.xls',sources);       % saving in XLS format


 % plotting residuals results:
 hf = figure(1);
 set(1,'visible','off');
 plot(1:m,(A*Es(eff)-C)/3600,'.');
 if NIter == 1
    ax = axis;
 else axis(ax);
 end;
 grid on;
 xlabel('measurement number');
 title(['Residual plot - Run ' num2str(NIter) '; WLSE = ' num2str(WLSE)]);
 fname = ['pr9_resid_' num2str(rus) num2str(rns)  '_run' num2str(NIter)];
       % filename, will be 'SC40_resid_03_run7.jpg' if
       % rus = 0; rns = 3; NIter = 7
 saveas(hf,fname,'jpg');

 % graphical output map showing source strengths:
 hf = figure(4);
 set(4,'visible','off');
 clf;
 im = imread('mapbw.jpg','jpg');     % write the map file (jpg)
 image(im);
 hold on;
 Min = min(Es);
```

```
   Max = max(Es);
   L = length(im);
   dh = L/GS;
   colorbar('YTickLabel',num2str([(Max-Min)/7:(Max-Min)/7:Max]'));
   J = jet(80);
   colormap(J(5:75,:));
   contour([dh/2:dh:L],[dh/2:dh:L],sources*500/Max,[250 300 350],'linewidth',1);
   contour([dh/2:dh:L],[dh/2:dh:L],sources*500/Max,[400 430 460 490],'linewidth',2);
   contour([dh/2:dh:L],[dh/2:dh:L],sources*500/Max,[100 150 200],'linewidth',1,'linestyle','-');

   title(['BAHJA SOUTH Reconstruction map in [kg/(hr*km^2)] - ALPINE (sqr): Iteration '
num2str(NIter)]);
   t = annotation('textbox',[.02,.22,0.21,0.68]);
   set(t,'string',strvcat(' ','Data Set: BJS 1min def.(-early-late).txt',' ',['Arraysize: ' num2str(GS) 'x'
num2str(GS) '; Averaging ' num2str(avtime) ' min'],' ','Plume blur: 0.2 gb, finite height on','
','Source projection: 250 m',' ','Added noise: 0 ppb;  AvPos: On',' ',...
    ['# samples = ' num2str(N)],' ',['# effective sources = ' num2str(n)],' ',['# measurements = '
num2str(m)],' ',...
      ['WLSE = ' num2str(WLSE)],' ',...
       ['Total flux = ' num2str(sum(Es/4)) ' kg/hr'],' ',['Random State: ' num2str(rus) '-'
num2str(rns)],' ',' ',date));
                          % write out all the parameters:
                          % includes those used to generate A
                          % matrix and this model's paramaters
set(t,'string',strvcat(' ','Data Set: Cylinder release',' ',['Arraysize: ' num2str(GS) 'x' num2str(GS) ';
Averaging ' num2str(avtime) ' min'],' ','Plume blur: 0.2 gb, finite height on',' ','Source projection:
250 m',' ','Added noise: 0 ppb;  AvPos: On',' ',...
      ['# samples = ' num2str(N)],' ',['# effective sources = ' num2str(n)],' ',['# measurements = '
num2str(m)],' ',...
      ['WLSE = ' num2str(WLSE)],' ',...
      ['Total flux = ' num2str(sum(Es/4)) ' kg/hr'],' ',['Random State: ' num2str(rus) '-'
num2str(rns)],' ',' ',date));
                          % write out all the parameters:
                          % includes those used to generate A
                          % matrix and this model's paramaters
   axis square;
   axis off;
   fname = ['pr9_' num2str(rus) num2str(rns)  '_run' num2str(NIter)];
%    saveas(hf,fname);              % uncomment if you want to save as Matlab figure
   saveas(hf,fname,'jpg');
end;
```

PARFUM

```
clear all

N = 10000; % number of samples
MaxEmis = 5; % Max. emission [kg/hr*m3]
format short g
rus = 0; rns = 0;
avtime = 9; % Averaging time, minutes
rand('state',rus);
randn('state',rns);

% ind = [1:40];
mind = [];
% mind = [];
A = load('A_pr.txt')';
% A = A(ind,:);
ns = size(A,2);
ind = 1:size(A,1);
eff = setdiff(find(sum(A)>0),mind);
A = A(:,eff);
C = load('C_pr.txt')'*3600; % measured conc-s [kg/hr*m3]
C = C(ind);
sig = load('sig_pr0.txt')*3600; % sigma concentrations [kg/hr*m3]
sig = sig(ind);
[m n] = size(A); % m - # measurements, n - # effective sources

S = betarnd(0.05,0.9,N,n)*MaxEmis;
Y = S*A'; % input data for the PI
save samples S;
clear S;

C(find(C<0)) = sig(find(C<0)); % in case some conc-s are negative

rs = repmat(sig,N,1);
rc = repmat(C',N,1);
W = [sum(Y<rc-rs)' sum(Y<rc)' sum(Y<rc+rs)' N*ones(m,1)];
[W(:,1) diff(W,1,2)]/N
min([W(:,1) diff(W,1,2)]/N)

% prior probability
p = ones(N,1)/N;

% PARFUM iteration
q = [0.05 0.45 0.45 0.05];
x = 1;
eps = 1e-6;
format short e;

[i1 j1] = find(Y<rc-rs);
[i2 j2] = find((Y>=rc-rs)&(Y<rc));
[i3 j3] = find((Y>=rc)&(Y<rc+rs));
[i4 j4] = find(Y>=rc+rs);
fprintf('Iterative change   |    min(p)     |   max(p)\n');

while (x>eps)
x = p;
Prime = zeros(N,m);

   for meas = 1:m
      p = x;
```

```
      %if q(1)>0
      % update  for the 1st quantile
      Aij = sum(p(i1(find(j1==meas)))); % P(A_ij)
      if Aij > 0
      p(i1(find(j1==meas))) = p(i1(find(j1==meas)))*q(1)/Aij;
      end;

      % update  for the 2nd quantile
      Aij = sum(p(i2(find(j2==meas)))); % P(A_ij)
      p(i2(find(j2==meas))) = p(i2(find(j2==meas)))*q(2)/Aij;

      % update  for the 3rd quantile
      Aij = sum(p(i3(find(j3==meas)))); % P(A_ij)
      p(i3(find(j3==meas))) = p(i3(find(j3==meas)))*q(3)/Aij;

      % if q(4)>0
      % update  for the 4th quantile
      Aij = sum(p(i4(find(j4==meas)))); % P(A_ij)
      if Aij >0
      p(i4(find(j4==meas))) = p(i4(find(j4==meas)))*q(4)/Aij;
      end;

      Prime(:,meas) = p/sum(p);

   end;

p = mean(Prime,2);

x = sum((x-p).^2);
% fprintf('%14.2e     |%12.2e   |%12.2e\n',[x min(p) max(p)]);

fprintf('%14.2e     |%12.2e   |%12.2e\n',[x min(p) max(p)]);
% for k = 1:m
%          fprintf('   %2.0f        %1.5f    %1.5f    %1.5f    %1.5f\n', ind(k),  sum(p(i1(find(j1==k)))),
sum(p(i2(find(j2==k)))), sum(p(i3(find(j3==k)))), sum(p(i4(find(j4==k)))));
% end;

end;

load samples;

WLSE = sum(((A*S'*p-C)./sig').^2)/m % Forward model, WLSE

fprintf('Meas-t  P(q1)    P(q2)    P(q3)    P(q4)\n');
for k = 1:m
   fprintf('    %2.0f          %1.5f    %1.5f    %1.5f    %1.5f\n', ind(k),  sum(p(i1(find(j1==k)))),
sum(p(i2(find(j2==k)))), sum(p(i3(find(j3==k)))), sum(p(i4(find(j4==k)))));
end;

% back to initial representation

SS = zeros(N, ns);
SS(:,eff) = S;
S = 4*SS;
clear SS;

Es = S'*p; % mean source values
% Vs = S'.^2*p - Es.^2; % source variance

% saving as Excel spreadsheet:
sources = zeros(16,16);
```

```
for i=1:16
   sources(17-i,:)=Es(i*16-15:i*16);
end;
% XLSWRITE('sources.xls',sources);

% plotting results

figure(4);
clf;
im = imread('mapbw.jpg','jpg');
image(im);
hold on;
Min = min(Es);
Max = max(Es);
L = length(im);
dh = L/16;
% contour([25:dh:800],[25:dh:800],sources*500/Max,[150 200 300 400 470],'linewidth',2);
% contour([dh/2:dh:L],[dh/2:dh:L],sources*500/Max,[200 300 400],'linewidth',2);
% contour([dh/2:dh:L],[dh/2:dh:L],sources*500/Max,[100 425 450 475],'linewidth',1);
colorbar('YTickLabel',num2str([(Max-Min)/7:(Max-Min)/7:Max]'));
J = jet(80);
colormap(J(5:75,:));
contour([dh/2:dh:L],[dh/2:dh:L],sources*500/Max,[250 300 350],'linewidth',1);
contour([dh/2:dh:L],[dh/2:dh:L],sources*500/Max,[400 430 460 490],'linewidth',2);
contour([dh/2:dh:L],[dh/2:dh:L],sources*500/Max,[100 150 200],'linewidth',1,'linestyle','-');

title('BJW: cylinder release - Reconstruction map in [kg/(hr*km^2)] - PARFUM');
t = annotation('textbox',[.02,.22,0.21,0.68]);
set(t,'string',strvcat(' ','Data Set: BJW cylinder (83,84,85,86,87).txt',' ',['Arraysize: 16*16; Averaging
' num2str(avtime) ' min'],' ','Plume blur: 0.2 gb, finite height on',' ','Source projection: 250 m',' 
',','Added noise: 0 ppb;  AvPos: On',' ',...
   ['# samples = ' num2str(N)],' ',['# effective sources = ' num2str(n)],' ',['# measurements = ' 
num2str(m)],' ',...
   ['Tolerance = ' num2str(eps)],' ',['WLSE = ' num2str(WLSE)],' ',...
   ['Total flux = ' num2str(sum(Es/4)) ' kg/hr'],' ',['Random State: ' num2str(rus) '-' num2str(rns)],'
',' ',date));
axis square;
axis off;

figure(5);
plot(1:N,p,'.');
title('Final weights of samples');
xlabel('Sample number');
ylabel('weight');


% conf = 0.9;
% [p j] = sort(p);
% qc = find(cumsum(p)>conf,1);
%
% Y(j(qc:N),:)
% [S(j(qc:N),:); mean(S(j(qc:N),:))]*A'
```

IPF

```
N = 50000; % number of samples
MaxEmis = 4; % Max. emission [kg/hr*m3]
format short g

A = load('A_pr.txt')';
ns = size(A,2);
eff = find(sum(A)>0);
A = A(:,eff);
C = load('C_pr.txt')'*3600; % measured conc-s [kg/hr*m3]
sig = load('sig_pr0.txt')*3600; % sigma concentrations [kg/hr*m3]
[m n] = size(A); % m - # measurements, n - # effective sources

S = betarnd(0.03,0.9,N,n)*MaxEmis;
Y = S*A'; % input data for the PI

C(find(C<0)) = sig(find(C<0)); % in case some conc-s are negative

rs = repmat(sig,N,1);
rc = repmat(C',N,1);
W = [sum(Y<rc-rs)' sum(Y<rc)' sum(Y<rc+rs)' N*ones(m,1)];
[W(:,1) diff(W,1,2)]/N
min([W(:,1) diff(W,1,2)]/N) % Possible Exception: empty set A_ij

% prior measure
p = ones(N,1)/N;

% IPF iteration
q = [0.05 0.45 0.45 0.05];
x = 1;
eps = 1e-6;
format short e;

[i1 j1] = find(Y<rc-rs);
[i2 j2] = find((Y>=rc-rs)&(Y<rc));
[i3 j3] = find((Y>=rc)&(Y<rc+rs));
[i4 j4] = find(Y>=rc+rs);

while (x>eps)

   x = p;

   for meas = 1:m

      %if q(1)>0
      % update  for the 1st quantile
      Aij = sum(p(i1(find(j1==meas)))); % P(A_ij)
      if Aij > 0
      p(i1(find(j1==meas))) = p(i1(find(j1==meas)))*q(1)/Aij;
      end;

      % update  for the 2nd quantile
      Aij = sum(p(i2(find(j2==meas)))); % P(A_ij)
      p(i2(find(j2==meas))) = p(i2(find(j2==meas)))*q(2)/Aij;

      % update  for the 3rd quantile
      Aij = sum(p(i3(find(j3==meas)))); % P(A_ij)
      p(i3(find(j3==meas))) = p(i3(find(j3==meas)))*q(3)/Aij;

      % if q(4)>0
```

```
      % update  for the 4th quantile
      Aij = sum(p(i4(find(j4==meas)))); % P(A_ij)
      if Aij >0
      p(i4(find(j4==meas))) = p(i4(find(j4==meas)))*q(4)/Aij;
      end;

      p = p/sum(p);

   end;

   x = sum((x-p).^2);

   [x min(p) max(p)]

end;

fprintf('Meas-t  P(q1)    P(q2)    P(q3)    P(q4)\n');
for k = 1:m
    fprintf('    %2.0f          %1.5f    %1.5f    %1.5f    %1.5f\n', ind(k), sum(p(i1(find(j1==k)))),
sum(p(i2(find(j2==k)))), sum(p(i3(find(j3==k)))), sum(p(i4(find(j4==k)))));
end;

WLSE = sum(((A*S'*p-C)./sig').^2)/m % Forward model, WLSE

% back to initial representation

SS = zeros(N, ns);
SS(:,eff) = S;
S = SS;
clear SS;

Es = S'*p; % mean source values
Vs = S'.^2*p - Es.^2; % source variance

% saving as Excel spreadsheet:
sources = zeros(16,16);
for i=1:16
    sources(17-i,:)=Es(i*16-15:i*16);
end;
XLSWRITE('sources.xls',sources);

% plotting results

figure(4);
clf;
im = imread('mapbw.jpg','jpg');
image(im);
hold on;
Min = min(Es);
Max = max(Es);
L = length(im);
dh = L/16;
% contour([25:dh:800],[25:dh:800],sources*500/Max,[150 200 300 400 470],'linewidth',2);
contour([dh/2:dh:L],[dh/2:dh:L],sources*500/Max,[200 300 400],'linewidth',2);
contour([dh/2:dh:L],[dh/2:dh:L],sources*500/Max,[100 425 450 475],'linewidth',1);
colorbar('YTickLabel',num2str([Min:(Max-Min)/5:Max]'));
title('BJW: cylinder release - Reconstruction map in [kg/(hr*km^2)]');
t = annotation('textbox',[.02,.3,0.21,0.6]);
set(t,'string',strvcat(' ','Data Set: BJW cylinder (83,84,85,86,87).txt',' ','Arraysize: 16*16',' ','Plume
blur: 0.2 gb, finite height on',' ','Source projection: 250 m',' ','Added noise: 0 ppb',' ',...
```

```
    ['# samples = ' num2str(N)],' ',['# effective sources = ' num2str(n)],' ',['# measurements = '
num2str(m)],' ',...
    ['Tolerance = ' num2str(eps)],' ',['WLSE = ' num2str(WLSE)],' ',' ',date));
axis square;
axis off;
```