SOFTWARE DEVELOPMENT FOR ODOUR IMPACT ASSESSMENT

by

Yue Ma

Department of Civil Engineering and Applied Mechanics
McGill University
Montreal, Canada

August, 2003

A thesis submitted to McGill University in partial fulfilment of the requirements for the degree of Master of Engineering

Your file Votre référence
ISBN: 0-612-98551-2
Our file Notre référence
ISBN: 0-612-98551-2

Abstract

Odour impacts are the source of many complaints in communities and, as such, methods for odour impact assessment are needed. Such assessments would benefit from the development of parameters that could be used to quantify and reflect the magnitudes of odour impacts experience by the community. Therefore, numerous parameters that can be used to assess odour impacts have been proposed in this research. These parameters account for variations in odour concentration, probability of response, degree of annoyance and population density that clearly influence the degree of impact of an odorous emission on a community. Most of these parameters require the evaluation of areas enclosed within contours and volumes under contours and, as such, are calculation intensive. In order to simplify the calculation of these parameters, algorithms have been developed and implemented into a user-friendly interface called OdorImp. This software was tested by applying it to three sets of synthetic data and two sets of data arising from actual case-studies. Comparisons were made between the results from OdorImp and exact values derived from simple cases and other values calculated using a commercial contour-evaluation program. It was demonstrated that the algorithms implemented in OdorImp are accurate and can be used to reliably evaluate the proposed odour impact parameters.

Résumé

Les odeurs sont à la source de nombreuses plaintes au sein des communautés, et donc il y a un besoin pour des méthodes d'évaluer l'impact des odeurs. Ces évaluations bénéficieraient du développement de paramètres pouvant être utilisé pour quantifier l'impact des odeurs subites par la communauté. De nombreux paramètres qui pourraient être utilisé ont été proposés lors de cette étude. Ces paramètres tiennent compte de variations dans la concentration des odeurs, de la probabilité d'une réponse, du degré de nuisance, et la densité de population - ce qui influence le degré d'impact sur une communauté d'une émission odoriférant. La plupart de ces paramètres demandent l'évaluation de surfaces enfermées dans des contours et de volumes sous des contours, et requièrent alors beaucoup de calcul. Afin de simplifier le calcul de ces paramètres, des algorithmes ont été développés et implantés en utilisant une interface graphique facile à utiliser du nom d'OdorImp. Ce logiciel a été vérifié en le faisant traiter trois séries de données synthétiques et deux séries provenant d'études réelles. Des comparaisons ont été faites entre les résultats obtenus d'OdorImp et des valeurs exactes obtenues de cas simples et d'autres valeurs calculées avec un logiciel commercial d'évaluation des contours. On a démontré que les algorithmes implantés dans OdorImp sont exacts et peuvent être utilisés de façon fiable afin d'évaluer les paramètres proposés pour quantifier les impacts des odeurs.

Acknowledgements

I wish to express my sincere appreciations my supervisor Dr. Jim Nicell for his excellent and patient supervision for this research. This thesis would not have been possible without his inspiring guidance and good suggestions.

I am grateful to Dr. Paul Henshaw of the University of Windsor for providing the data for case study 2 and Mr. Tamer Gorgy for providing the data for case study 1.

Special thanks are extended to Mr. Ji Peng for all his support. And special thanks are also extended to my friends Mr. Alex Plate and Mr. Serge Leonard for providing me with the translation of the abstract and their friendship.

A very special thanks is owed to my dear husband. He helped me solve many computer problems and encouraged me in the face of difficulties. This research is my gift to him to thank him for his support in the past ten years. I also would like to express my deepest gratitude to my mother, my father, and my sister for their encouragement and invaluable love.

Table of Contents

List of Figures

List of Tables

Nomenclature

NOTE: Dimension are expressed in L (distance), M (mass), and time (T)

aX, bX, cX	x coordinates of points a, b, and c
aY, bY, cY	y coordinates of points a, b, and c
aZ, bZ, cZ	z coordinates of points a, b, and c
A	Annoyance on a scale of 0 to 10, in annoyance units, au ($0 \leq A \leq 10$)
A_{max}	Peak value of annoyance
$A(x, y)$	Annoyance of the odour at a receptor located at point (x, y), au
a	Persistence of annoyance, dimensionless ($0 \leq a \leq 1.0$)
C	Concentration of an odour, in M/L^3
ContV	Contour value
$C_{50\%}$	Threshold at which 50% of the population responds to an odour, in M/L^3
C_{5au}	Odour concentration, in OU, at which the population annoyance, A, is 5 au
C_{ou}	Odour concentration, in odour units, OU
$C_{ou}(x, y)$	Concentration of the odour at a receptor located at point (x, y), in OU
C_{max}	Peak values of an odour concentration, in OU
$C(x, y)$	Concentration at a receptor located at point (x, y), in M/L^3 or OU
D	Dilutions of an odour, in dilutions
$D_{50\%}$	Threshold at which 50% of the population responds to an odour, in dilutions
$D(x, y)$	Dilutions of an odour at a receptor located at point (x, y), in dilutions
D_{5au}	Dilution of an odour at which the population annoyance, A, is 5 au
F	Footprint area under contour of a half sphere or a taper, in L^2
$F(A)$	Footprint area under an A annoyance contour, in L^2
$F(C)$	Footprint area under the C concentration contour, in L^2
$F(P)$	Footprint area under the P probability contour, in L^2
$F_W(A)$	Annoyance weighted footprint area under an A annoyance contour, in $au \cdot L^2$
$F_W(P)$	Probability weighted footprint area under a P probability contour, in L^2

$F_W(C)$	Concentration weighted footprint area under a C concentration contour, in OU· L^2
F_{WA}	Total annoyance weighted footprint area in an infinite area, in au· L^2
F_{WC}	Total concentration weighted footprint area in an infinite area, in OU· L^2
F_{WP}	Total probability weighted footprint area in an infinite area, in L^2
$F_{WA}(T)$	Total annoyance weighted footprint area in a study area, in au· L^2
$F_{WC}(T)$	Total concentration weighted footprint area in a study area, in OU· L^2
$F_{WP}(T)$	Total probability weighted footprint area in a study area, in L^2
H	Plume release height above ground in simple Gaussian dispersion Model, in L
M	Emission rate of pollutant from source, in M/T or OU· L^3/T
N_P	Total number of people affected in the region, in capita
NA	Population-weighted annoyance at any given receptor, in au·capita /L^2
NC	Population-weighted concentration at any given receptor, in OU·capita/L^2
NP	Population-weighted probability at any given receptor, in capita/m^2
NA_{max}	Peak value of population-weighted annoyance, in au·capita
NC_{max}	Peak value of population-weighted concentration, in OU·capita
NP_{max}	Peak value of population-weighted probability, in capita
N_{WA}	Annoyance-weighted population in an infinite area, in au·capita
N_{WC}	Concentration-weighted population in an infinite area, in OU·capita
N_{WP}	Probability-weighted population in an infinite area, in capita
$N(A)$	Number of people affected in the region bounded by an annoyance contour, A, in capita
$N_W(A)$	Annoyance-weighted population in a region bounded by a annoyance contour A, in au·capita
$N_{WA}(T)$	Annoyance-weighted population in a study area, in au·capita
$N(C)$	Number of people affected in the region bounded by an concentration contour, C, in capita
$N_W(C)$	Concentration-weighted population in a region bounded by a concentration contour, C, in OU·capita
$N_{WC}(T)$	Concentration-weighted population in a study area, in OU·capita

$N(P)$	Number of people affected in the region bounded by the probability contour, P, in capita
$N_W(P)$	Probability-weighted population in a region bounded by probability contour P, in capita
$N_{WP}(T)$	Probability-weighted population in a study area, in capita
$N_D(x, y)$	Density of the population at a receptor located at point (x, y), capita/ L^2
P	Probability of response, in % ($0 \leq P \leq 100$)
P_{max}	Peak value of probability of response
$P(x, y)$	Probability of response of an odour at a receptor located at point (x, y), in % ($0 \leq P(x, y) \leq 100$)
p	Persistence of response, dimensionless ($0 \leq p \leq 1.0$)
Ra	Radius of sphere and the bottom plan of taper, in L
R	Ratio of $D_{5au}/D_{50\%}$ or $C_{50\%}/C_{5au}$
u	Horizontal wind velocity, in L/T
V	Volume of half sphere and taper, in L^3
x	Distance directly downwind from source in simple Gaussian dispersion model, in L
y	Perpendicular (crosswind) distance from source in simple Gaussian dispersion model, in L
σ_y	Horizontal dispersion parameter in simple Gaussian dispersion model, in L
σ_z	Vertical dispersion parameter in simple Gaussian dispersion model, in L
Δx	Grid cell dimension in the x direction, in L
Δy	Grid cell dimension in the y direction, in L

List of Abbreviations

3D	Three Dimensions
AWFA	Annoyance Weighted Footprint Area
CWFA	Concentration Weighted Footprint Area
BLUE	Best Linear Unbiased Estimator
BPIP	Building Profile Input Program
D/T	Dilution-to-Threshold
GIS	Geographical Information System
GUI	Graphical User Interface
ISC	Industrial Source Complex
OIM	Odour Impact Model
OME	Ontario Ministry of Environment
OU	Odour Units
PWFA	Probability-Weighted Footprint Area
NPA	Number of People Affected
USEPA	United States Environmental Protection Agency

1 Introduction

Air pollution has become one of the most important environmental issues resulting from the development of industries and agricultural operations (Artis, 1984). Odours are ranked as the major generators of public complaints regarding air pollutants (National Research Council Committee on Odors, 1979). According to statistics gathered from regulatory agencies, in 1994 more than 60% of air pollution complaints in the USA were related to odours with an estimated total of over 12,000 registered complaints reported to 25 agencies (Leonardos, 1996). These complaints originated as a result of a wide variety of industries and operations including agriculture, sewage treatment plants, paint facilities, refineries, plastics facilities, resin and chemical manufacturers, rendering plants, pulp mills, and landfills (Leonardos, 1996). Odour problems are very complex not only because their diverse sources but also people's reactions to odours are quite different (Artis, 1984).

Exposure to odorous emissions may not result in physical harm to the human body; however, long term or frequent exposures to odours certainly affects the quality of life and usually generates unpleasant psychological reactions (Gostelow, et. al., 2001). In addition to potential physiological problems, odours are frequently considered a nuisance and often cause complaints. In general, communities are not satisfied with the approach of assessing the degree of nuisance by using personal observations and the judgment of local authorities (Harreveld, et. al., 1999). In addition, the owners and operators of facilities from which odorous emissions originate would prefer to rely on objective strategies for assessing the impacts of odorous emission on surrounding population. Therefore, research is required to develop odour impact assessment methods that satisfy community and industry needs. A uniform method to quantitatively define the degree of odour impact may provide a basis for regulatory agencies and industries to minimize or eliminate odour nuisances for communities more effectively (Henshaw, et. al., 2002).

The ideal assessment method should take into account the many characteristics of odours, such as intensity, duration, frequency, quality, pervasiveness, acceptability, etc. that may have an influence on the degree of impact (Schulz and Harreveld, 1996). Most current assessment methods are based on either statutory nuisance laws or the dilution-to-threshold principle. Nuisance laws usually use some terms as a basis for judging impact, such as "discomfort" or "loss of enjoyment" (Government of Ontario, 1990), which are entirely subjective and are open to interpretation by the local regulatory agency and courts. Moreover, no action can be taken until an impact has already been felt by the community. The legal proceedings involved in such cases tend to be very expensive, time-consuming and risky for both the plaintiff and the defendant to undertake (Nicell, 1999). These shortcomings prevent this approach from effectively protecting the public and the industry involved (Henshaw, et. al., 2002).

In order to introduce a measure of objectivity into approaches to odour impact assessment, methods have been developed based on the dilution–to-threshold principle. This principle provides the basis for the most popular measure of odour concentration, which is expressed in "odour units" (OU). The threshold of response is defined as the number of dilutions at which 50% of a panel of odour judges responds to a stimulus as being different from odour free blanks (Nicell, 1986). The greater the number of dilutions that are required to reduce the odour to a level where only 50% of the population can respond to the odour, the greater must be the concentration of the odour. Thus, the threshold can be used as a surrogate measure of odour concentration since a high threshold reflects a high concentration of an odour. The number of odour units (OU) in a sample is simply defined as the number of dilutions that are required to reduce the odour concentration to the threshold concentration.

While the dilution-to-threshold method has certainly introduced a level of objectivity in odour impact measurements, as illustrated in Figure 1.1, the threshold only represents the response of the population at a single concentration. It fails to account for the full range of dilutions of the odour that may be experienced in a community. In addition, the threshold method neglects the important observation that the sensitivities of individuals to odours can be substantially different. For example, an odour that is present in ambient air at a level corresponding to the population threshold (i.e., at 1 OU) is likely to be at a concentration that is many times higher than the personal threshold of a particularly sensitive individual.

In addition, even if two different odours have the same threshold, they probably do not have the same of level hedonic character (e.g., pleasantness or unpleasantness).

One sample might be more offensive at a particular odour concentration than another. Uniform quantitative analyse of annoyance is hard to achieve. Panellists are often asked to describe the odour using various schemes (Gostelow et. al., 2001). For example, the scale can range from 0 (odour perceivable) to 6 (very strong) (Cheremisinoff and Young, 1988) or 0 (tolerable) to 10 (unbearable), as shown in the curve in Figure 1.1 (Nicell, 1994).

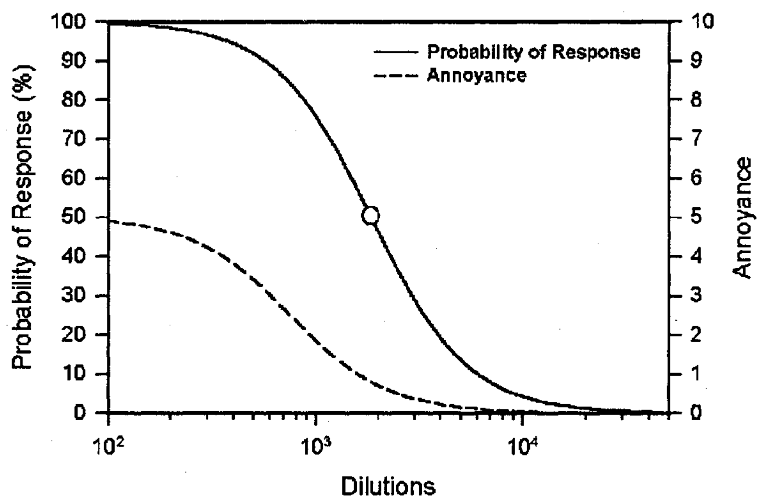


Figure 1.1: Dose-response relationship (Nicell, 2003)

The Odour Impact Model (OIM), which was first developed in 1985, can be used to establish dose-response relationships such as the curves shown in Figure 1.1 for a population subjected to a range of dilutions of an odour sample (Nicell, 1986). These relationships can be used to describe the proportion of a population that will respond an odour (i.e., probability of response, 0 to 100%) and the degree of annoyance (on a scale of 0 to 10) that they experience as it is diluted from its source strength. The OIM overcomes some of the drawbacks mentioned above and provides a measure of the response of the population to the whole range of odour concentrations that can be experienced in the field.

Based on the use of the OIM approach, Sikdar (2001) and Henshaw et.al (2002) proposed a new method for assessing odour impact. Figure 1.2 shows the steps of this method.

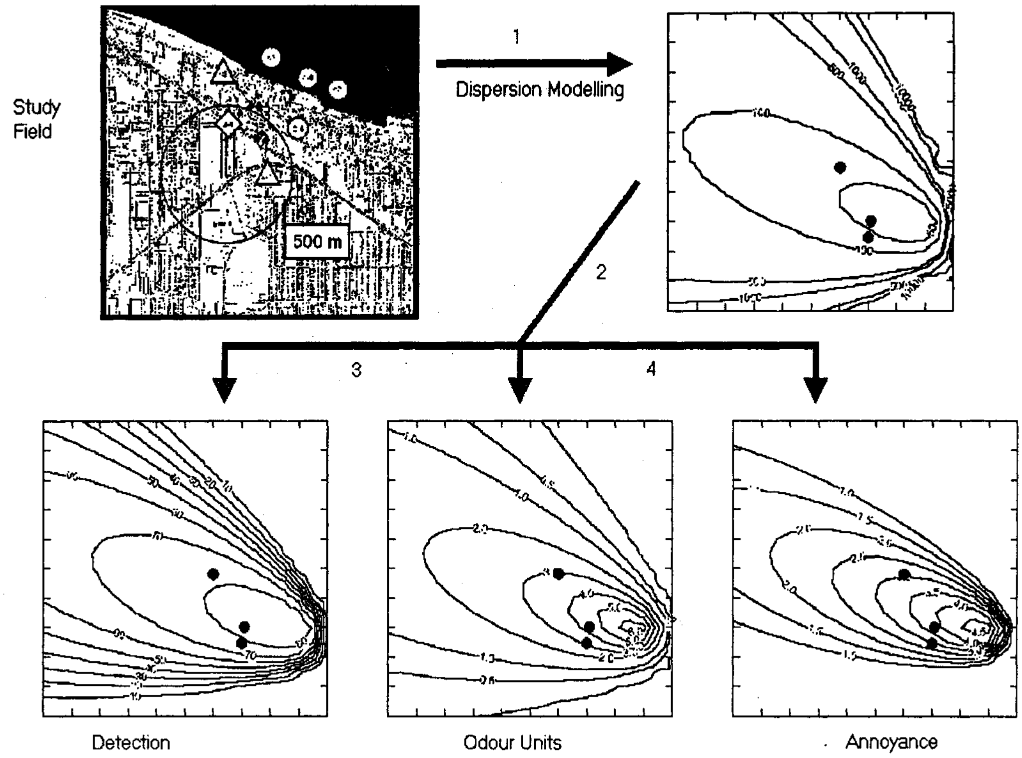


Figure 1.2: A proposed approach to odour impact assessment developed by Henshaw et. al. (2002) and Sikdar (2001)

The central idea of the approach is to combine the relationships of the OIM with a dispersion model to predict the spatial variations of population response throughout the community. They proposed to first calculate the dilutions using a dispersion model (e.g., such as the Industrial Source Complex or Aermod dispersion models of the USEPA) and secondly convert the predicted dilutions at each spatial location to odour concentration expressed in OU (Note: It is possible to use dispersion models to directly predict odour concentrations in odour units, without first predicting dilutions). Once this is done, the probability of response and the degree of annoyance at each spatial location can be calculated from curves such as those shown in Figure 1.1. In order to facilitate this, Nicell (2003) developed several equations that can be fit to OIM data and that can then be used to calculate the degree of annoyance and probability of response based on either the number of dilutions or odour units at each spatial location.

Based on this approach Henshaw et. al. (2002) and Sikdar (2001) proposed the development of contours of concentration (OU), probability of response, and annoyance at the receptors shown in Figure 1.2. They proposed that these contours could be used as a basis for evaluating parameters that would reflect the magnitude of an odour impact. For example, it was proposed that the size of a region enclosed by a particular contour (e.g., the 50% or 10% contours of probability of response) could be reflections of the magnitude of an odour impact. Thus, if one were to evaluate a "footprint" as being the size of such a region, this could be used as a quantitative measure of impact. It was also suggested that the size of the impact would be greater when more people respond to an odour. Therefore, they used the probability of response contours and the population density of the region to estimate the number of people that would respond to the odour. Of course, many other types of parameters could be envisioned, including those that would incorporate annoyance estimates. Henshaw et al (2002) suggested that these and other types of parameters should be investigated in detail to see which of them correlates best with community impact.

While the proposed approach to odour impact assessment is promising, its greatest drawback is the time consuming nature of the methods that must be followed to estimate such parameters. For example, after producing dispersion modeling predictions, specialized software is required to quantify areas inside contours or to integrate under contours (e.g., such as the software package called Surfer that is produced by Golden Software) in order to evaluate many of the parameters proposed by Sikdar (2001) and Henshaw et. al. (2002). In addition, Sikdar (2001) used a GIS package as part of her assessment procedure. Such specialized software is not only expensive but often requires training in order to use them efficiently and effectively. These problems could be overcome if software was available that would automate and simplify the approach to odour impact parameter estimation and would give researchers the ability to investigate the effectiveness of such parameters in predicting or confirming odour impact.

Therefore, the objective of this research is to develop software to directly calculate odour impact parameters assuming that the user can provide an input file of odour concentration from a dispersion model of their choice. The sub-objectives of this research include:

(i) the proposal of other parameters that may be used to assess odour impact by incorporating concentration, probability of response and degree of annoyance estimates into parameter evaluations;

(ii) the development and implementation of algorithms for calculating each of the odour impact parameters;

(iii) the testing of the accuracy of the algorithms (e.g., in area and volume calculations) using synthetic dispersion model data and comparing the results with theoretical values and those estimated using other software.

(iv) the application of the software to analyze real data arising from odour impact assessment studies; and

(v) the implementation of the program in a user-friendly interface.

2 Literature Review

Odours cause a variety of undesirable reactions in people. These reactions vary from emotional stresses such as unease, discomfort, headaches, or depression to physical symptoms including sensory irritations, headaches, respiratory problems, nausea, or vomiting (National Research Council Committee on Odors, 1979). Regulatory agencies are in the process of developing objective strategies for measuring and regulating the release of odorous emission on surrounding communities in order to minimize odour impact effectively (Ontario Ministry of Environment, 2001). In addition, owners and operators of facilities that produce odorous emissions would benefit from a non-arbitrary criterion for compliance to aid themselves in achieving compliance with odour regulations.

2.1 Approaches to odour impact assessment and regulation

Often the method of assessing the extent of the impact of an odorous emission on a community is based on an analysis of complaints from the neighbourhood. In responding to complaints, regulatory agencies often are forced to respond to the situation through the public nuisance provisions of common law. This nuisance approach is still used in many countries, including the United States and Australia (Schulz and Harreveld, 1996). Usually, it is the local authorities' responsibility to decide whether or not an unacceptable odour impact exists and the degree of the impact (Harreveld et. al.,1999). This method totally depends on personal observations. Most importantly, the explicit conditions that establish whether or not a nuisance condition exists are defined ambiguously and can be interpreted differently by different people. This subjective approach and the relatively costly and time consuming procedures involved often leave the community and the industry dissatisfied with the outcome (Nicell, 1999).

In view of these difficulties, the courts, the community population, industries and regulatory personnel have a need for an objective basis upon which odour impact can be assessed and which can be used as a basis for regulatory decisions (Schulz and Harrevald, 1996). This can only be achieved through a standardized quantitative approach based on sound science. This is currently attempted through the application of the dilution-to-threshold principle.

2.1.1 Dilution-to-threshold principle

The Dilution-to-Threshold (D/T) principle is based on the assumption that a sample of odorous air can be described in terms of the volume to which it must be diluted for its intensity to be reduced to the sensory threshold level. The threshold, $D_{50\%}$, of an odorous gas is the most popular measure of odour concentration and is defined as the dilution (or concentration) of an odour sample at which 50% of a panel of odour judges perceives the odour as being different from odour-free blanks (Nicell, 1999). This is usually accomplished by exposing a group of odour judges to a range of dilutions of the odour sample and interpolating the number of dilutions at which 50% of the group respond to the odour. The methods and techniques of determining thresholds have been discussed in detail by Sikdar (2001). The more dilutions that are required to make an odour sample undetectable, the stronger the sample must be and, thus, is a reflection of odour concentration.

Such concentrations are normally expressed in odour units (OU). One odour unit is defined as the number of dilutions of the odour required to reduce the odour concentration to a level at which 50% of the panel of odour judges is able to detect it. For example, if an odour with a threshold value of $D_{50\%}$ is emitted from a source and dispersed over a neighbourhood, the odour concentration, $C(x, y)$, at any particular location, (x, y), in that neighbourhood can be calculated from:

$$C(x,y) = \frac{D_{50\%}}{D(x,y)} \quad (2.1)$$

where $D_{50\%}$ is the threshold of the odour at the source (expressed in dilutions) and $D(x, y)$ is the number of dilutions of the odour that is achieved between the source and the receptor.

After pointing out drawbacks of odour impact assessments based solely on the dilution-to-threshold principal, as was discussed in Chapter 1, Nicell (1994, 2003) suggested that an alternative approach based on the Odour Impact Model (OIM) should be used.

2.1.2 Odour impact model

The Odour Impact Model (OIM) was developed by Poostchi (1985) and later modified by Nicell (1986, 1994) in which the same techniques used in determining the threshold of an odour are used to establish dose-response relationships. That is, rather than interpolating from the data the number of dilutions at which 50% of the population respond to the odour, the response of the population to the full range of concentration is recorded. This range can extend from as high as the source concentration to as low as the point where the odour is undetectable to all odour judges. As shown in Figure 2.1, this model describes the proportion of the population that responds to the odour and their degree of annoyance as a function of dilutions.

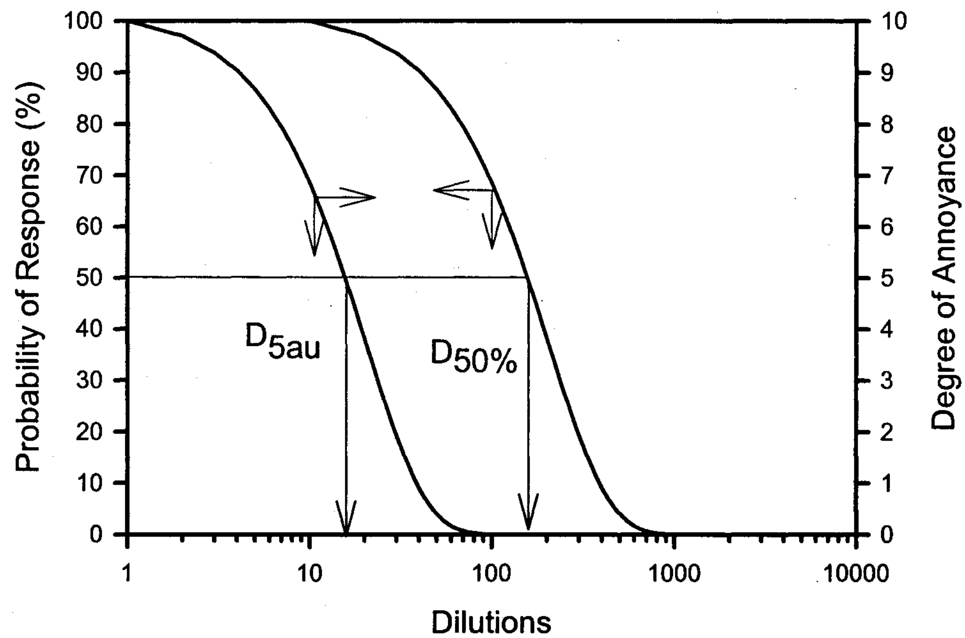


Figure 2.1: Idealized odour impact model

In order to construct the probability of response curve, each panellist is introduced to a series of concentrations of the odour. Usually, these samples are presented in parallel with a number of odour-free blanks and the panellist is asked to attempt to identify the odorous sample. The probability of response is expressed on a scale of 0 to 100% and represents the fraction of the panel of odour judges that were able to distinguish the odour from odour-free blanks at each dilution level, as shown in Figure 2.1. The threshold of the odour ($D_{50\%}$) corresponds to the number of dilutions at which 50% of the population responds to the odour.

A similar procedure is followed for the annoyance curves. At each dilutions level, the degree of annoyance of each panellist is recorded. The degree of annoyance is evaluated on a scale of 0 to10 where panel members are asked to rate their annoyance at each dilution level based on the categories shown in Figure 2.2. The arithmetic means of the annoyance values of all panellists at each dilution level are calculated and are plotted as a function of dilutions, as shown in Figure 2.1. The annoyance threshold, D_{5au}, is the number of dilutions at which the degree of annoyance of the population is 5.

Tolerable	Unpleasant	Very Unpleasant	Terrible	Unbearable
0 to 2	2 to 4	4 to 6	6 to 8	8 to 10

Figure 2.2: Annoyance levels of Nicell (1994)

Nicell (2003) noted that all probability curves have several characteristics: (1) the probability of response approaches 0% at low concentrations; (2) the probability of response approaches 100% at high concentrations; (3) the curve appears to be symmetrical about the point of inflection corresponding to the threshold. An equation that fits with the above characteristics is given by equation 2.2 (Nicell, 2003):

$$P = \frac{100}{1 + (\frac{C_{50\%}}{C})^{\frac{1-p}{p}}} \tag{2.2}$$

where P is probability in %, C and $C_{50\%}$ are the concentrations of the odour (which are inversely proportional to dilutions, D and $D_{50\%}$), and p is called the persistence and is dimensionless. When p approaches 1, the odour tends to be detected by a significant fraction of the population even when C is far lower than $C_{50\%}$. For odours with a p-value near 0, when C is less than $C_{50\%}$, the probability of response of the population quickly drops to zero. Figure 2.3 shows the relationship how the values of persistence influence the probability of response curve. As can be seen in Figure 2.3, the persistence parameter, p, is responsible for the steepness of the curves.

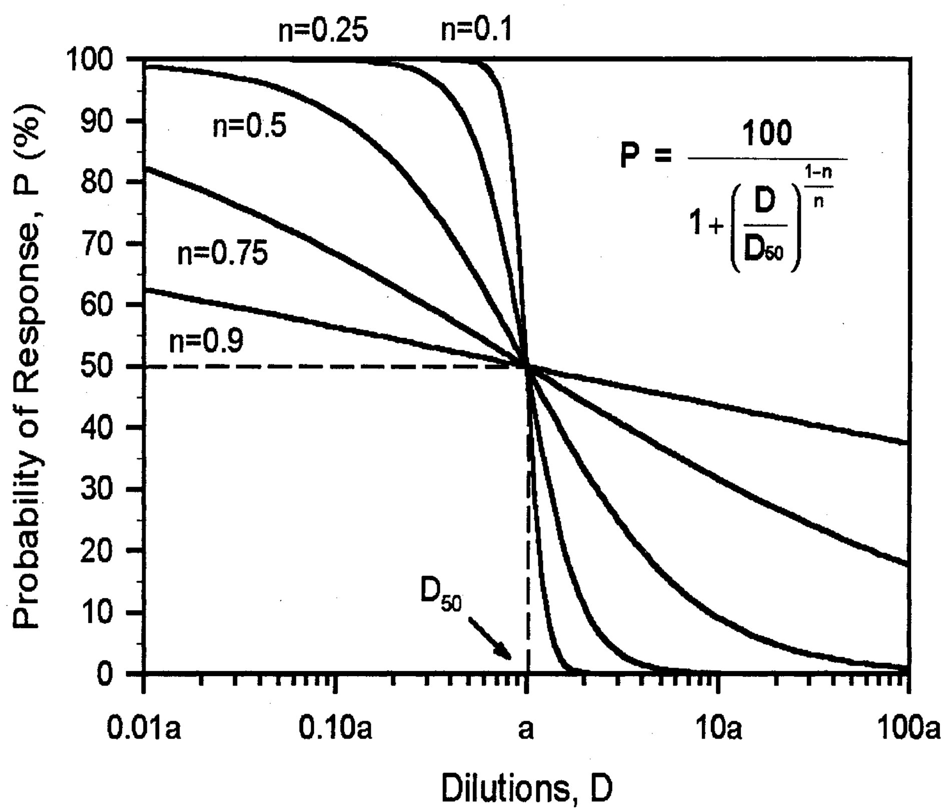


Figure 2.3: Effect of persistence on the probability of response curve

Since concentration is inversely proportional to dilutions, equation 2.2 can be rewritten as:

$$P = \frac{100}{1+(\frac{D}{D_{50\%}})^{\frac{1-p}{p}}} \tag{2.3}$$

Nicell (2003) showed that combining equations 2.3 and 2.1 results in the following equation that relates the probability of response to the odour concentration C_{ou}, expressed in OU:

$$P = \frac{100}{1 + (C_{ou})^{\frac{1-p}{p}}} \tag{2.4}$$

At a specific location (x, y), the probability of response can be expressed by:

$$P(x, y) = \frac{100}{1 + (C_{ou}(x, y))^{\frac{1-p}{p}}} \tag{2.5}$$

where $P\ (x, y)$ is the probability (in %) of people response the odour at the location (x, y) and $C_{ou}\ (x, y)$ is the concentration in OU of the odour at that location.

In addition, the same form of equation can be used to provide a relationship between the degree of annoyance experienced by the panellist (i.e., quantified on a scale of 0 to 10) and concentration or the number of dilutions of the odour (Nicell, 2003):

$$A = \frac{10}{1 + (\frac{C_{5au}}{C})^{\frac{1-a}{a}}} \tag{2.6}$$

or,

$$A = \frac{10}{1 + (\frac{D}{D_{5au}})^{\frac{1-a}{a}}} \tag{2.7}$$

where C_{5au} is the odour concentration at which the population annoyance is 5 au; D_{5au} is the number of dilutions at which the annoyance on a scale of 0 to 10 is equal to 5 au (see Figure 2.1), and a can be interpreted as a measure of the persistence of the annoyance ($0 < a < 1$). Combining equations 2.7 and 2.1 results in the following equation that expresses the annoyance as a function of odour concentration (Nicell, 2003):

$$A = \frac{10}{1 + (C_{ou} \bullet R)^{\frac{a-1}{a}}} \tag{2.8}$$

where R is the ratio of D_{5au} over $D_{50\%}$ or $C_{50\%}$ over C_{5au}. The value of R must always be less than 1 because people must be able to respond to the odour before becoming annoyed (Nicell, 2003). R describes the tendency of an odour to cause annoyance at concentrations relative to its threshold value. Odours with higher R-values would tend to have greater impacts since the population would tend to register annoyance at concentrations that are closer to the threshold.

At a specific location (x, y), the annoyance can be expressed by

$$A(x,y) = \frac{10}{1+(C_{ou}(x,y) \bullet R)^{\frac{a-1}{a}}} \qquad (2.9)$$

where $A(x, y)$ is the annoyance of the odour at the location (x, y).

Several applications of the OIM have been suggested as a means for regulating odours. For example, Nagy (1991) suggested an upper limit of 16% as the allowable percent response of the community at any given location and Poostchi (1985) suggested choosing a maximum allowable degree of annoyance of 2.0 in the surrounding areas.

The Odour Impact Model by itself does not account for all the variables that affect the impact of an odour on a community; e.g., meteorological conditions, stack height and emission conditions, impact of surrounding buildings, etc.). However, it represents a significant improvement over the dilution-to-threshold approach by allowing odour 'quality' and persistence (a measure of the variability in the sensitivity of members of a population to odours) to be incorporated into estimates of the impact of odorous emissions on surrounding communities (Nicell, 1994).

Nicell and Tsakaloyannis (1997) proposed a methodology that combined the OIM approach with a dispersion modelling. Sikdar (2001) and Henshaw et. al. (2002) conducted further research to refine this approach. Figure 2.4 shows the details of this new approach, which involves four parts that can each be broken down into steps:

1. Step 1, 2 and 3 in Figure 2.4: This part involves evaluating the odour impact model for a particular odour sample drawn from a source and then fitting the data to the equations for probability of response and annoyance in order to extract threshold and persistence parameters.

2. Step 4, 5, 6, and 7 in Figure 2.4: This part involves predicting the concentrations of the odour across a grid of receptors. Sikdar (2001) and Henshaw et. al. (2002) used the dispersion model ISC-Aermod that was produced by the USEPA for regulatory purposes. Meteorological data (Aermet), a description of the dimensions of surrounding building (Building profile input program), and source data need to be

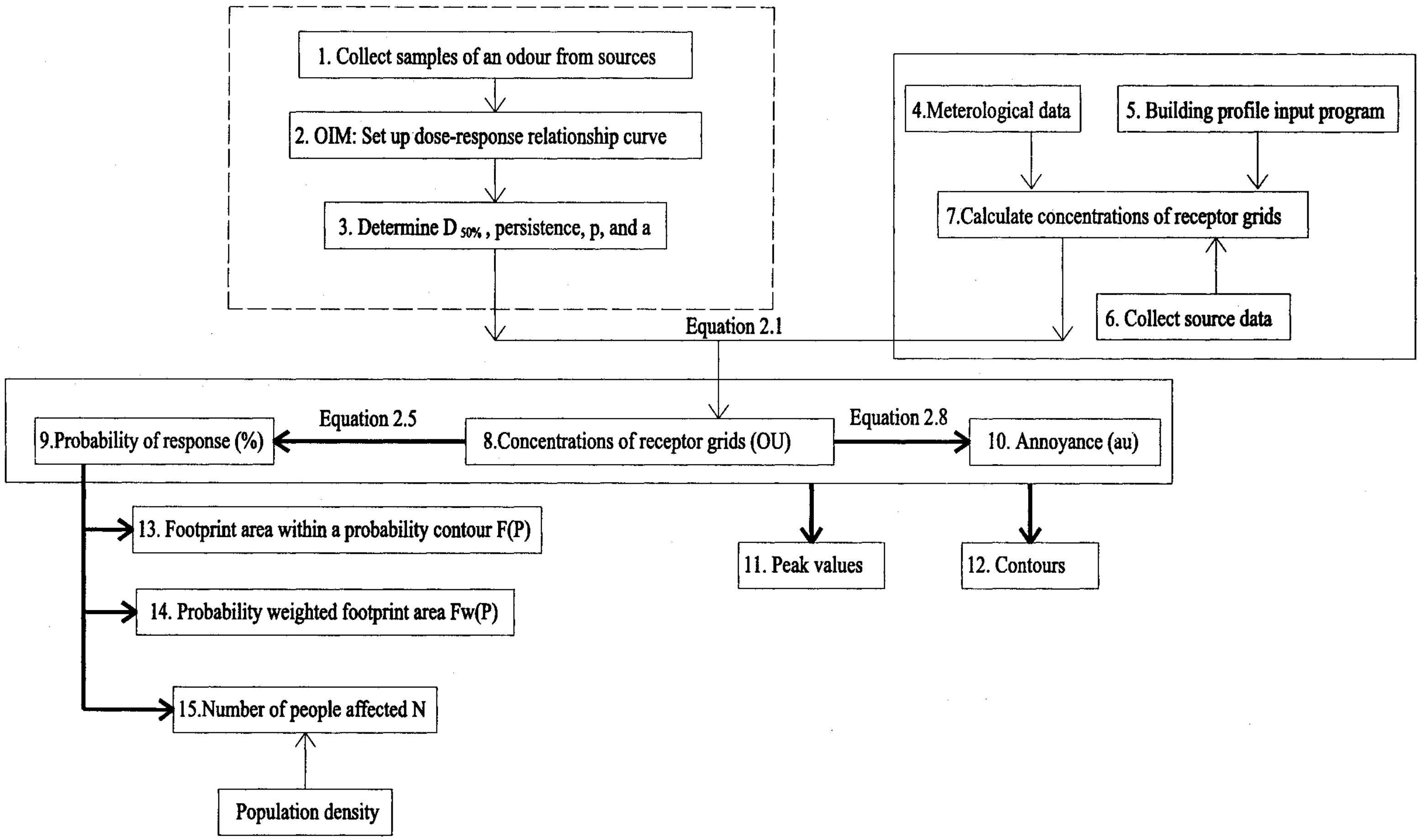


Figure 2.4: Details of method proposed by Henshaw, et. al. (2001)

prepared and entered into the model before running ISC-Aermod. Note that any suitable dispersion model can be used for this purpose.

3. Step 8, 9, and 10 in Figure 2.4: In this part, the concentrations of the odour at each receptor are expressed in odour units (OU) and then transformed into probability of response and annoyance using equations 2.5 and 2.8.

4. Step 11, 12, 13, 14, and 15 in Figure 2.4: The grids of receptor data are then used as a basis for the calculation of several odour impacts. This is done by first developing contours of odour concentration, probability of response and annoyance. Then the information within these contours is analyzed in a variety of ways to evaluate quantitative parameters that are meant to reflect the magnitude of odour impact.

2.2 Impact parameters

Sikdar (2001) and Henshaw et. al. (2002) proposed some parameters to measure the impact of odours. The odour impact parameters can be divided into three categories: namely, point parameters, area parameters, and volume parameters.

2.2.1 Point parameters

Point parameters include the peak concentration, probability and annoyance. The impact of an odour is directly related to its concentration and the frequency at which it is experienced in the community. The higher the concentration and the more frequently it is experienced, the greater will be the odour impact (Sikdar, 2001). The impact is also a function of the fraction of people who experience the odour and the frequency at which this fraction of the population experiences the odour. This fraction corresponds to the probability of response, $P(x, y)$, as determined by equations 2.4 or 2.5. The greater the probability of response, the more people will experience the odour and, thus, the greater will be the odour impact. Finally, the impact of an odour is also a function of the degree of annoyance experienced by the panellist. The annoyance reflects the severity of the odour influence, which is determined by equations 2.8 or 2.9. The greater the degree of annoyance, the greater will be the odour impact.

Peak values of concentration, probability, and annoyance indicate the worst situation that is encountered in the study area.

2.2.2 Area parameters

Plots of the contours of odour concentration, probability of response, and annoyance reveal the spatial extent of odour impact, as can be seen in Figures 2.5 and 2.6.

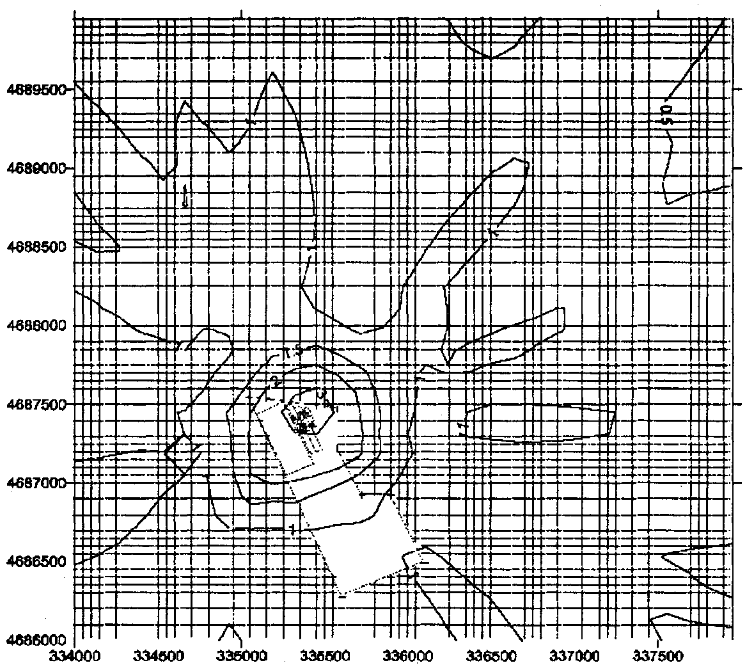


Figure 2.5: Contours for peak hourly concentration (in OU) (Sikdar, 2001)

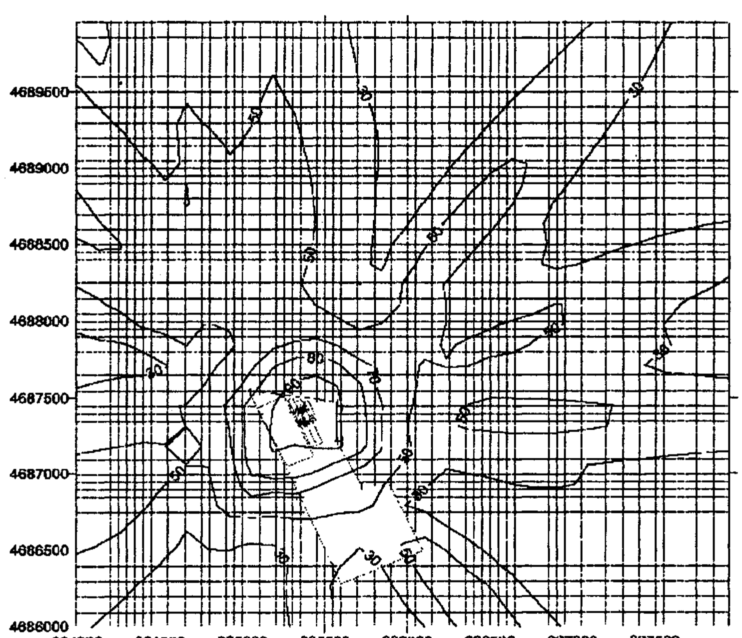


Figure 2.6: Contours for peak hourly probability of response (Sikdar, 2001)

Sikdar (2001) and Henshaw et. al. (2002) proposed that the impact of an odour can be expressed in terms that reflect the geographical size of the region that experiences an odour. Thus, they defined a footprint area, *F(P)*, as the size of the geographical area within a contour corresponding to a specified percentage (*P*) of the population experiencing the odour stimulus. That is, the larger is the area enclosed within a particular contour, the larger is the impact, as illustrated in Figure 2.7. This footprint area can be expressed mathematically as an integral over an area as defined by: (Sikdar, 2001)

$$F(P) = \iint_{R(P)} dxdy \tag{2.10}$$

where *R(P)* represents the region bounded by the probability contour, *P*, and *x* and *y* represent the Cartesian coordinate variables used to describe receptor locations in the study area.

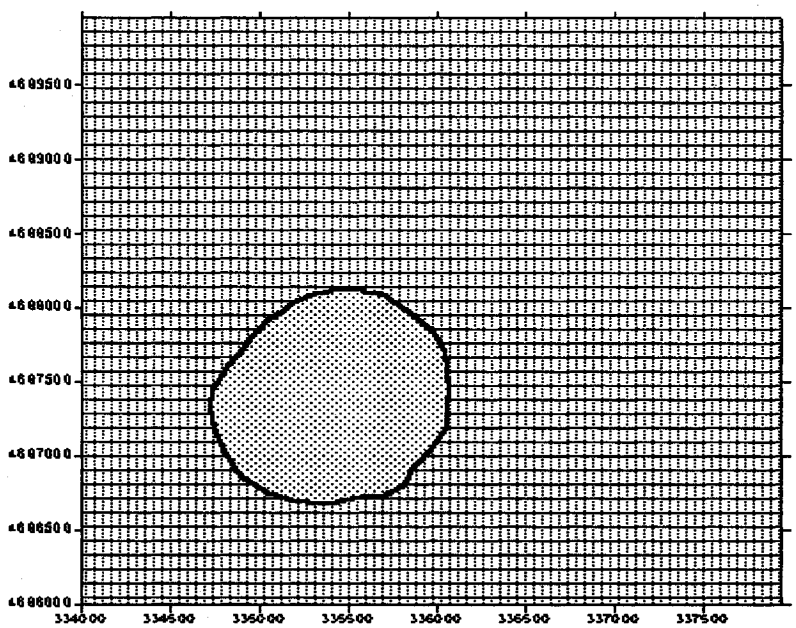


Figure 2.7: Footprint area enclosed by the 10% contour. In this example, the footprint area inside the 10% contour, *F(10%)* is 56.2 km^2

2.2.3 Volume parameters

The footprint area provides an indication of the size of the region in which an odour impact is experienced rather than the magnitude of the impact within that region (Sikdar, 2001). That is, even if two completely different odours have similarly-sized footprint areas enclosed by a particular probability contour, the total fraction of people who experience the odour inside those contours might be quite different. For example, in regions on the fringe of a 10% contour, only 10% of the population would respond to the odour. However, at other locations within this contour, it is possible that 100% of the people would respond to the odour. Thus, it is reasonable to conclude that not only is the size of the footprint region important, but so is the probability of response at every location within that region. Therefore, Sikdar (2001) introduced the concept of a probability weighted footprint area (PWFA), which would sum up the areas enclosed within a contour that experience the odour but would weight each area according to the probability of response within that area. The PWFA can be expressed as:

$$F_W(P) = \frac{1}{100} \iint_{R(P)} P(x, y)\,dx\,dy \qquad (2.11)$$

where $F_W(P)$ is the weighted footprint area inside a probability contour, P; $R(P)$ represents the region bounded by probability contour, P; $P(x, y)$ is the probability of response (in %) at each receptor. As shown in Figure 2.8, $F_W(P)$ is the volume of a space that extends vertically from a lower limit of 0 to an upper limit of $P(x, y)$ and which is limited in its horizontal direction by a selected contour P. The overall volume is the sum of volumes V1 and V2, shown in Figure 2.8. In this particular case, the PWFA is shown for a region enclosed by the 50% contour.

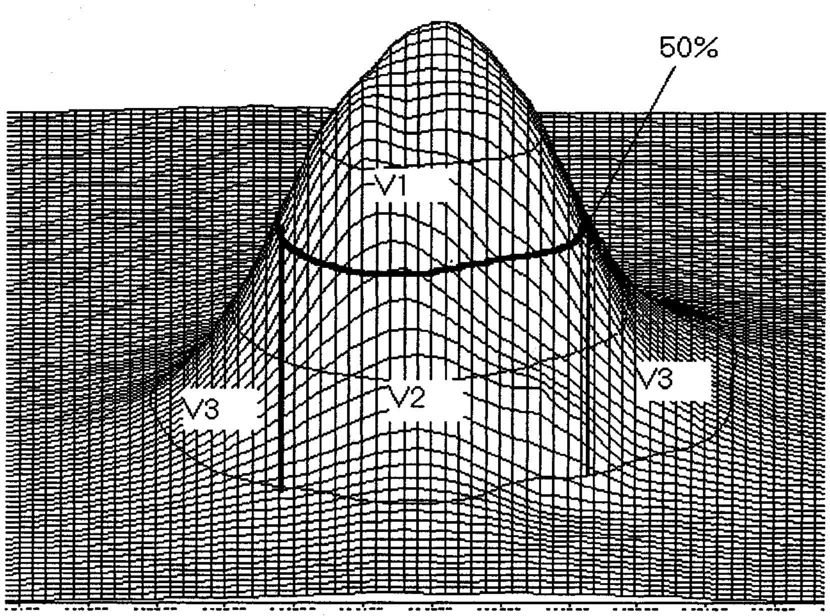


Figure 2.8: Geometrical meaning of $F_W(P)$

However, over an infinite grid, the effect of the odour will be negligible at any point distant from the source, as the odour becomes diluted to levels well below the odour threshold. Henshaw et al (2002), suggested that it might be more reasonable not to restrict the calculation of this impact parameter to the area inside any particular contour, but instead calculate it over the entire modelled area. This can be expressed by:

$$F_{WP} = \frac{1}{100}\iint P(x,y)dxdy \tag{2.12}$$

In Figure 2.8, F_{WP} would be the entire volume under the 3-dimensional curve. In performing actual dispersion modelling studies, the modellers must always define a study area, R. The total PWFA in this area can be defined as:

$$F_{WP}(T) = \frac{1}{100}\iint_R P(x,y)dxdy \tag{2.13}$$

Only if the probabilities of response at the receptors on the boundary of the study region are all 0 will the value of $F_{WP}(T)$ be equal to the true F_{WP}.

2.2.4 Number of people affected (NPA)

The impact of an odour may also be assessed in terms of the size of the population that is impacted by an odorous emission (Sikdar, 2001). That is, one might argue that the impact of an odorous emission would be negligible if there is no population in the area that is exposed to the odour. Thus, the greater the population in an area, the greater will be the impact. The population, $N_W(P)$, that has more than P percent probability to respond to the odour can be assessed using the following equation:

$$N_W(P) = \iint_{R(P)} \frac{P(x,y) \times N_D(x,y)}{100} dxdy \tag{2.14}$$

where $N_W(P)$ is the number of people affected in the region bounded by the probability contour, P. $P(x, y)$ is the probability of response at each receptor and $N_D(x, y)$ is the population density at each receptor (persons/unit area).

The total number of people affected in the region R can be expressed by:

$$N_{WP}(T) = \iint_R \frac{P(x,y) \times N_D(x,y)}{100} dxdy \tag{2.15}$$

or, if the grid cell is rectangular:

$$N_{WP}(T) = \sum \frac{P(x,y) \times N_D(x,y)}{100} \Delta x \Delta y \tag{2.16}$$

where Δx and Δy are the dimensions of each grid cell.

This method can be varied to also ignore populations that are included within the plant boundaries of an offending source. In this way, only the population beyond the plant boundaries would be considered in the impact assessment.

3 Method

The main tools of this research were Visual Basic and Surfer 7.0. Visual Basic was used to develop the software OdorImp. Surfer 7.0 was used to examine and evaluate the accuracy of the results arising from OdorImp.

3.1 Visual Basic

The choice of a computer language for a particular task is made depending upon the adaptive ability of the computer language to the task. Microsoft Windows is the most popular operating system that is used in personal computers. Among numerous computer languages, Visual Basic is one of the fastest and simplest that may be used to develop standard Windows applications. Visual Basic was developed in 1991 by the Microsoft Corporation, which provides the Graphical User Interface (GUI). GUI means that a program communicates with users by showing graphics and other standard objects. Users respond to those graphics and objects by clicking a mouse or using a keyboard. (Duffy, 1995)

Most application programs are composed of a series of executive steps, and include three stages: input, processing, and output (Duffy, 1995). Terminal users don't need to understand the codes of the programs in detail. Therefore, creating simple, clear, effective user interfaces for input and output is critical during programming. Visual Basic helps programmers decide on the interface for users, determine which events the objects on the window should recognize, and write the event procedures for those events (Schneider, 1995).

Visual Basic was chosen for use in the current work for a variety of reasons. Firstly, the ultimate purpose of this work is to provide practitioners with the means of conducting odour impact analyses with a minimal amount of training. Thus an intuitive graphical interface that is consistent in format to other applications is desired. Interfaces developed with Visual Basic can be designed to be very consistent with other commonly-used Windows applications, which should make it easer to learn how to use the application. Secondly, it would be desirable to ensure that the routines that are developed during this research could easily be incorporated into other software. A package named OdorCalc has already been developed in Visual Basic to provide the means for evaluating odour parameters such as thresholds and persistence values. Eventually, it is expected that the program arising from the current work will be integrated with OdorCalc, and thus a common language would facilitate the merging of these programs.

3.2 Surfer 7.0

Surfer is a powerful, flexible and easy-to-use contouring and 3D surface mapping package. Surfer can easily and accurately transform XYZ data into contour, wireframe, shaded relief, image, post, and vector maps. In this work, as described below, the functions of Surfer were used for creating contours and performing calculations on grid files of receptor data. (Golden Software Inc., 1999)

3.2.1 Drawing contours

The input data for Surfer must be contained in either a text file or an Excel file containing *X*, *Y* and *Z* coordinates of grids values (Golden Software Inc., 1999). The first step after inputting these grid coordinates is to generate a grid data file (*.grd). Surfer has several methods for spatial interpolation: inverse distance to a power, kriging, minimum curvature, modified Shepard's method, natural neighbour, nearest neighbour, etc. The purpose of spatial interpolation is to predict unknown values from data observed at known locations. Kriging was chosen for this work because it is a method that minimizes the error of predicted values, which are estimated by spatial distribution of known values (Chao, 2002). Kriging is a method that is associated with the acronym BLUE (i.e., best linear unbiased estimator). It is "linear" since the estimated values are weighted linear combinations of the available data. It is "unbiased" because the mean of error is 0. It is "best" since it aims at minimizing the variance of the errors. The difference between kriging and other linear estimation method is its aim to minimize the error variance (Chao, 2002).

Contours of grid values were created using the *Contour Map* option. Through use of the *Level* and the *Label* menus under the contour map properties menu, the manner in which the contours were displayed was specified. In the *Level* menu the *Start Levels*, *End Levels* and the *Interval* of Contours were set. Using the *Label* option, the Contours to be labelled, the nature of the line, and the type of fill of the contours were specified.

3.2.2 Calculations of volume and area

In Surfer, the volume under a grid is calculated using three methods: Trapezoidal Rule, Simpson's Rule, and Simpson's 3/8 Rule. The difference in the volume calculations by the three different methods measures the accuracy of the volume calculations. If the three volume calculations are reasonably close, the true volume is close to these values. If the three values differ somewhat, a new denser grid file should be used before performing the volume calculations again.

The Positive Volume (Cut) is the volume of material in those places where the upper surface is above the lower surface. The Negative Volume (Fill) is the volume of material in those places where the upper surface is below the lower surface. Figure 3.1 shows the concepts.

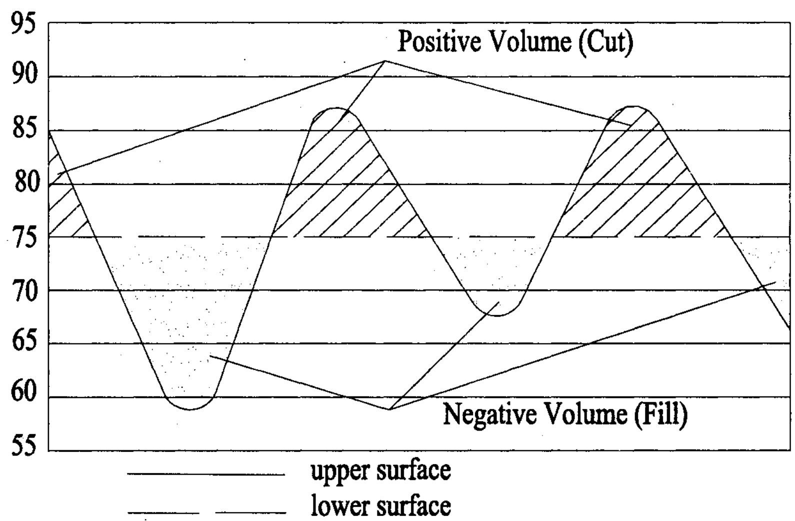


Figure 3.1: Cross-section showing the relation between the upper and lower surfaces and the cut and fill volumes. The lower surface is defined by *Z*=75.
(Golden Software Inc., 1999)

In Figure 3.1, the positive volume is the same as V1 in Figure 2.8. However, the Probability Weighted Footprint Area (PWFA) described earlier is the sum of V1, V2, and/or V3 in Figure 2.8. Areas that are calculated in Surfer are in terms of planar areas and surface areas. The Positive Planar Area represents the planar area where the upper surface is above the lower surface. The Negative Planar Area represents the planar area where the upper surface is below the lower surface. The Blanked Planar Area is the sum of the areas over the blanked regions on both the upper and lower surfaces. The Total Planar Area represents the planar area for the entire grid.

In order to use the Grid | Volume command to calculate net volumes, cut and fill volumes, planar areas, and surface areas, the following steps are required:

1. Use the Grid | Volume command to display the Open Grid dialog box.
2. Specify the name of the grid file to use in the volume and area calculations. This can be the grid file for either the upper or the lower surface.
3. Click OK and the Grid Volume dialog box is displayed. The specified grid file is shown for both the upper and lower surface.
4. Specify the Upper Surface and Lower Surface parameters and click OK.
5. The Grid File option is used to specify a grid to use as the upper or lower surface. To use the grid file displayed, activate the Grid File option. To change the grid file for either the upper or lower surface, click the Grid File option and then click the open file icon to select another grid file. The Constant option is used to specify the level of the planar surface to use as the upper or lower surface. Specify the level of the planar surface by entering the value into the Z = edit box. The specified value is in Z data units.
6. Click OK in the Grid Volume dialog box and the results are displayed in the Grid Volume Report.

3.3 Odour data

In order to test the algorithms that were developed during this study, odour data was required for input. Initially, synthetic odour data was generated. This data was generated to provide an ideal set of data that could be used for evaluation purposes. For example, algorithms were developed (see Chapter 4) to evaluate footprint areas and volumes under curves. In order to check the accuracy of these algorithms, synthetic data were generated for which exact values of areas and volumes could be calculated. In addition, these synthetic data were used to test the ability of the algorithms for comparison with Surfer 7.0.

After testing with synthetic data, the abilities of the algorithms were tested using real odour impact data for comparison with Surfer. These data were non-ideal (with grids of irregular surfaces) and were generated using dispersion modelling results from several field studies.

3.3.1 Synthetic data

Synthetic data were generated using a simple Gaussian dispersion model and also based on perfect geometrical shapes including a half sphere and a taper.

(1) Simple Gaussian dispersion model

A simple Gaussian dispersion model that can be used to predict the ground level concentration of a gaseous contaminate at a location (x, y) is as follows (Beychok, 1994):

$$C(x,y) = \frac{M}{\pi u \sigma_z \sigma_y} e^{-\frac{y^2}{2\sigma_y^2}} e^{-\frac{H^2}{2\sigma_z^2}} \quad (3.1)$$

Where *C (x, y)* = concentration at a point x, y (mass/volume or OU)

x = distance directly downwind from source (distance)

y = perpendicular (crosswind) distance from source (distance)

M = emission rate of pollutant from source (mass/time or OU · volume/time)

u = horizontal wind velocity (distance/time)

H = plume release height above ground (distance)

σ_z = vertical dispersion parameter (distance)

σ_y = horizontal dispersion parameter (distance)

The value of σ_z and σ_y are both a function of downwind distance (x) and are often calculated using the following empirical correlation of dispersion with distance:

$$\sigma = e^{a+b(\ln x)+c(\ln x)^2} \quad (3.2)$$

for which the particular coefficients a, b, and c that correspond to functions for σ_z and σ_y are available for different conditions of atmospheric turbulence (Beychok, 1994).

Default values for these parameters used in this study were: M = 230 00 000 OU· m^3/s; u = 4 m/s; H = 12.3 m; a = 4.694, b = 1.0629 and c = 0.0136 for σ_z; and a = 5.058, b = 0.9024 and c = -0.0096 for σ_y. These values result in a grid such as the one shown in

Figure 3.2. Note that this model was also incorporated into OdorImp to provide users with a tool that can serve as a basis for practicing odour impact assessments. . The code for generating data from the simple Gaussian model is shown in Appendix 1.1.

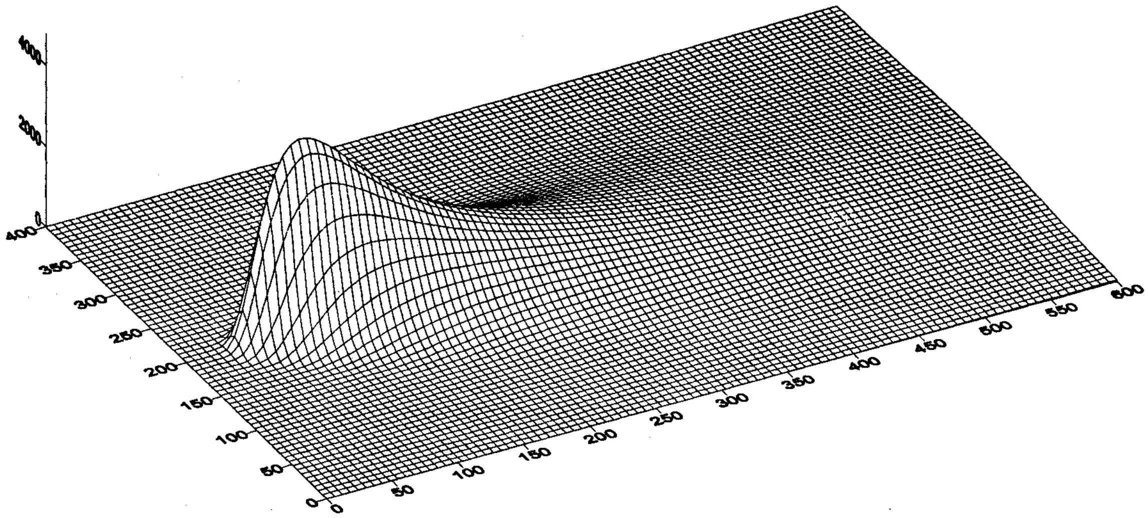


Figure 3.2: 3D profile from the simple Gaussian model

(2) Selected geometries (half sphere and taper)

The radius of the half-sphere was 1000 m and the radius of the bottom surface of the taper was 1000 m. Both had a height of 1000 m. Figures 3.3 and 3.4 show their 3D profiles, as produced by Surfer. Codes for generating the half sphere and taper are shown in Appendices 1.2 and 1.3.

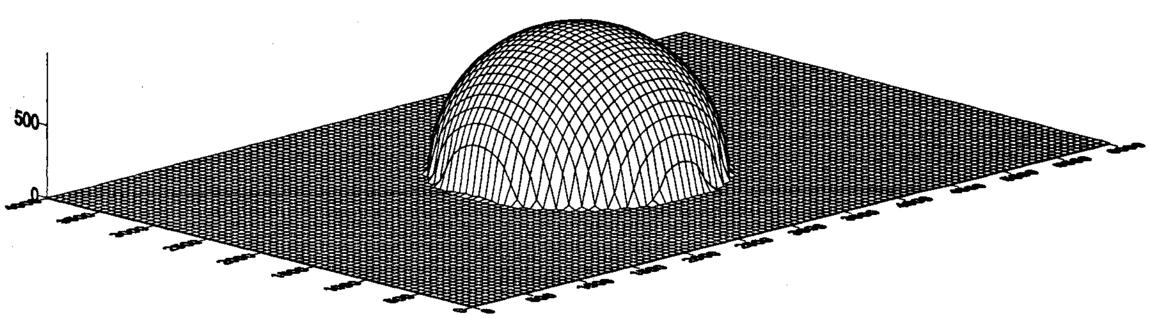


Figure 3.3: 3D profile for the half sphere

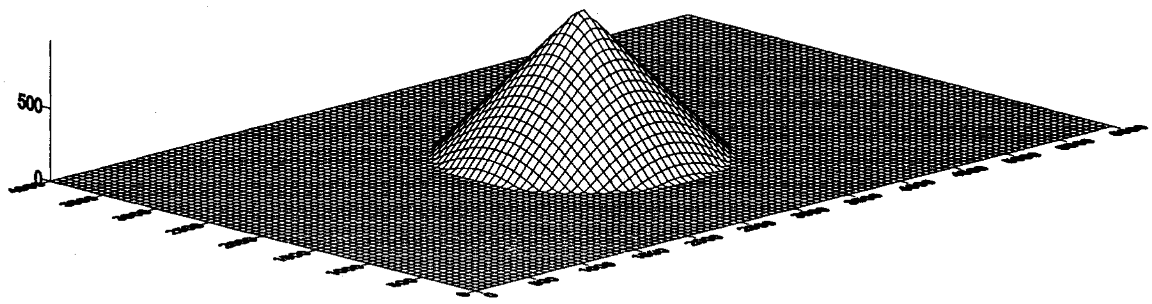


Figure 3.4: 3D profile for the taper

3.3.2 Real odour data

Odour data was generated through the Lakes Environmental (Waterloo, Ontario) software interface for the Industrial Source Complex Aermod (ISC-Aermod) dispersion model developed by the USEPA. Concentrations of odour for the receptor grid from ISC-Aermod were recorded in odour units (OU).

Three input files are needed to run ISC-Aermod. Two of them are outputs from Aermet (*.pfl and *.sfc), which deals with the meteorological data. The other one is an output from the BPIP (*.bpo) module, which deals with the building profiles and other characteristics for sources and calculates the building downwash (Lake Environmental Inc., 2000).

ISC-Aermod is run in either the urban or rural modes for dispersion coefficient calculations according to the particular case study being modelled. The output file (*.plt) of ISC-Aermod contains the X and Y coordinates, average concentrations, elevations, averaging time, etc. The format of the input file for OdorImp is a text file that is modified from *.plt and only keeps the x and y coordinates and the average concentrations. The manner in which these files can be produced will be discussed in detail in the following sections.

In this research, two sets of dispersion model predictions based on real field data measurements were generated by Gorgy (2003) and Sikdar (2001) using ISC-Aermod. Gorgy (2003) used a pig farm located in Quebec, Canada for his study. At this pig farm, 24 fans were used to ventilate the livestock house. Gorgy modelled these fans as separate point sources and generated a 50 × 50 grid of receptors in a 1091 m × 987 m study area. He predicted the 1-hour time-averaged (short term) concentrations in OU of each grid receptor. The output file containing the concentration, X and Y coordinates of grid receptors became the input file for OdorImp.

The other case involved a study of an industrial facility with 18 stacks which is located in South Western Ontario, Canada (Sikdar, 2001). The study area was 16 km^2 (4 km × 4 km). The uniform distance between adjacent grids points was 0.05 km. ISC-Aermod can be set to record the peak hourly value of the concentration at each receptor. The percentile plots can be generated by selecting an option of Aermod, such as a 99th percentile which represents the frequency of occurrence of the values at each grid that would be exceeded 1% of the time. Four sets of data which contain the peak hourly concentration, the 90th percentile, the 95th percentile and 99th percentile concentrations in OU and coordinates, respectively, were generated by Sikdar (2001).

4 Development of Odour Impact Parameters

Based on the approach used by Sikdar (2001) and Henshaw et. al. (2002) to propose odour impact parameters, additional parameters were developed in this research, as described below. All of the proposed parameters are listed in Table 4.1.

It should be noted that the evaluation of the relative usefulness of the various parameters is beyond the scope of the present work. The purpose here is to develop algorithms that can be used as tools to evaluate an extensive range of potential parameters that have merit for odour impact assessment. It is left to future researchers to apply the impact parameters developed in the following sections to field studies in order to evaluate their ability to reflect community impact.

4.1 Point parameters

Peak values may be used to express the worst-case manner in which odours are experienced at particular receptors in the community. As suggested by Sikdar (2001), peak values of concentration (C_{max}), probability (P_{max}), and annoyance (A_{max}) reflect the worst odour impacts at which the highest concentrations, responses and annoyances are experienced by the population.

It is also conceivable that the impact could be considered worse when there are more people at a given location. Therefore, additional peak parameters can be proposed in which the concentrations, probability of response, and annoyances are weighted according to the number of people at a given receptor. At any given receptor, the population-weighted concentration (*NC*), the population-weighted probability (*NP*) and the population-weighted annoyance (*NA*) may be calculated from:

$$NC = N_D(x, y) \times C(x, y) \tag{4.1}$$

$$NP = N_D(x, y) \times P(x, y) \tag{4.2}$$

$$NA = N_D(x, y) \times A(x, y) \tag{4.3}$$

where $C(x, y)$, $P(x, y)$ and $A(x, y)$ are the concentration (in OU), probability (in %) and annoyance (in au) values at each receptor location and $N_D(x, y)$ is the population density (capita/m^2) at those locations. Once these values have been calculated across an entire

Table 4.1: Proposed odour impact parameters

Parameters	Symbol	Units	Description
Point parameters	C_{max} P_{max} A_{max}	OU % au	Peak odour response experienced at a particular receptor
	NC_{max} NP_{max} NA_{max}	OU·capita/m^2 capita/m^2 au·capita/m^2	Peak odour response experienced at a particular receptor weighted according to the population impacted
Area parameters	$F(C)$ $F(P)$ $F(A)$	m^2 m^2 m^2	Footprint area bounded by selected contours of *C*, *P* or *A*
Volume parameters	$F_W(C)$ $F_W(P)$ $F_W(A)$	OU·m^2 m^2 au·m^2	Weighted footprint in areas bounded by selected contours of *C*, *P* or *A*
	$F_{WC}(T)$ $F_{WP}(T)$ $F_{WA}(T)$	OU·m^2 m^2 au·m^2	Weighted footprint area bounded by the selected study area (i.e., area for which dispersion modelling has been performed)
	F_{WC} F_{WP} F_{WA}	OU·m^2 m^2 au·m^2	Weighted footprint areas that are unbounded (i.e., footprint encompasses the entire spatial extent over which the odour is dispersed)
	$N(C)$ $N(P)$ $N(A)$	capita capita capita	Population impact within an areas bounded by selected contours of *C*, *P* or *A*
	$N_W(C)$ $N_W(P)$ $N_W(A)$	OU·capita capita au·capita	Weighted population impact within an area bounded by selected contour of *C*, *P* or *A*
	$N_{WC}(T)$ $N_{WP}(T)$ $N_{WA}(T)$	OU·capita capita au·capita	Weighted population impact within an area bounded by the selected study area (i.e., area for which dispersion modelling has been performed)
	N_{WC} N_{WP} N_{WA}	OU·capita capita au·capita	Weighted population impact in an unbounded area (i.e., population impact in the entire region in which the odour is dispersed)

N/A = not applicable

grid of receptors, the worst case values may be found. Thus, the peak parameters, NC_{max}, NP_{max}, and NA_{max} are suggested as potential impact parameters.

4.2 Area parameters

Sikdar (2001) proposed using the footprint area, *F(P)*, contained within a specific probability contour as a measure of odour impact (see equation 2.10). This approach attempts to quantify the magnitude of the odour impact in terms of the size of the region that is being impacted upon. The same approach can be applied for the measurement of footprint areas inside concentration and annoyance contours; i.e., *F(C)* and *F(A)*, footprint areas with units of area. These footprint areas can be expressed as follows:

$$F(P) = \iint_{R(P)} dxdy \tag{2.10}$$

$$F(C) = \iint_{R(C)} dxdy \tag{4.4}$$

$$F(A) = \iint_{R(A)} dxdy \tag{4.5}$$

where *R(P), R(C)* and *R(A)* represents the regions bounded by the probability contour *P* (%), concentration contour *C* (OU), and annoyance contour *A* (au), respectively. In each case, the greater is the footprint area, the greater the odour impact would be.

4.3 Volume parameters

4.3.1 Probability-, concentration-, and annoyance-weighted footprint areas

Sikdar (2001) proposed using the probability weighted footprint area (PWFA) (see equation 2.11) to measure the odour impact.

$$F_W(P) = \frac{1}{100} \iint_{R(P)} P(x,y)dxdy \tag{2.11}$$

This concept can be expanded to the concentration and annoyance. Within a specific concentration contour, the footprint area can be expressed as:

$$F_W(C) = \iint_{R(C)} C(x,y)dxdy \tag{4.6}$$

where $F_W(C)$ is the concentration weighted footprint area (CWFA) of a region *R(C)* which is bounded by a concentration contour, *C*; and *C(x, y)* is the concentration at each receptor.

Within a specific annoyance contour, the footprint area can be expressed as:

$$F_W(A) = \iint_{R(A)} A(x,y)dxdy \tag{4.7}$$

where $F_W(A)$ is the annoyance weighted footprint area (AWFA) of a region $R(A)$ which is bounded by an annoyance contour, A; and $A\ (x, y)$ is the annoyance at each receptor.

In a study area, such as a town or a city over which the dispersion of odours is modelled, the total concentration, probability, and annoyance weighted footprint areas can be expressed by:

$$F_{WP}(T) = \frac{1}{100}\iint_R P(x,y)dxdy \tag{2.13}$$

$$F_{WC}(T) = \iint_R C(x,y)dxdy \tag{4.8}$$

$$F_{WA}(T) = \iint_R A(x,y)dxdy \tag{4.9}$$

where R represents the entire study region and $P(x, y)$, $C(x, y)$, and $A(x, y)$ are the probability (in %), concentration (in OU) and annoyance (in au) at each receptor respectively.

In an infinite area (unbounded area):

$$F_{WP} = \frac{1}{100}\iint P(x,y)dxdy \tag{2.12}$$

$$F_{WC} = \iint C(x,y)dxdy \tag{4.10}$$

$$F_{WA} = \iint A(x,y)dxdy \tag{4.11}$$

This type of parameters reflects the magnitude of the impact over the entire extent of the space over which an odour can be dispersed. The units of the PWFA, the CWFA and the AWFA are m^2, $OU{\cdot}m^2$ and $au{\cdot}m^2$, respectively.

4.3.2 Population in concentration-, probability-, and annoyance-contours

The measure of the number of people within a specific contour is a potential impact parameter. For example, if a local regulatory agency sets 1 OU as an upper limit for odour concentration in a neighbourhood, it may be argued that the situation of an area with 10,000 people who suffer more than 1 OU of odour is more serious than an area with 5,000 people who suffer more than 1 OU of odour. Alternatively, bounds of probability of response, P, and annoyance, A, could be chosen as the basis for population calculations. Mathematically, these population impact parameters can be represented by:

$$N(C) = \iint_{R(C)} N_D(x,y)dxdy \quad (4.12)$$

$$N(P) = \iint_{R(P)} N_D(x,y)dxdy \quad (4.13)$$

$$N(A) = \iint_{R(A)} N_D(x,y)dxdy \quad (4.14)$$

where *N(C)* is the number of people within an area which is bounded by a concentration contour, *C*; *N(P)* is the number of people within an area bounded by a probability contour, *P*; *N(A)* is the number of people within an area which is bounded by an annoyance contour, *A*; and $N_D(x, y)$ is the population density at each receptor location (capita/unit area).

4.3.3 Concentration-, probability-, and annoyance-weighted populations

Sikdar (2001) proposed calculating the number of people affected in a region by multiplying the probability of response at any given receptor by the local population density and then summing over a selected region. This produces a population estimate that has been weighted according to the fraction of persons that would actually respond to the odour and is expressed by:

$$N_W(P) = \iint_{R(P)} \frac{P(x,y) \times N_D(x,y)}{100} dxdy \quad (2.14)$$

or across an infinite area as:

$$N_{WP} = \iint \frac{P(x,y) \times N_D(x,y)}{100} dxdy \quad (4.15)$$

Similarly, the number of people affected can also be weighted according to the average concentration experienced by the population at each receptor. This concentration-weighted population may be expressed as:

$$N_W(C) = \iint_{R(C)} C(x,y) \times N_D(x,y)dxdy \quad (4.16)$$

where $N_W(C)$ (in OU·capita) is the concentration-weighted population in a region bounded by a concentration contour *C*; *C(x, y)* is the concentration at each receptor; and $N_D(x, y)$ is the population density at each receptor. This value may also be calculated over the entire study region, *R*, rather than just within a selected contour, as follows:

$$N_{WC}(T) = \iint_R C(x,y) \times N_D(x,y)dxdy \quad (4.17)$$

or across an infinite area as:

$$N_{WC} = \int\int C(x,y) \times N_D(x,y)dxdy \tag{4.18}$$

Similarly, the population can also be weighted according to the average annoyance experienced at any given receptor. This can be expressed as follows in a region bounded by a particular annoyance contour, A:

$$N_W(A) = \int\int_{R(A)} A(x,y) \times N_D(x,y)dxdy \tag{4.19}$$

where $N_W(A)$ (in au·capita) is the annoyance-weighted population in the region bounded by the annoyance contour, A and $A(x, y)$ is the annoyance at each receptor.

The annoyance-weighted population (in au·capita) that is contained within a selected study area, R, may be expressed as:

$$N_{WA}(T) = \int\int_{R} A(x,y) \times N_D(x,y)dxdy \tag{4.20}$$

or across an infinite area as:

$$N_{WA} = \int\int A(x,y) \times N_D(x,y)dxdy \tag{4.21}$$

5 Development

In this chapter, algorithms that were developed for drawing contours, calculating footprint areas, probability-, annoyance- and concentration-weighted footprint areas, and population impact parameters are discussed in detail. Note that the nomenclature for all impact parameters discussed here is contained in Table 4.1.

5.1 Algorithm for drawing contours

A contour plot is a set of level curves of different heights of a function of two variables, usually expressed in x and y coordinates. A contour can be expressed as a level curve of height h of a function f (x, y), i.e. f (x, y) = h (Aramini, 1981). There are two basic algorithms, which are referred to as the level curve tracing algorithm and recursive subdivision algorithm (Aramini, 1981). Each method based on the two algorithms has its own advantages and disadvantages.

The level curve tracing algorithm is a direct way to do contour plotting in the case where the Z values are available only at vertices of a rectangular grid. The principle that judges if an edge is intersected by a contour is that the contour value, h, is between the values at the two nodes of the edge, a and b, ($a < h < b$). There are some variations of algorithm (Aramini, 1981).

The algorithm presented by Snyder (1978) may be summarized as follows (Rand, 2002):

a) Given matrixes of X and Y coordinates and Z values of the grids.
b) The program traces each node to look for any line segment, which must be crossed by a contour because some contour value lies between the values of Z at the nodes.
c) Having found such a segment, the program calculates the intersection point of the contour and the segment by linear interpolation between the nodes. It also stores the information that the current contour value has been located on the current segment, so that this operation will not be repeated.

d) The program then attempts to locate a neighbouring segment if it is crossed by the same contour. If it finds one, it determines the intersection points as in step c) and then draws a straight line segment between the previous intersection point and the current one. This step is repeated until no such neighbour can be found, taking care to exclude any segment, which has already been dealt with.
e) Steps b), c), and d) are repeated until no segment can be found whose intersection with any contour value has not already been processed.

The algorithm of Cottafava and Le Moli (1969) is different from the one of Snyder. The general approach is, first, to search all the intersection points between edges of the grid in any inspection order and then to reorder them in a fixed direction. This algorithm can be summarized as:

a) Find all intersected edges and use a Boolean variable to mark them. Arbitrarily add an infinitesimal value to the grids which are exactly on the contour;
b) The order of EAST, SOUTH, WEST, and NORTH is chosen arbitrarily;
c) Search a starting point of a branch of the contour. This is accomplished by scanning all the edges in the fixed order until a stored intersection is found.
d) Follow the branch by searching which edge of the element has a stored intersection in the order. When a stored intersection is found, then cancel it from storage to avoid meeting it again. Actually, a contour must meet the boundary or one vertical edge at least once. If an element is intersected four times by the contour, the following situations may happen (Figure 5.1) according to the entry edge and the order of EAST, SOUTH, WEST, and NORTH. For situation (a), the entry edge is EAST or South; situation (b), the entry edge is West; situation (c), the entry edge is North;
e) The intersection coordinates are calculated by a linear interpolation, and the contour line is drawn;
f) The analysis continues for the cell adjacent to the intersected edge by repeating the same procedure from step c;
g) A contour stops when no intersection is found in this branch; otherwise, the contour stops on the boundary.
h) Repeat step c to step f until no intersected edge is found.

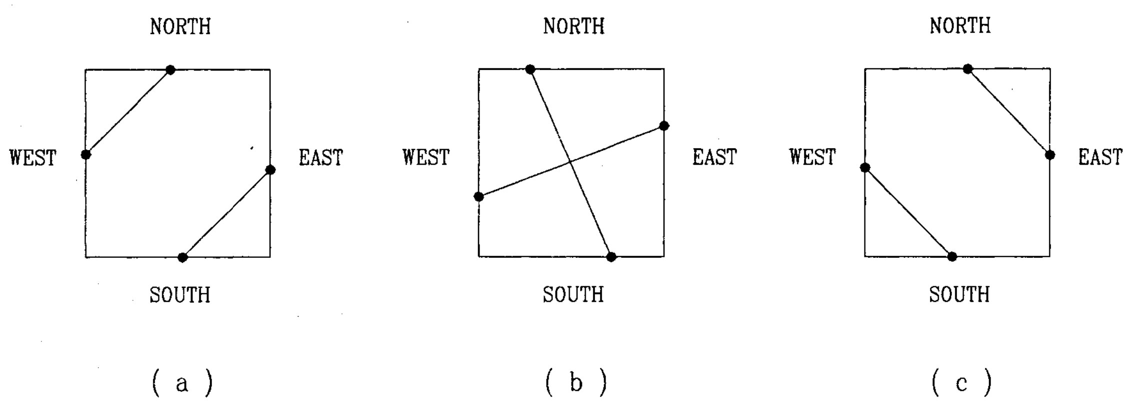


Figure 5.1: Three situations of intersections (Cottafava and Le Moli, 1969)

Algorithms presented by Snyder and Cottafava and Le Moli are very typical level curve tracing algorithms. Moreover, Aramini (1981), Wright (Aramini, 1981), and Karney (Aramini, 1981) modified the algorithms in various ways in order to reduce using computer storage or to use triangular grids instead of rectangular grids, etc. However, there are some deficiencies that this kind of algorithm cannot conquer (Aramini, 1981). The most important one is the ambiguity associated with cells in which all four edges are intersected by the same level curve. In Cottafava and Le Moli's algorithm, this is overcome by setting a fixed order.

According to the recursive subdivision algorithm, the region to be plotted should be first divided into an initial coarse grid (Aramini, 1981). For each cell in this initial grid, a test for whether the minimum of the Z values at each of the four nodes is greater than or equal to the contour value or not, is checked for each contour. If this test is met then the cell is divided into four equally sized subcells. This test is repeated for each of the subcells recursively until this test fails or some minimal cell size is reached. If the test is satisfied, the point in the centre of the cell is considered to be on the level curve. Sometimes the contour obtained by this algorithm can be a set of discrete points rather than a connected set of line segments.

After studying the algorithms discussed above, the algorithm presented by Cottafava and Le Moli (1969) was selected. The main reasons are: firstly, the grids of the OdorImp are rectangular and similar to the one of Cottafava and Le Moli; secondly, this algorithm saves computer storage; thirdly, it is simpler as it only has to deal with contours intersecting edges instead of having to deal with the intersections and vertices; and, lastly, it is simpler to develop computer code for this algorithm.

Table 5.1 details how a starting point is found using the selected algorithm and Table 5.2 shows how the contour is drawn once a starting point is found. The order of North, East, South, and West was chosen in OdorImp. The computer code for drawing contours is shown in Appendix 2.1.

Table 5.1: Algorithm for finding a starting point

Scan each horizontal and vertical edge. If an intersection point is found, then set the value of the corresponding element in an array to 1, and record its coordinates at the same time.
Create a four-dimensional array to record the series number of edges for each element in the order of NORTH, EAST, SOUTH, and WEST.
Scan every edge to search for a starting point of the contour, as follows: 1. Determine to which element the edge belongs. 2. Check if the edge is on the boundary; if yes, then find the starting point. Use a Boolean variable--Boundary to record if the starting point is on the boundary. 3. If not, check if the other edge, which is intersected by the contour in the same element, is on the boundary. If yes, then this intersected point is the starting point. If no other edges are on the boundary, then the first edge is still the starting point. 4. When the starting point is found, record the element number in NumE and the position of the intersection in NumS, and set variable Boundary = 0 or 1, and go to the subroutine-- Draw to starting drawing the contour. 5. When finish a branch of the contour, continue to scan the other edges to search for new branches, until the scan is complete.

Table 5.2: Algorithm for drawing a contour

Subroutine: Draw
If Boundary = 1 1.1 Set boundary = 0 1.2 Search other intersections in the same element in a fixed order. 1.3 Draw a line segment between the two intersected points. 1.4 Set values of Side(i) of the two points as 0 to avoid meeting them again. 1.5 Move to the step 2.1.
If Boundary = 0 2.1 Search the adjacent element of the edge. 2.2 Search other intersections in the adjacent element in a fixed direction. 2.3 Draw a line segment between the two intersected points. 2.4 Set values of Side(i) of the second intersected point as 0 to avoid meeting it again. 2.5 Move to the adjacent element, and repeat the steps 2.1 to 2.5 until it stops at the boundary or it returns to the starting point.

5.2 Algorithm for volume and area parameter evaluation

5.2.1 Algorithm description

(1) Volume parameters

The calculation of concentration-, probability- and annoyance– weighted footprint areas, (*Fw(C), Fw (P),* and *Fw(A)*), is a solid geometric problem. Table 5.3 shows the algorithm and Table 5.4 shows all situations that may occur and the corresponding equations to calculate the area. Computer code for calculating *Fw(C), Fw(P)* and *Fw(A)* is given in Appendix 2.2.

Table 5.3: Algorithm for calculating *Fw(C), Fw(P)* and *Fw(A)*

Set PWFA=0
Do a loop for all elements: 1. Find the series numbers of four vertices of each element. 2. Get X, Y coordinates and Z values of four vertices. 3. Calculate intersected points of four edges, respectively. X1, Y2, X3, Y4 are the coordinates of intersected points for the North, East, South, and West edges, respectively. 4. Get case value for each element and calculate the PWFA of each element according to Table 5.4 by calling a subroutine Volume(). 5. Accumulate PWFA.

Subroutine Volume() mentioned in the algorithm above solves the following problem. Each element can be divided into several similar parts like irregular prisms as shown in Figure 5.2 (1).

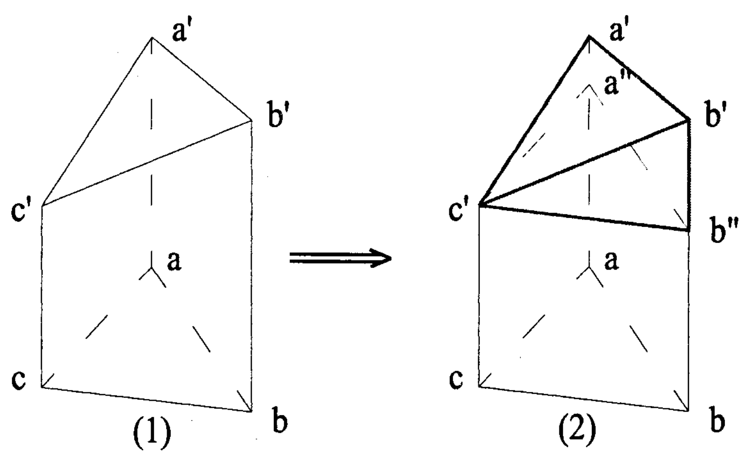
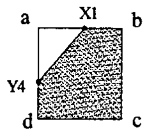
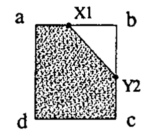
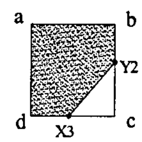
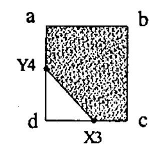
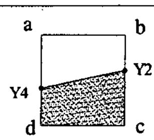
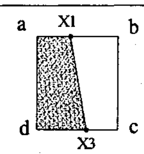
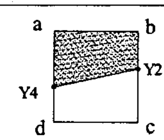
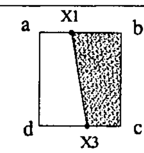
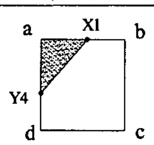
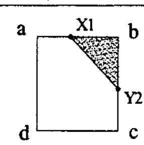
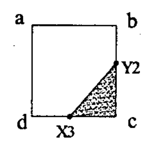
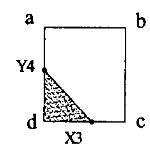
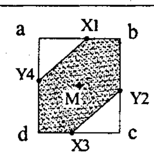
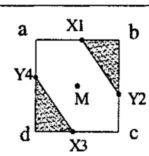
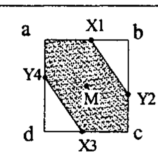
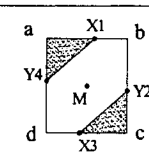
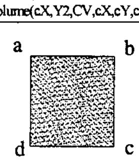
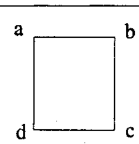


Figure 5.2: 3D graph illustrating spatial concept of PWFA

Table 5.4: Case value, conditions and probability weighted footprint area equations for all situations

				
Case Value	21	22	23	24
Conditions	aZ<CV,bZ>CV,cZ>CV,dZ>CV	aZ>CV,bZ<CV,cZ>CV,dZ>CV	aZ>CV,bZ>CV,cZ<CV,dZ>CV	aZ>CV,bZ>CV,cZ>CV,dZ<CV
Equations	Fw()=Volume(aX,aY,aZ,bX,bY,bZ,cX,cY,cZ) +Volume(cX,cY,cZ,dX,dY,dZ,aX,aY,aZ) -Volume(aX,Y4,CV,aX,aY,aZ,X1,aY,CV)	Fw()=Volume(aX,aY,aZ,bX,bY,bZ,cX,cY,cZ) +Volume(cX,cY,cZ,dX,dY,dZ,aX,aY,aZ) -Volume(X1,bY,CV,bX,bY,bZ,bX,Y2,CV)	Fw()=Volume(aX,aY,aZ,bX,bY,bZ,cX,cY,cZ) +Volume(cX,cY,cZ,dX,dY,dZ,aX,aY,aZ) -Volume(cX,Y2,CV,cX,cY,cZ,X2,cY,CV)	Fw()=Volume(aX,aY,aZ,bX,bY,bZ,cX,cY,cZ) +Volume(cX,cY,cZ,dX,dY,dZ,aX,aY,aZ) -Volume(X3,dY,CV,dX,dY,dZ,dX,Y4,CV)
				
Case Value	31	32	33	34
Conditions	aZ<CV,bZ<CV,cZ>CV,dZ>CV	aZ>CV,bZ<CV,cZ<CV,dZ>CV	aZ>CV,bZ>CV,cZ<CV,dZ<CV	aZ<CV,bZ>CV,cZ>CV,dZ<CV
Equations	Fw()=Volume(cX,Y2,CV,cX,cY,cZ,dX,dY,dZ) +Volume(dX,dY,dZ,dX,Y4,CV,cX,Y2,CV)	Fw()=Volume(aX,aY,aZ,X1,aY,CV,X3,dY,CV) +Volume(X3,dY,CV,dX,dY,dZ,aX,aY,aZ)	Fw()=Volume(aX,Y4,CV,aX,aY,aZ,bX,bY,bZ) +Volume(bX,bY,bZ,bX,Y2,CV,aX,Y4,CV)	Fw()=Volume(X1,bY,CV,bX,bY,bZ,cX,cY,cZ) +Volume(cX,cY,cZ,X3,cY,CV,X1,bY,CV)
				
Case Value	41	42	43	44
Conditions	aZ>CV,bZ<CV,cZ<CV,dZ<CV	aZ<CV,bZ>CV,cZ<CV,dZ<CV	aZ<CV,bZ<CV,cZ>CV,dZ<CV	aZ<CV,bZ<CV,cZ<CV,dZ>CV
Equations	Fw()=Volume(aX,Y4,CV,aX,aY,aZ,X1,aY,CV)	Fw()=Volume(X1,bY,CV,bX,bY,bZ,bX,Y2,CV)	Fw()=Volume(cX,Y2,CV,cX,cY,cZ,X2,cY,CV)	Fw()=Volume(X3,dY,CV,dX,dY,dZ,dX,Y4,CV)
				
Case Value	511	512	521	522
Conditions	aZ<CV,bZ>CV,cZ<CV,dZ>CV M>CV	aZ<CV,bZ>CV,cZ<CV,dZ>CV M<CV	aZ>CV,bZ<CV,cZ>CV,dZ<CV M>CV	aZ>CV,bZ<CV,cZ>CV,dZ<CV M<CV
Equations	Fw()=Volume(aX,aY,aZ,bX,bY,bZ,cX,cY,cZ) +Volume(cX,cY,cZ,dX,dY,dZ,aX,aY,aZ) -Volume(aX,Y4,CV,aX,aY,aZ,X1,aY,CV) -Volume(cX,Y2,CV,cX,cY,cZ,X3,cY,CV)	Fw()=Volume(X1,bY,bZ,bX,bY,bZ,bX,Y2,cZ) +Volume(X3,dY,CV,dX,dY,dZ,dX,Y4,CV)	Fw()=Volume(aX,aY,aZ,bX,bY,bZ,cX,cY,cZ) +Volume(cX,cY,cZ,dX,dY,dZ,aX,aY,aZ) -Volume(X1,bY,CV,bX,bY,bZ,bX,Y2,CV) -Volume(X3,dY,CV,dX,dY,dZ,dX,Y4,CV)	Fw()=Volume(aX,Y4,CV,aX,aY,aZ,X1,aY,CV) +Volume(cX,Y2,CV,cX,cY,cZ,X3,cY,CV)
				
Case Value	1	6		
Conditions	aZ>CV,bZ>CV,cZ>CV,dZ>CV	aZ<CV,bZ<CV,cZ<CV,dZ<CV		
Equations	Fw()=Volume(aX,aY,aZ,bX,bY,bZ,cX,cY,cZ) +Volume(cX,cY,cZ,dX,dY,dZ,aX,aY,aZ)	Fw()=0		

aX, bX, cX, dX: Coordination in X direction for vertice a, b, c, d;
aY, bY, cY, dY: Coordination in Y direction for vertice a, b, c, d;
aZ, bZ, cZ, dZ: Z values for vertice a, b, c, d;
X1, Y2, X3, Y4: intersections in the four edges;
CV: Contour value;
M: Mean value of Z of vertice a, b, c, d.

From the lowest Z value point c' cut a parallel plan (c'a"b") to plan (abc), then the irregular prism is divided into a regular prism (abca"b"c,) and a tetrahedron with top point c' and bottom plan a'a''b'b'', as shown in Figure 5.2 (2). The theoretical solution of volume can be solved using the equations outlined below where aX and aY are the X and Y coordinates of point a and aZ is the height of the edge aa'. A similar nomenclature is used for points, b and c.

$$Edge: ab = \sqrt{\left((aX - bX)^2 + (aY - bY)^2\right)} \tag{5.1}$$

$$Edge: bc = \sqrt{\left((bX - cX)^2 + (bY - cY)^2\right)} \tag{5.2}$$

$$Edge: ac = \sqrt{\left((aX - cX)^2 + (aY - cY)^2\right)} \tag{5.3}$$

$$Angle: A\_a = \arccos\left(\frac{ab^2 + ac^2 - bc^2}{2 \times ab \times ac}\right) \tag{5.4}$$

$$Area: abc = 0.5 \times ab \times ac \times \sin(A\_a) \tag{5.5}$$

$$Volume: abc - c'a''b'' = Area: abc \times cZ \tag{5.6}$$

$$Area: a'a''b'b'' = 0.5 \times (aZ + bZ) \times ab - ab \times cZ \tag{5.7}$$

$$Height = ac \times \sin(A\_a) \tag{5.8}$$

$$Volume: c' - a'a''b'b'' = \frac{1}{3} \times Area: a'a''b'b'' \times Height \tag{5.9}$$

$$Volume: abca'b'c' = Volume: abcc'a''b'' + Volume: c' - a'a''b'b'' \tag{5.10}$$

The above equations only express the solution for the situation where cZ is the lowest one. Equations are different when aZ or bZ is the lowest. However the principle is the same in these cases. When calling the subroutine Volume(), only the X, Y, and Z values of the three vertices are needed.

The calculation of the total concentration-, probability- and annoyance-weighted footprint areas is simpler. Table 5.5 summarizes the relevant algorithm and Appendix 2.3 contains the computer code for calculating $F_{WC}(T)$, $F_{WP}(T)$ and $F_{WA}(T)$.

Table 5.5: Algorithm for calculating $F_{WC}(T)$, $F_{WP}(T)$ and $F_{WA}(T)$

Set TotalVolume = 0
Do a loop for all elements: 1. Find the series numbers of four vertices of each element. 2. Get X, Y coordinates and Z values of four vertices. 3. Calculate the volume of each element by calling a subroutine Volume(). 4. Accumulate TotalVolume.

(2) Area parameters

$F(P)$ is a special case of $F_W(P)$. That is, equation 2.10 is equivalent to equation 2.11 when $P(x, y)$ constantly equals 100. Therefore, the algorithm for calculating $F(P)$ is the same as the one for calculating $Fw(P)$. The difference is that when calculating $F(P)$, the values of aa', bb' and cc' shown in Figure 5.2 must be set equal to 100. In the computer code contained in Appendix 2.2., a parameter was set to distinguish when $F(P)$ and $F_W(P)$ are being calculated. Similarly, $F(C)$ is a special case of $F_W(C)$ when concentration is 1 OU; and $F(A)$ is a special case of $F_W(A)$ when annoyance is 1 au.

In the development of this algorithm, another method to calculate $F(P)$ was also developed but was not incorporated into the software. According to this method, the calculation of $F(P)$ can be solved as a simple plan geometric problem rather than being treated as a special case of $F_W(P)$. The alternative algorithm for calculating footprint area is listed in the Table A2-1. All possible situations that may happen and the relevant equations are contained in Table A2-2. The corresponding code for calculating the footprint area is given in Appendix 2.4.

(3) Population impact as a special case of volume

The algorithm for calculating the concentration-, probability- and annoyance-weighted populations (i.e., $N_W(C)$, $N_W(P)$ and $N_W(A)$), is similar to the one used for calculating the concentration-, probability- and annoyance-weighted footprint areas (i.e., $F_W(C)$, $F_W(P)$ and $F_W(A)$). The only difference is that there needs to be an additional step No. 5 in Table 5.6 to deal with the population density when calculating $N_W(C)$, $N_W(P)$ and $N_W(A)$, as described below. Table 5.6 gives the algorithm for calculating $N_W(C)$, $N_W(P)$ and $N_W(A)$. The computer code for calculating these parameters is shown in Appendix 2.5.

Table 5.6: Algorithm for calculating $N_W(C)$, $N_W(P)$, and $N_W(A)$

Set NPA = 0
Do a loop for all elements: 1. Find the series numbers of four vertices of each element. 2. Get X, Y coordinates and Z values of four vertices. 3. Calculate intersected points of four edges respectively. X1, Y2, X3, Y4 are the coordinates of intersected points for the North, East, South, West edges respectively. 4. Get case value for each element and calculate the volume of each element according to Table 5.5 by calling a subroutine Volume(). 5. Find out the population density of the element and calculate the number people affected in this element. 6. Accumulate NPA.

Users should evaluate the population density of the study area using local census information. Firstly, they should divide the study area into several blocks, in which it will be assumed that each block has a uniform population density. Secondly, they should find the coordinates of the upper-left and lower-right corner of each block. As a simplification, users should make sure that edges of each block are on the grid lines that were used for dispersion modelling. Finally, users should input all the data into the OdorImp. Figure 5.3 shows an example of how regions of various population densities are represented in OdorImp. The method for inputting this data will be described in the next chapter.

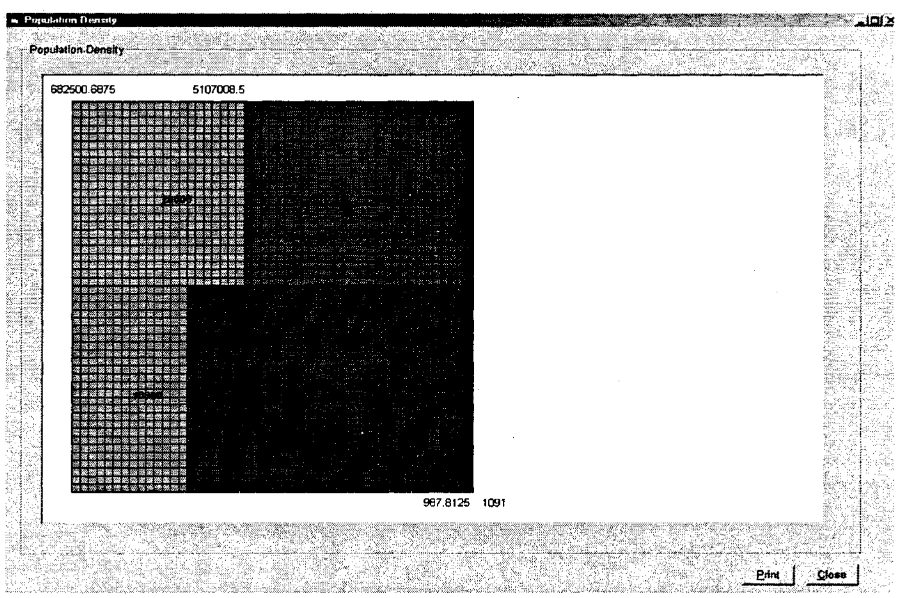


Figure 5.3: Example of population density

Table 5.7 summarizes the algorithm for calculating the total concentration-, probability- and annoyance-weighted populations; i.e., $N_{WC}(T)$, $N_{WP}(T)$, and $N_{WA}(T)$. The computer code is contained in Appendix 2.6.

Table 5.7: Algorithm for calculating $N_{WC}(T)$, $N_{WP}(T)$ and $N_{WA}(T)$

Set NPA = 0
Do a loop for all elements: 1. Find the series numbers of four vertices of each element. 2. Get X, Y coordinates and Z values of four vertices. 3. Calculate the volume of each element by calling a subroutine Volume(). 4. Find out the population density of the element and calculate the number people affected in this element. 5. Accumulate NPA.

The calculation of the population within a concentration contour, *N(C)*, is a special case of calculating concentration weighted population, $N_W(C)$, when *C(x, y)* is 1 OU as can be seen when comparing equations 4.12 and 4.16. Therefore, the algorithms are same. The only difference is that the concentration of each grid point should be set to 1 when calculating population in a concentration contour, *N(C)*. The other two parameters, *N(P)* and *N(A)*, are evaluated similarly. Computer code for calculating *N(C)*, *N(P)* and *N(A)* may be found in Appendix 2.5.

5.2.2 Testing

In order to verify that the algorithms and computer codes were developed and implemented correctly, these algorithms were tested for accuracy. This was accomplished by comparing the output of the computer codes with values that were accurately known for simple geometries and by comparing results with those evaluated using a commercial package, Surfer 7.0, which may be used for similar purposes. A half sphere, a taper and a Gaussian dispersion model were chosen to test the algorithms. Their 3D profiles were listed in Figure 3.3, 3.4 and 3.2 respectively.

(1) Area parameters

a. Half sphere

Table 5.8 shows the calculated footprint area for the half sphere of radius 1000 m according to OdorImp, theory, and Surfer under different contours and their relative differences. The contours are in specified in meters, which is equivalent to the idea of specifying the contours in terms of annoyance, concentration and probability. Grid points that were input into OdorImp and Surfer had a grid spacing of 50 m. The following equation was used to calculate the theoretical value:

$$F = \pi \times (R_a^{\ 2} - ContV^2) \tag{5.11}$$

where R_a is the radius of the sphere (in m) and *ContV* is the contour value of interest (in m). Figure 5.4 shows tendencies in differences between theoretical and approximated values as a function of the contour that was selected for analysis.

A brief sensitivity analysis was performed to evaluate the effect of grid spacing on the accuracy of OdorImp and Surfer, relative to the theoretical values. Results are shown in the Figures 5.5 to 5.7. As can be seen, the results of footprint area from OdorImp for the half sphere are quite accurate and when the grid spacing is denser, the accuracy is improved.

Table 5.8: Results of footprint area for the half sphere for a grid spacing of 50 m

Contour (m)	Footprint area (m^2)			Relative difference (%)		
	OdorImp	Surfer	Theory	OdorImp vs. Theory	Surfer vs. Theory	OdorImp vs. Surfer
100	3106653	3136602	3110176	-0.11	0.85	-0.95
200	2986074	2972020	3015928	-0.99	-1.46	0.47
300	2847220	2816462	2858849	-0.41	-1.48	1.09
400	2631207	2621689	2638937	-0.29	-0.65	0.36
500	2350492	2346272	2356194	-0.24	-0.42	0.18
600	2006441	2002470	2010619	-0.21	-0.41	0.20
700	1598726	1595497	1602212	-0.22	-0.42	0.20
800	1128022	1125212	1130973	-0.26	-0.51	0.25
900	593855	591929	596902	-0.51	-0.83	0.33

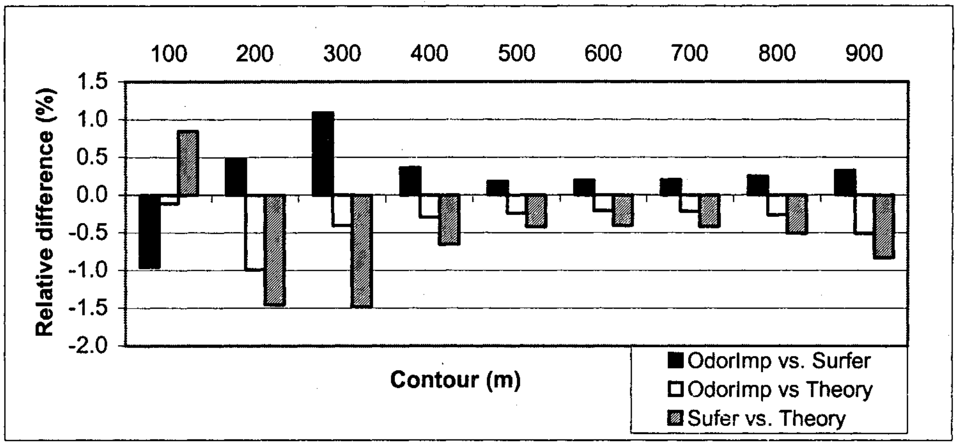


Figure 5.4: Results of footprint areas for the half sphere for a grid spacing of 50 m

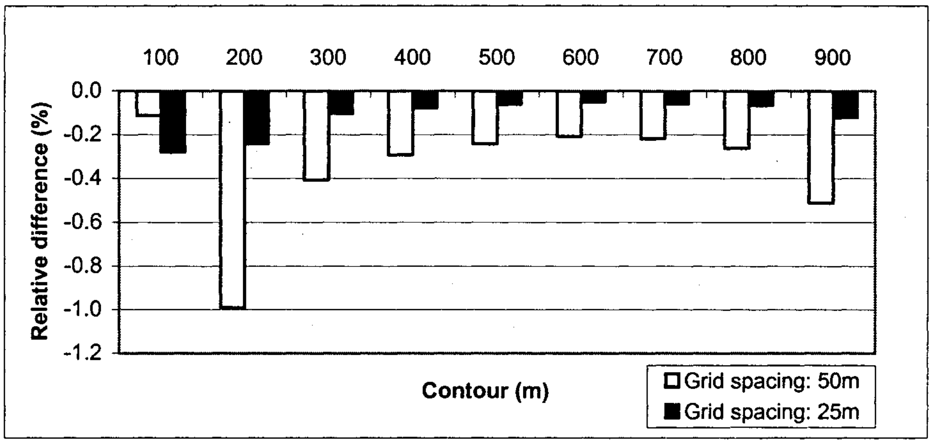


Figure 5.5: Relative differences in footprint area for the half sphere between OdorImp and theory for grid spacings of 50 m and 25 m

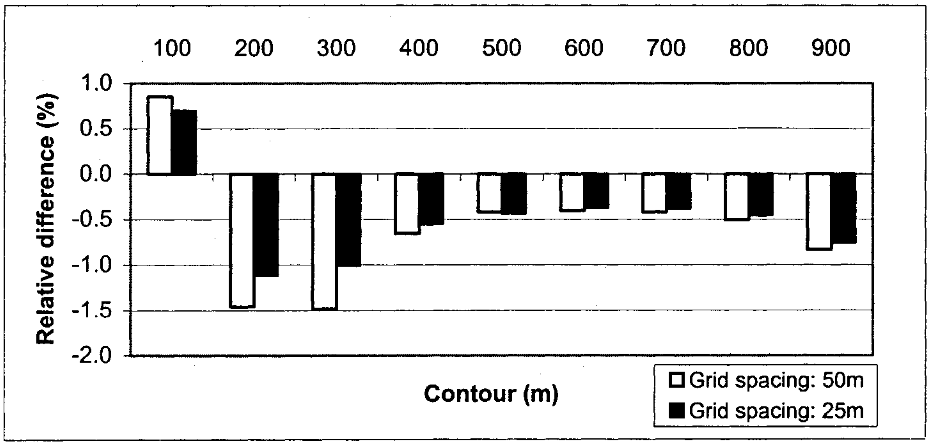


Figure 5.6: Relative differences in footprint area for the half sphere between Surfer and theory for grid spacings of 50 m and 25 m

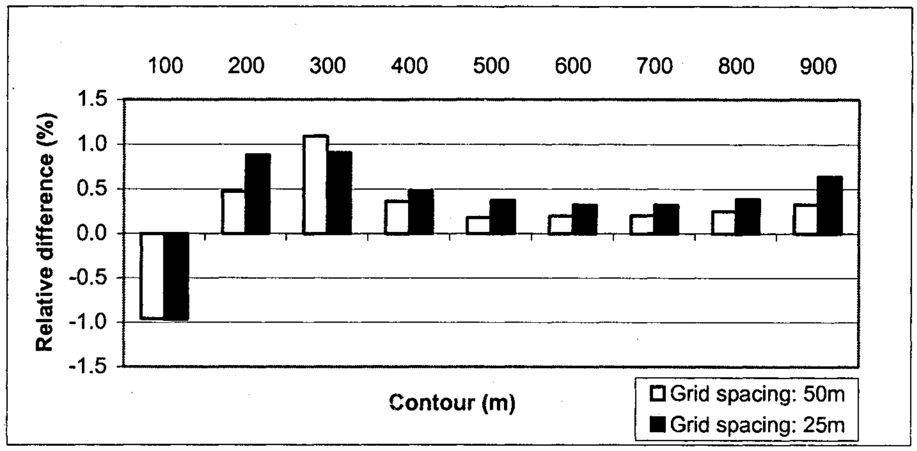


Figure 5.7: Relative differences in footprint area for the half sphere between OdorImp and Surfer for grid spacings of 50 m and 25 m

b. Taper

Table 5.9 and Figure 5.8 show the calculated footprint areas for a taper of radius 1000 m according to theory, OdorImp, and Surfer under different contours and their relative differences. Similar to the half sphere, the contours are in expressed in meters which are analogous to contours of annoyance, concentration and probability. Grid points that were input into OdorImp and Surfer had a grid spacing of 50 m. The following equation was used to calculate the theoretical footprint area (in m^2) for the taper:

$$F = \pi \times (R_a^2 - ContV^2) \quad (5.12)$$

where R_a is the radius (in m) of the bottom plan of the taper and $ContV$ is the contour value (in m).

Table 5.9: Results of footprint area for the taper for a grid spacing of 50 m

Contour (m)	Footprint area (m^2)			Relative difference (%)		
	OdorImp	Surfer	Theory	OdorImp vs. Theory	Surfer vs. Theory	OdorImp vs. Surfer
100	2543072	2541534	2548200	-0.20	-0.26	0.06
200	2009059	2008301	2013392	-0.22	-0.25	0.04
300	1537789	1537167	1541503	-0.24	-0.28	0.04
400	1129315	1128758	1132533	-0.28	-0.33	0.05
500	783684	783164	786481	-0.36	-0.42	0.07
600	501001	500461	503348	-0.47	-0.57	0.11
700	281157	280588	283133	-0.70	-0.90	0.20
800	124084	123369	125837	-1.39	-1.96	0.58
900	29670	29323	31459	-5.69	-6.79	1.18

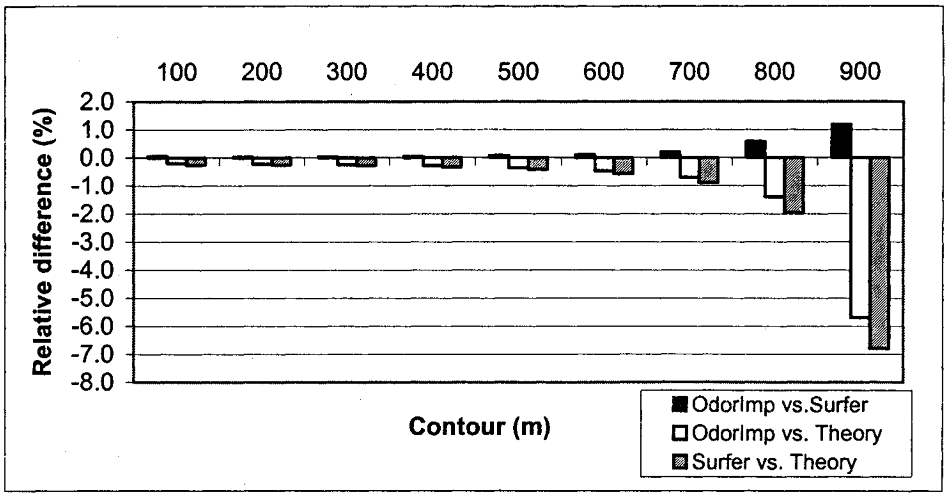


Figure 5.8: Results for footprint area of the taper for a grid spacing of 50 m

As above, a sensitivity analysis was done by changing grid space from 50 m to 25 m. Figures 5.9 to 5.11 show the results for the taper analyses.

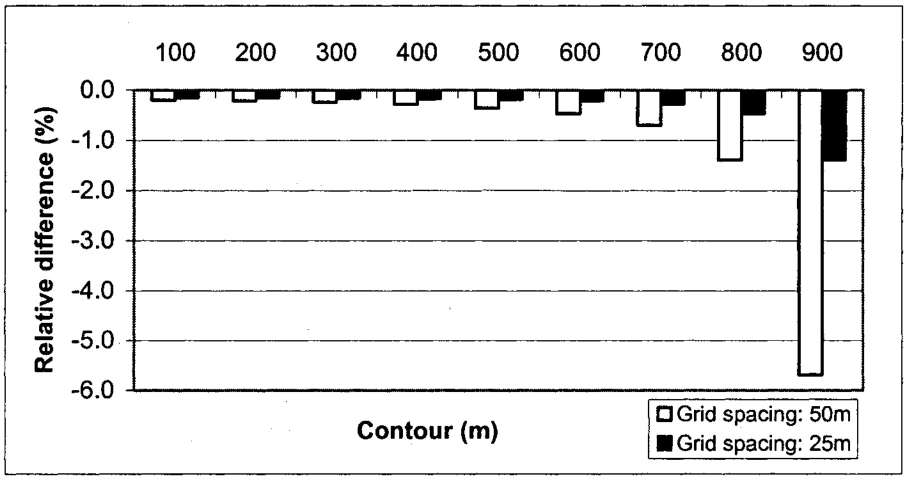


Figure 5.9: Relative differences in footprint area for the taper between OdorImp and theory for grid spacings of 50 m and 25 m

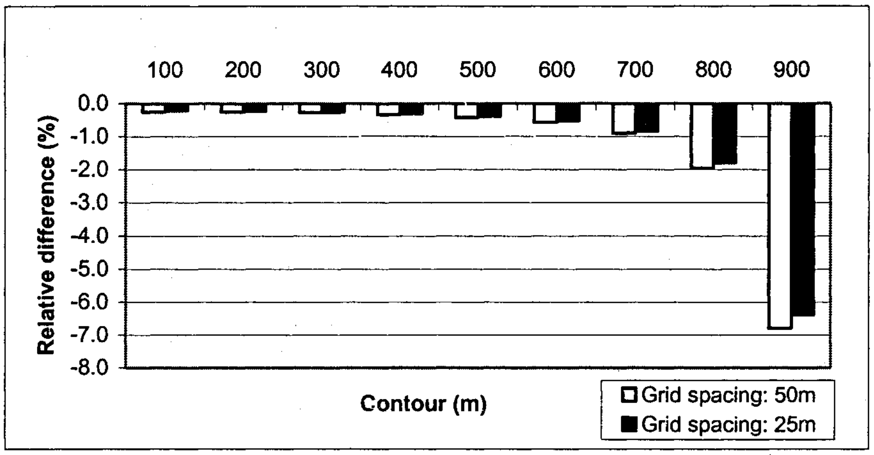


Figure 5.10: Relative differences in footprint area for the taper between Surfer and theory for grid spacings of 50 m and 25 m

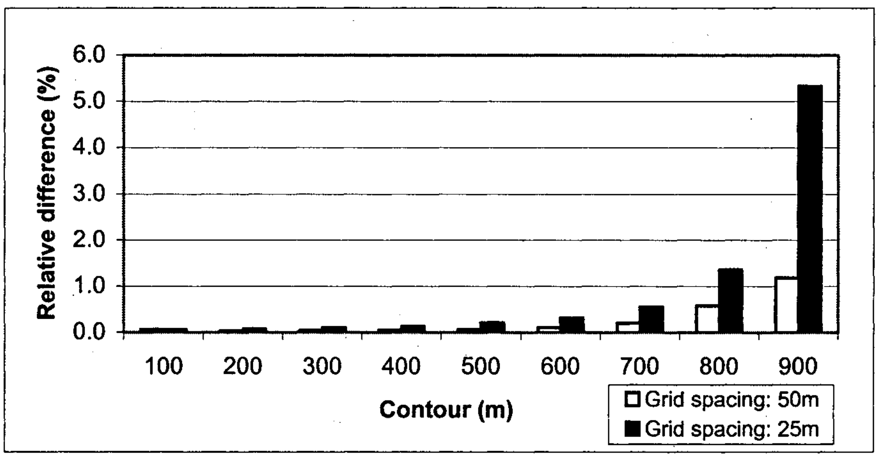


Figure 5.11: Relative differences of footprint area of the taper between OdorImp and Surfer for grid spacings of 50 m and 25 m

c. Gaussian dispersion model

The ability of the algorithm was also tested on artificial data that was produced by a Gaussian dispersion model. This test was performed to evaluate the capabilities of the algorithms when applied to a contour surface that was not as uniform as a sphere or taper. Comparisons could only be done between the results of OdorImp and Surfer because it was not possible to calculate theoretical values for the equations used in the Gaussian dispersion model. Here, the concentration is in OU. The results from OdorImp and Surfer for two grid spaces are shown in Table 5.10. Notably, the relative difference between the two programs was always less than 1%.

Table 5.10: Results of footprint area for the Gaussian dispersion model under grid spacings of 50 m and 25 m

Contour (OU)	$F(C)$ (OU·m^2) with grid spacing = 50 m			$F(C)$ (OU·m^2) with grid spacing = 25 m		
	OdorImp	Surfer	Relative difference (%)	OdorImp	Surfer	Relative difference (%)
0.2	5165177	5167063	-0.04	5149361	5169471	-0.39
0.4	3581534	3585063	-0.10	3567519	3582467	-0.42
0.6	2875094	2881514	-0.22	2863171	2878313	-0.53
0.8	2454636	2460933	-0.26	2443418	2457570	-0.58
1.0	2168301	2175679	-0.34	2157438	2172158	-0.68
1.2	1956996	1964473	-0.38	1946945	1960750	-0.70
1.4	1793399	1801967	-0.48	1783675	1798060	-0.80
1.6	1661450	1669992	-0.51	1652338	1665980	-0.82
1.8	1553831	1561274	-0.48	1544114	1557153	-0.84
2.0	1461806	1470963	-0.62	1452740	1466744	-0.95

(2) Volume parameters

Similar testing was also conducted to test the accuracy of the volume algorithm in OdorImp.

a. Half sphere

Table 5.11 shows the calculated volume, V, for a half sphere of radius 1000 m by OdorImp under different contours and a comparison of these results to those evaluated using Surfer 7.0 and theory. The following equation was used to calculate the theoretical volume, V, (m^3) for a half sphere:

$$V = \frac{2\pi}{3}\left(R_a^{\,3} - ContV^3\right) \qquad (5.13)$$

where R_a is the radius of the half sphere (in m) and $ContV$ is the contour value (in m). Here, the concept of the volume of the half sphere is analogous to the concepts of concentration-weighted footprint area, $F_W(C)$, probability-weighted footprint area, $F_W(P)$, and annoyance-weighted footprint area, $F_W(A)$. However, the unit of the volume is m^3 instead of OU·m^2, m^2 or au· m^2 for $F_W(C)$, $F_W(P)$ or $F_W(A)$, respectively. Figure 5.12 shows the tendency in the relative errors as a function of the selected contour.

Table 5.11: Results of volume for half sphere for a grid spacing of 50 m

Contour (m)	Volume (m^3)			Relative difference (%)		
	OdorImp	Theory	Surfer	OdorImp vs. theory	Surfer vs. theory	OdorImp vs. Surfer
100	2082926342	2092300672	2081131390	-0.45	-0.53	0.09
200	2065072793	2077639906	2056731660	-0.60	-1.01	0.41
300	2030417048	2037846400	2017740580	-0.36	-0.99	0.63
400	1954551017	1960353782	1948992611	-0.30	-0.58	0.29
500	1827793236	1832595683	1824381997	-0.26	-0.45	0.19
600	1638104844	1642005732	1634768129	-0.24	-0.44	0.20
700	1372628394	1376017559	1369707398	-0.25	-0.46	0.21
800	1019115823	1022064793	1016460673	-0.29	-0.55	0.26
900	564562884	567581063	562638818	-0.53	-0.87	0.34

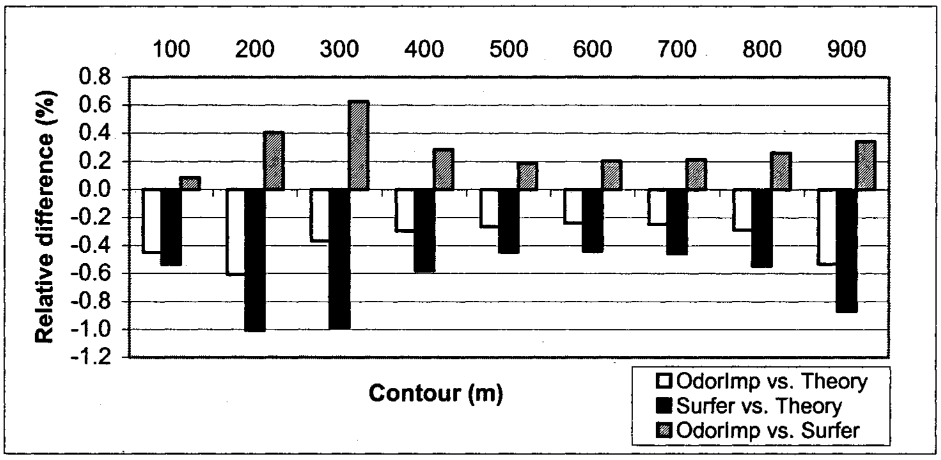


Figure 5.12: Results of volume for the half sphere for a grid spacing of 50 m

A sensitivity analysis was done by changing grid space from 50 m to 25 m. Figure 5.13 to Figure 5.15 show the results. Similar to the results of footprint area, the results of volume from OdorImp for the half sphere are quite accurate and when the grid spacing is denser, the accuracy is improved.

Table 5.12 summarizes the results for the total volume of the half sphere (when ContV is 0) of the half sphere. The concept of the total volume of the half sphere (m^3) is analogous to the total concentration-weighted footprint area, $F_{WC}(T)$, total probability-weighted footprint area, $F_{WP}(T)$, and total annoyance-weighted footprint area, $F_{WA}(T)$ in OU·m^2, m^2 and au·m^2, respectively.

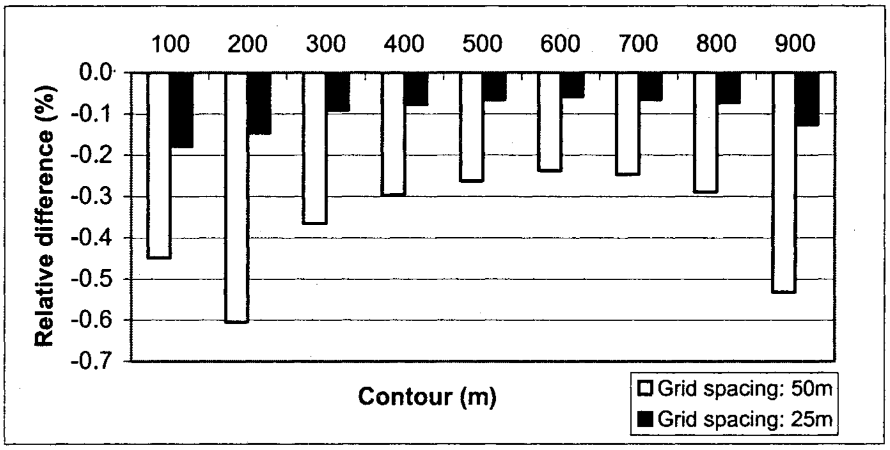


Figure 5.13: Relative differences of volume of the half sphere between OdorImp and theory for grid spacings of 50 m and 25 m

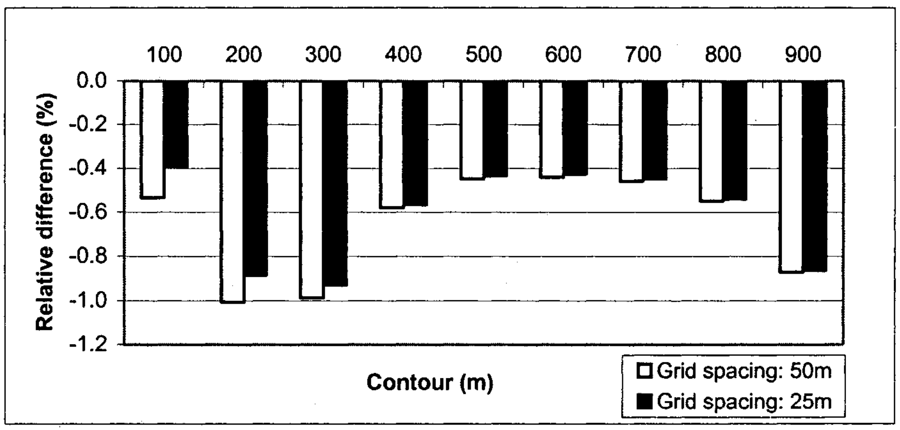


Figure 5.14: Relative differences of volume of half sphere between Surfer and theory for grid spacings of 50 m and 25 m

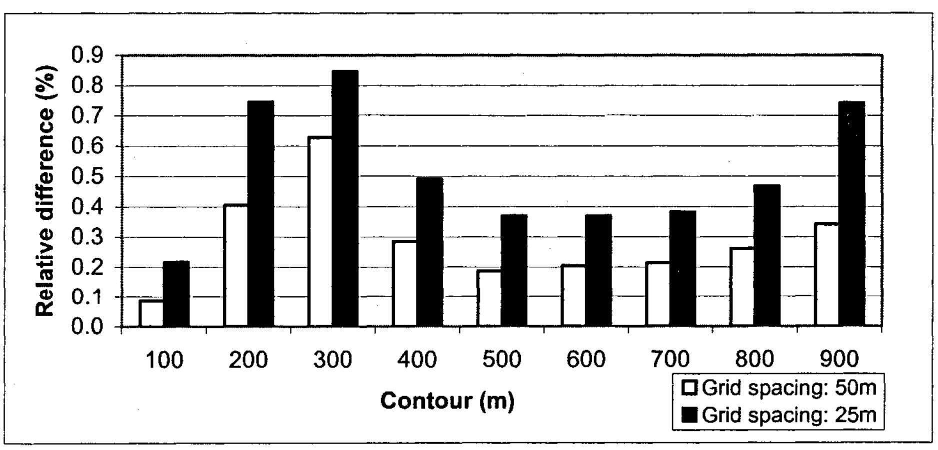


Figure 5.15: Relative differences of volume of half sphere between OdorImp and Surfer for grid spacings of 50 m and 25 m

Table 5.12: Results of total volume evaluations of the half sphere

Grid spacing	Total volume (m^3)			Relative difference (%)		
	OdorImp	Surfer	Theory	OdorImp vs. Theory	OdorImp vs. Surfer	Surfer vs. Theory
50 m	2091776862	2091822299	2094395067	-0.13	0.00	-0.12
25 m	2093945219	2093570924		-0.02	0.02	-0.04

b. Taper

Table 5.13 shows the results for the evaluation of volume of the taper of radius 1000 m for a grid spacing of 50 m. Equation 5.14 was used to calculate the theoretical volume, V (in m^3), under different contours of the taper:

$$V = \frac{\pi}{3}(R_a - ContV)^3 \tag{5.14}$$

where R_a is the radius of bottom plan (in m) of the taper and *ContV* is the contour value (in m). Similar to the case of the half sphere, here the concept of volume is analogous to the concentration-weighted footprint area, $F_W(C)$, probability-weighted footprint area, $F_W(P)$, and annoyance-weighted footprint area, $F_W(A)$. Figures 5.16 to 5.19 show the tendencies in the relative differences as a function of contour value and grid spacing. The results of the taper and the half sphere had the same tendencies. Table 5.14 gives the total volume of the taper when *ContV* equals 0 in equation 5.14.

Table 5.13: Results of volume for the taper for a grid spacing of 50 m

Contour (m)	Volume (m^3)			Relative difference (%)		
	OdorImp	Theory	Surfer	OdorImp vs. Theory	Surfer vs. Theory	OdorImp vs. Surfer
100	1016563950	1017876002	1015564930	-0.13	-0.23	0.10
200	936957852	938288990	936073051	-0.14	-0.24	0.09
300	819637766	821002866	818811743	-0.17	-0.27	0.10
400	677161961	678584002	676391947	-0.21	-0.32	0.11
500	522116993	523598767	521399958	-0.28	-0.42	0.14
600	367124855	368613532	366435767	-0.40	-0.59	0.19
700	224719988	226194667	224044110	-0.65	-0.95	0.30
800	107410766	108908543	106652833	-1.38	-2.07	0.71
900	27649972	29321531	27237166	-5.70	-7.11	1.52

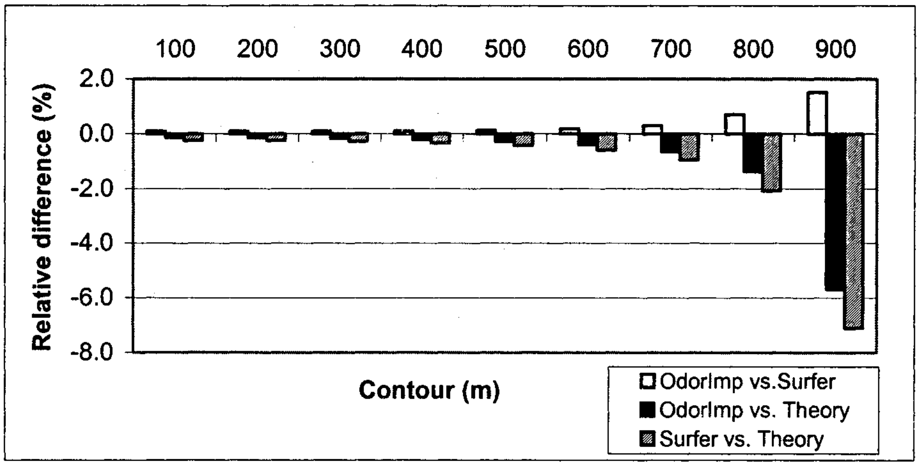


Figure 5.16: Results of volume for the taper for a grid spacing of 50 m

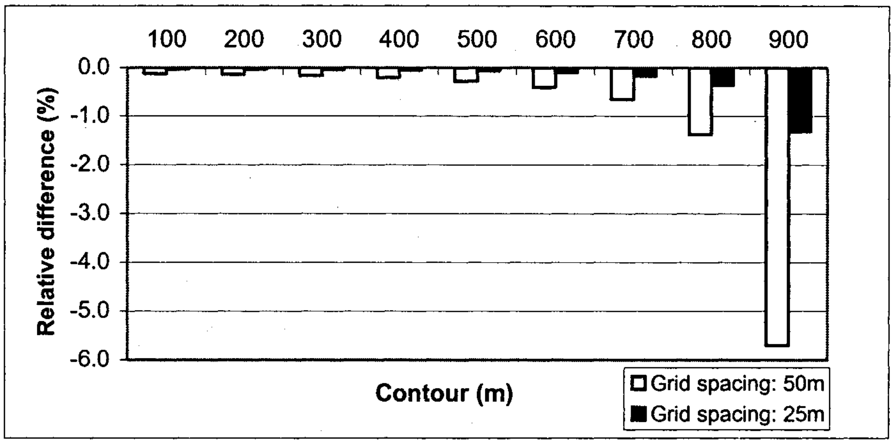


Figure 5.17: Relative differences of volume of the taper between OdorImp and theory for grid spacings of 50 m and 25 m

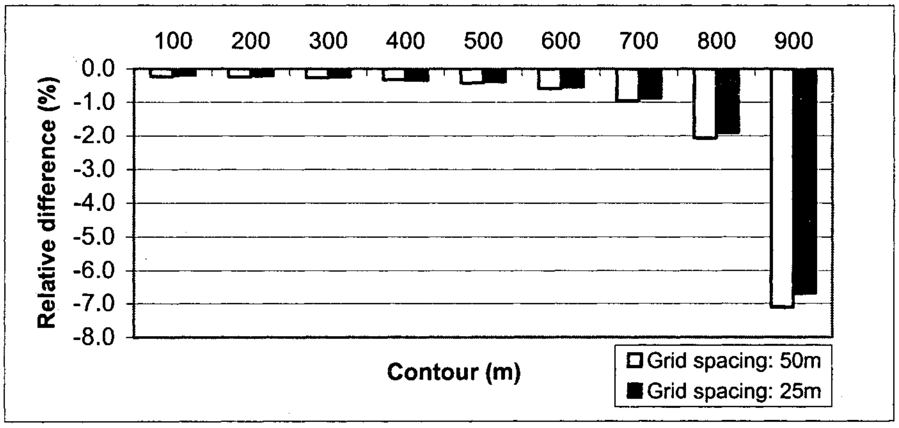


Figure 5.18: Relative differences of volume of the taper between Surfer and theory for grid spacings of 50 m and 25 m

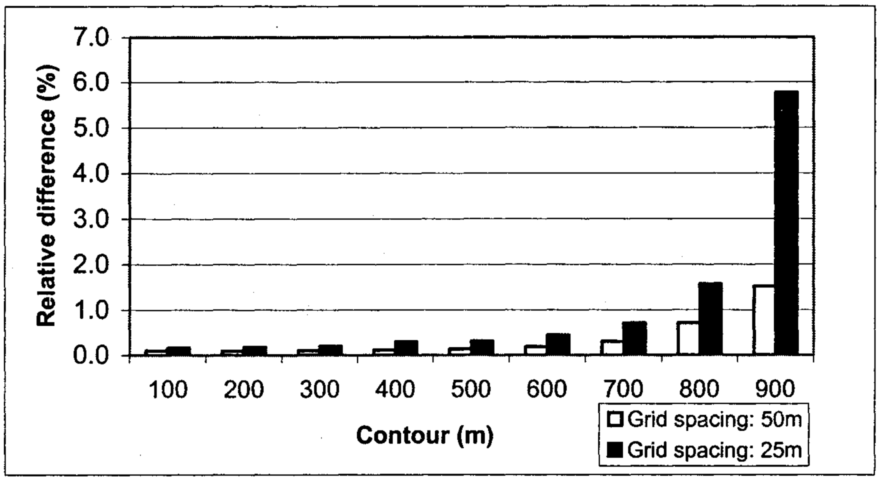


Figure 5.19: Relative differences of volume of the taper between OdorImp and Surfer for grid spacings of 50 m and 25 m

Table 5.14: Results of the total volume of the taper

Grid spacing	Total volume (m^3)			Relative difference (%)		
	OdorImp	Surfer	Theory	OdorImp vs. Theory	OdorImp vs. Surfer	Surfer vs. Theory
50 m	1046104727	1047188540	1047197533	-0.10	-0.10	0.00
25 m	1047189227	1047171068		0.00	0.00	0.00

c. Gaussian dispersion model

Similar tests were conducted using data generated from the Gaussian dispersion model. As before, in this case no theoretical values could be calculated as a basis for comparison. The concentrations generated are in OU. Results of concentration-weighted footprint area, $F_W(C)$, and the total concentration-weighted footprint area, $F_{WC}(T)$, are shown in Tables 5.15 and 5.16.

5.2.3 Discussion

When calculating the values of footprint areas and volumes under contours, the relative errors between OdorImp and theoretical values are very small, with OdorImp tending to slightly underestimate theoretical values. This is reasonable because the algorithm in OdorImp approximates a smooth curved convex surface with a group of planar surfaces. For example, in Figure 5.2, the surface a'b'c' represents a convex surface

Table 5.15: Results of *Fw(C)* for the Gaussian dispersion model for grid spacings of 50 m and 25 m

Contour (OU)	$F_W(C)$ ($OU \cdot m^2$) with grid spacing = 50 m			$F_W(C)$ ($OU \cdot m^2$) with grid spacing = 25 m		
	OdorImp	Surfer	Relative difference (%)	OdorImp	Surfer	Relative difference (%)
0.2	79871725	80331998	-0.57	78836909	79014548	-0.22
0.4	79377056	79884224	-0.63	78382388	78565359	-0.23
0.6	78979218	79539531	-0.70	78030370	78220380	-0.24
0.8	78641457	79268128	-0.79	77732555	77928866	-0.25
1.0	78339442	78992896	-0.83	77469893	77673495	-0.26
1.2	78035650	78761610	-0.92	77231950	77441987	-0.27
1.4	77774861	78550929	-0.99	77013641	77231070	-0.28
1.6	77530655	78353405	-1.05	76810041	76933387	-0.16
1.8	77302326	78168954	-1.11	76617833	76848752	-0.30
2.0	77081983	77997588	-1.17	76437904	76677201	-0.31

Table 5.16: Results of the $F_{WC}(T)$ of the Gaussian dispersion model

Grid spacing	$F_{WC}(T)$ ($OU \cdot m^2$)		Relative difference (%)
	OdorImp	Surfer	
50 m	80632955	80700563	-0.08
25 m	78601996	78705506	-0.13

that has been approximated as a planar triangular surface. In some situations it is possible that portions of the surface may be convex or concave. Therefore, the relative errors on occasion may be negative or positive. It should be noted that Surfer has the same tendencies as OdorImp relative to theoretical values.

As can be seen from Table 5.8 and Figure 5.4, the results of the footprint areas of the half sphere from OdorImp are very close to the theoretical values and are more accurate than Surfer. In addition, Figure 5.5 shows that footprint areas calculated by OdorImp are quite sensitive to grid spacing and that relative errors between OdorImp and theory were reduced significantly with denser grids. Figure 5.6 shows that footprint areas calculated by Surfer are relatively stable and do not change significantly with denser grid spacing. Figure 5.7 shows that the relative differences between OdorImp and Surfer results becomes greater when the grid spacing is changed from 50 m to 25 m. This growing deviation between the two programs appears to arise from the tendency of

OdorImp to be more accurate than Surfer when smaller grid spacing is used. This can be seen in Figures 5.5 and 5.6, where as the grid spacing is reduced, the error associated with OdorImp is reduced more than the error associated with Surfer.

As can be seen from the values of footprint areas for the taper in Table 5.9, the results from OdorImp are fairly close to theoretical values and are more accurate than those calculated using Surfer. Note that the high error between the OdorImp value and the theoretical value for the 900 m contour results from the fact that this contour is only several grid spaces in width. Under this condition, very few grid points are used to evaluate the area and, thus, there is a tendency toward greater error. As shown in Figure 5.9, a halving of the grid spacing to 25 m significantly improves the accuracy of the method. This demonstrates the need to use a reasonable grid density in order to ensure the accuracy of parameter evaluation. The results for the calculation of the volume of the taper have the same characteristics as for the footprint area and, similarly, OdorImp achieved better results than Surfer in these cases. The dependence on grid density is also revealed through Figures 5.9 and 5.17, where the relative error becomes greater when the footprint area becomes smaller for a fixed grid spacing. For example when the contour is 100 m, the footprint area is the largest (thereby encompassing a large number of grid points) and its relative error is the smallest. When the contour is 900 m, the footprint area is the smallest (thereby encompassing few grid points), and its relative error is largest.

However, that the results for the half sphere do not have exactly the same tendency (see Figures 5.5 and 5.13). In Figure 5.5, the relative error is largest when the contour is 200 m. The relative error continues reducing as larger contours are chosen up until the contour is 600 m. Then the error decreases again. This is quite different from the taper. The high error associated with low contour values such as 200 m is likely to arise from the steep angle on the edge of the half sphere, where neighbouring grid points in this region are significantly different in value. Such large derivatives would tend to result in greater errors in the integration techniques that were used both in OdorImp and in Surfer (see Figures 5.6 and 5.14). This demonstrates that the accuracy of OdorImp and Surfer would also be influenced by the nature of the shape of the contour surface. Thus, it is recommended that when regions of the contours have steep slopes, a finer grid spacing should be used.

Surfer is a commercial software package that is used widely. Because the details of the algorithms used in Surfer are unknown, it is difficult to explain how the differences between OdorImp and Surfer arise. However, as can be seen by comparing the results of OdorImp with Surfer, the relative differences are fairly small whenever it is a regular shape, such as a half sphere or a taper, or it is an irregular shape, such as the Gaussian dispersion model. In addition, in many cases, OdorImp provided results that were more accurate than Surfer. Therefore, it is concluded that the algorithms used in OdorImp are reliable and sufficiently accurate for the evaluation of odour impact parameters.

6 Software Description and its Application to Field Data

The algorithms described in Chapter 5 were implemented in Visual Basic in a program called "OdorImp" to provide a user-friendly means for calculating the odour impact parameters described in Chapter 4. This software includes an interface for the input of dispersion model and odour impact model data and then output of odour impact analysis results in both graphical and tabular formats. The various capabilities of the software and instructions for its use will be described below. Its application will then be demonstrated by applying it to two case studies involving odour impacts. The first case study involves the assessment of odours from a pig farm in a rural setting, as described in detail by Gorgy (2003). The second case study involves the assessment of the odour impact of an industrial facility in South-western Ontario located inside an urban environment. This study was described in detail by Sikdar (2001).

6.1 Description of OdorImp

6.1.1 Data preparation

Before using OdorImp, several sets of data should be prepared in advance by the software user. The first one is the file of grid concentrations expressed in odour units (OU), which must be in text (*.txt) format. In the case of a dispersion model such as ISC-Aermod, the output file is in a format *.plt file. When using this particular dispersion model, users can convert *.plt into *.txt by following steps.

1. Open Excel;
2. Specify the *.plt file and open it;
3. The dialog box " Text Import Wizard" appears;
4. Choose the "Delimited" option button, then press "Next" button;
5. Choose the "Space" and "Tab" check boxes, then press "Next" button;
6. Press "Finish" button to finish the conversion;

7. Delete the title rows. Only keep three columns of *X* coordinates, *Y* coordinates and concentrations. Delete the other left columns. Make sure to delete all separate point data that are not on the grids;
8. Press menu "Data" and choose "Sort", first sort the data by *X* coordinates in ascending order, then by *Y* coordinates in ascending order. Then press OK.
9. Save the *.plt as *.txt. Reminder information appears. Choose "Yes" to keep the *.txt in Text.

The second data file that can be input into the model is the data of population density in the study region. The use of this data is optional and is only required if odour impact parameters based on population are to be evaluated. As mentioned previously, population densities must be specified in rectangular-shaped regions. The data that are required include *X*, *Y* coordinates of up-left and down-right corners of population blocks and the population density values. No advance preparation of an input file is required since the data can be input directly in OdorImp.

It is also assumed that the user has already analyzed odour impact model data and has evaluated parameters such as persistence, p, persistence of annoyance, a, and R (see equations 2.5 and 2.9). Note that these parameters may be evaluated using the existing software package called OdorCalc, into which it is envisioned OdorImp will ultimately be integrated.

6.1.2 OdorImp applications

OdorImp has a main menu including items of *New*, *Open project*, *Save project*, *Gaussian,* and *Exit*; and a tab controller including items of *Odour Specifications/Dispersion Model Data*, *Population Density*, *Contour Specifications*, and *Impact Parameters*.

In the main menu, item *New* is used to start a new project, item *Open* project is used to open a existing project, item *Save project* is used to save the current project, item *Gaussian* is used to create a simple output of a Gaussian dispersion model, and item *Exit* is used to exit OdorImp. Figure 6.1 shows the initial interface of OdorImp. In this window, users specify the *.txt input file, which is prepared according to steps 1 to 9

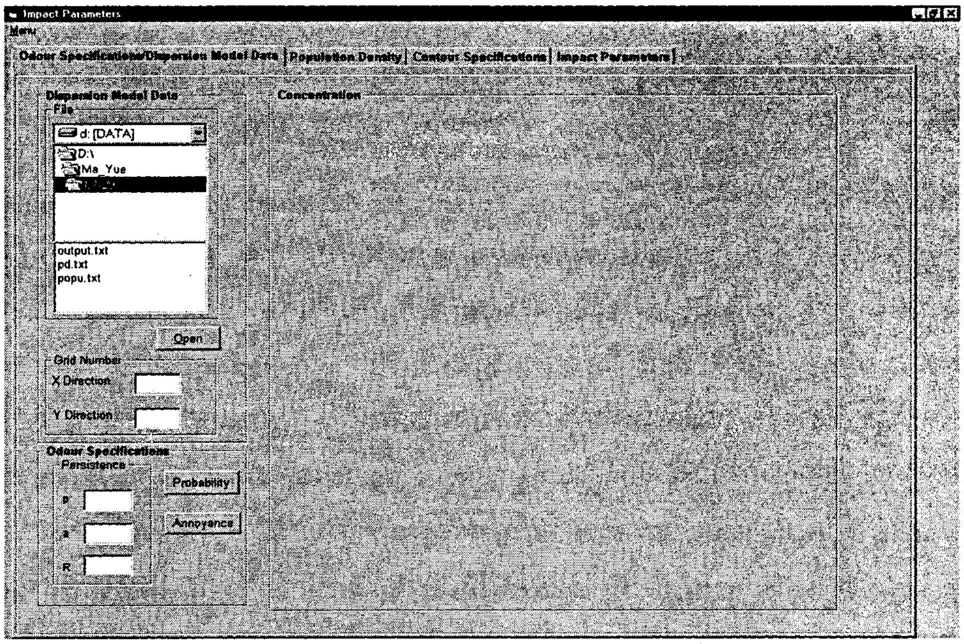


Figure 6.1: Opening window of OdorImp showing the *Odour Specifications/Dispersion Model Data* interface

above, and open it. Figure 6.2 shows an example of a project. The number of grid lines in the *X* and *Y* directions are calculated automatically. The input file shouldcontain an exact number of records. For example, if there are 6400 grid receptors in a study (e.g., 80 grids for *X* and *Y* directions, respectively), then the input file should contain exactly 6400 records. Each record must contain the *X*, *Y* coordinators of the grid and the concentration at this location in OU. If not, OdorImp will remind users to modify the input file to make sure it is correct. The grid on the right of the interface shows the concentration distributions relative to the original point (the up-left corner of the study area). If users plan to obtain the impact parameters of probability of response and annoyance, they should input the odour persistence, p, the annoyance persistence, a, and the ratio of $C_{50\%}$ over C_{5au}, R, then press the buttons of *Probability* and *Annoyance* to calculate probability of response and annoyance, respectively.

Upon selecting the *Population Density* tab, the interface shown in Figure 6.3 appears. In this interface, users can create and edit a population density file. There is a grid table in this window to show the data that are inputted (Figure 6.3). Users can move the cursor by the four arrows on the keyboard " ←", "↑", "→", "↓" to edit every unit in

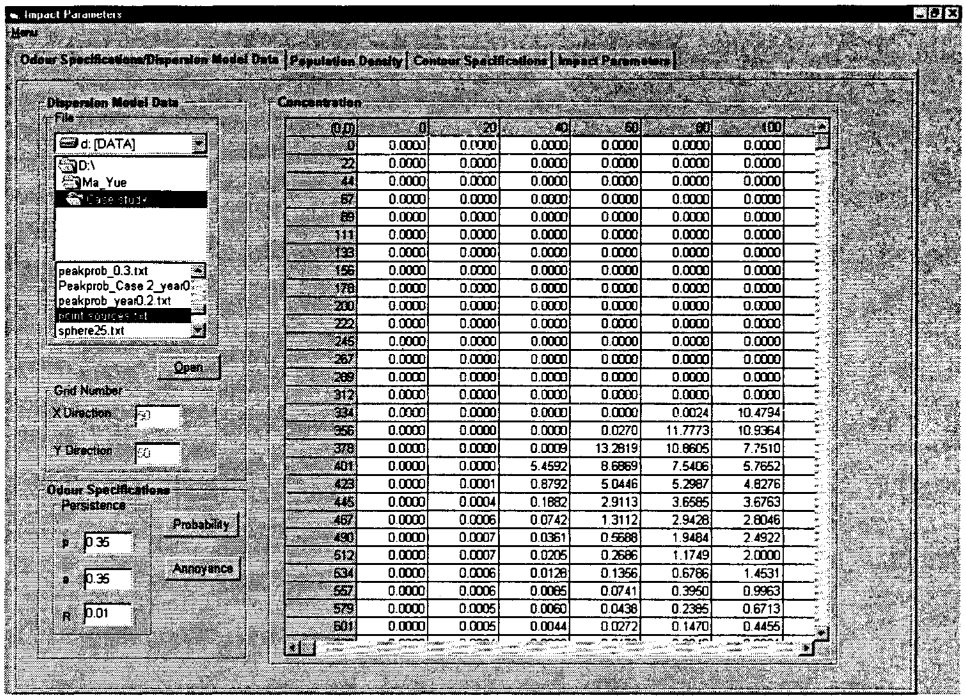


Figure 6.2: An example of a project

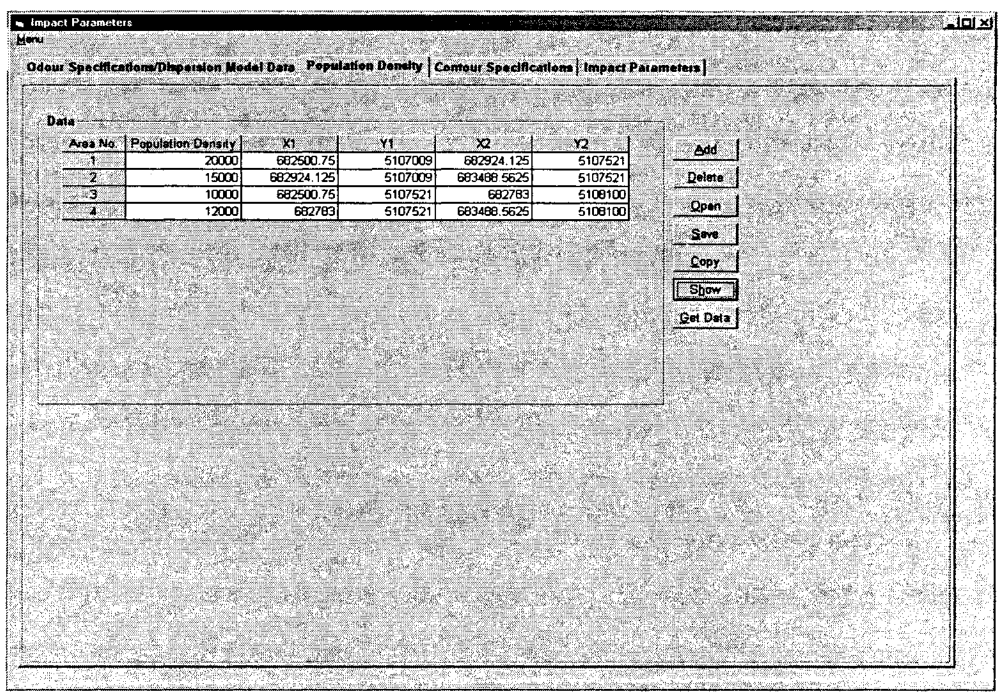


Figure 6.3: *Population density* window

the grid table. Moreover, users can use a group of command buttons to finish the following functions:

1. *Add*: add a new empty row;

2. *Delete*: delete the current row;
3. *Open*: open an existing file;
4. *Save*: save current data into a text file;
5. *Copy*: copy the data in the current row to a new empty row;
6. *Show*: show the graph of population density distribution, see Figure 5.3;
7. *Get Data*: read all numbers in the grid table into a particular array in the computer program. This step must be done after every modification and before the calculation to update the population density in the computer program.

Upon selecting the *Contour Specifications* tab, the interface in Figure 6.4 is shown. Users input and modify the contour starting values, increments and numbers of contours for concentration, probability of response, and annoyance respectively in this window. If users haven't already pressed the command button to calculate the probability of response and annoyance in advance in the window shown in Figure 6.1, they are unable to edit the contour specifications of probability and annoyance. Figure 6.5 shows an example of contours of concentration generated by OdorImp.

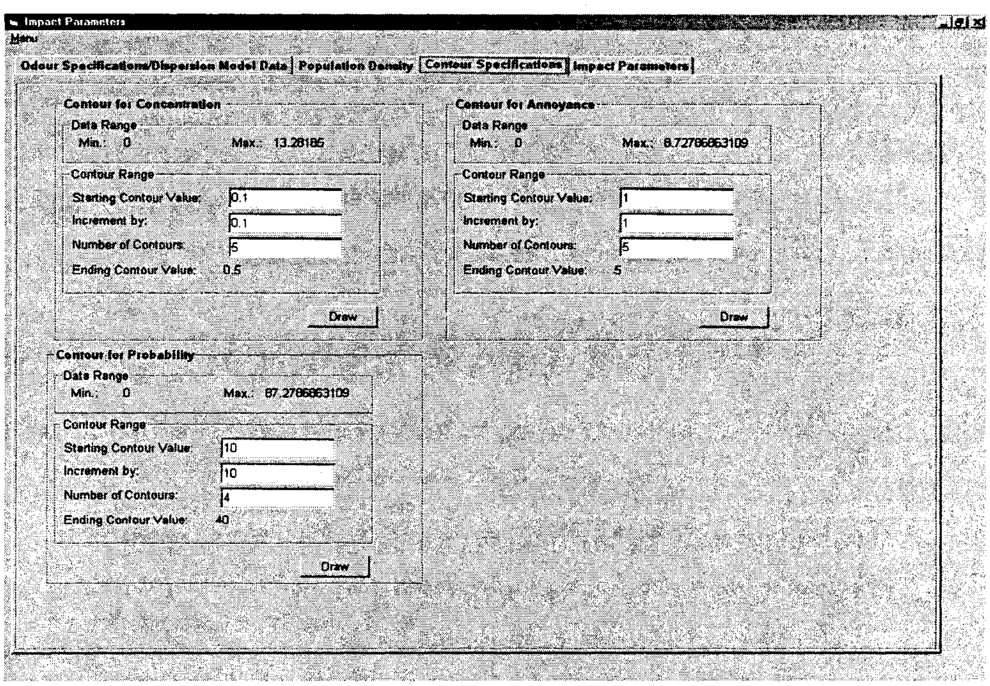


Figure 6.4: ***Contour Specification*** **window**

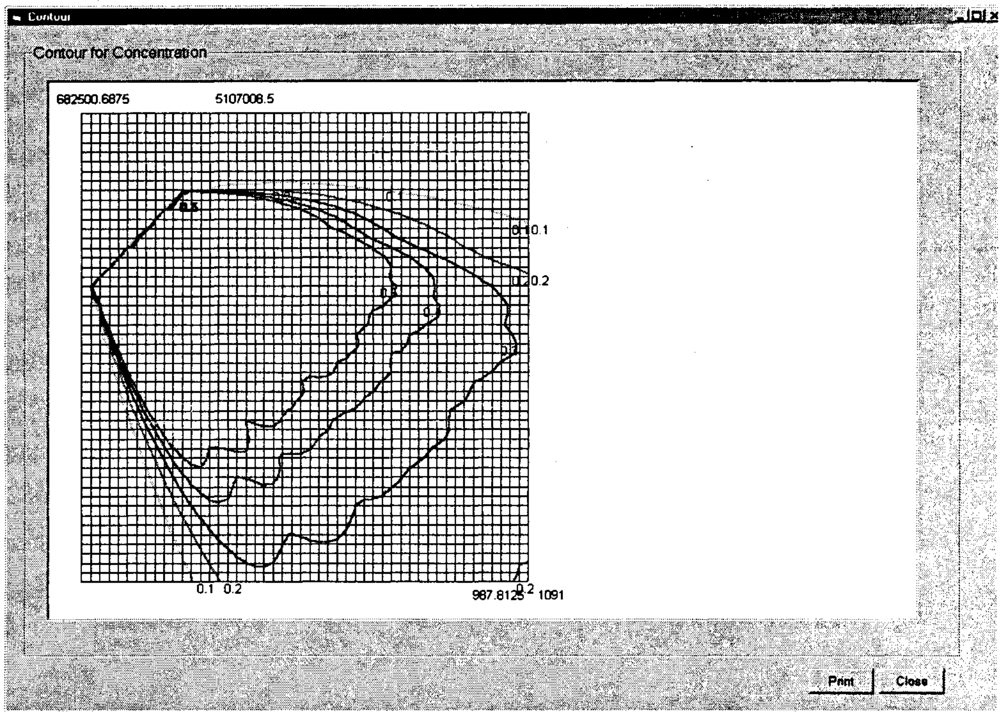


Figure 6.5: An example of contours of concentration generated by OdorImp

If a contour extends beyond the study area, OdorImp will provide a message to remind users to enlarge the study area that has been modelled using the dispersion model that is used to generate input data in order to cover the full extent of the contour. Otherwise odour impact parameters based on the extent of the selected contour would not be accurate.

Upon selection of the *Impact Parameters* tab, the interface shown in Figure 6.6 is shown. Users use this window to evaluate all odour impact parameters discussed in Chapter 4. Users select the parameters that they are interested in by choosing from the three option controllers and three check boxes and then pressing the *Run* command button. The user can press the *Save* button to save all results into a text file.

Figure 6.7 shows an example of the calculation of impact parameters related to concentration. The maximum concentration, C_{max}, and its position and the maximum population-weighted concentration (NC_{max}) and its position are generated as outputs from the model. Impact parameters including footprint area, $F(C)$, concentration-weighted footprint area ($F_W(C)$), population-weighted concentration ($N_W(C)$), and population in a concentration contour ($N(C)$) are presented in four grid tables, respectively.

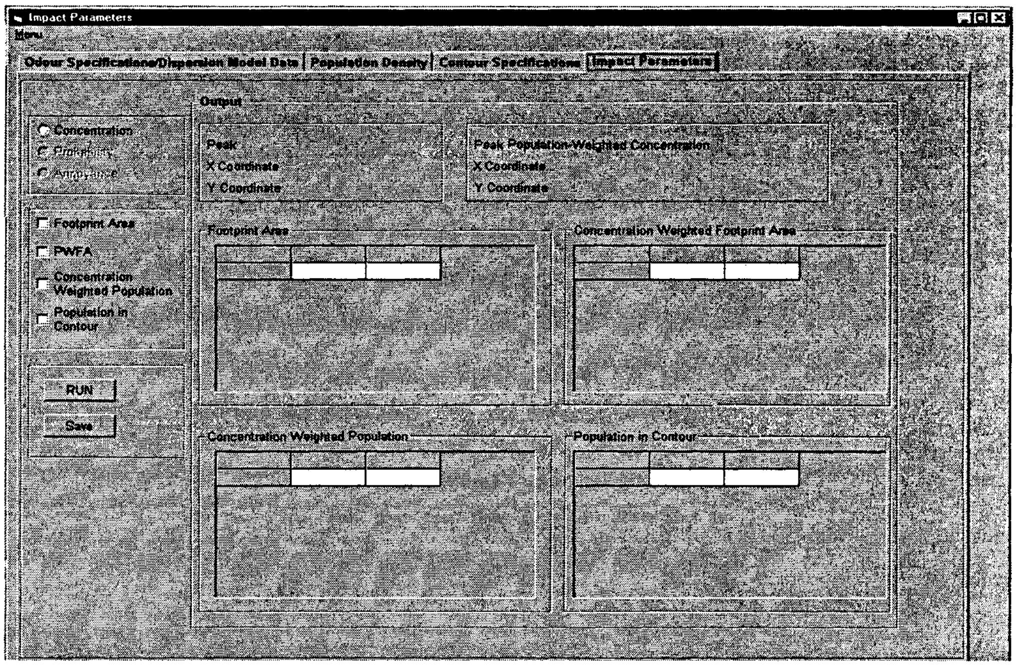


Figure 6.6: ***Impact Parameters*** **window**

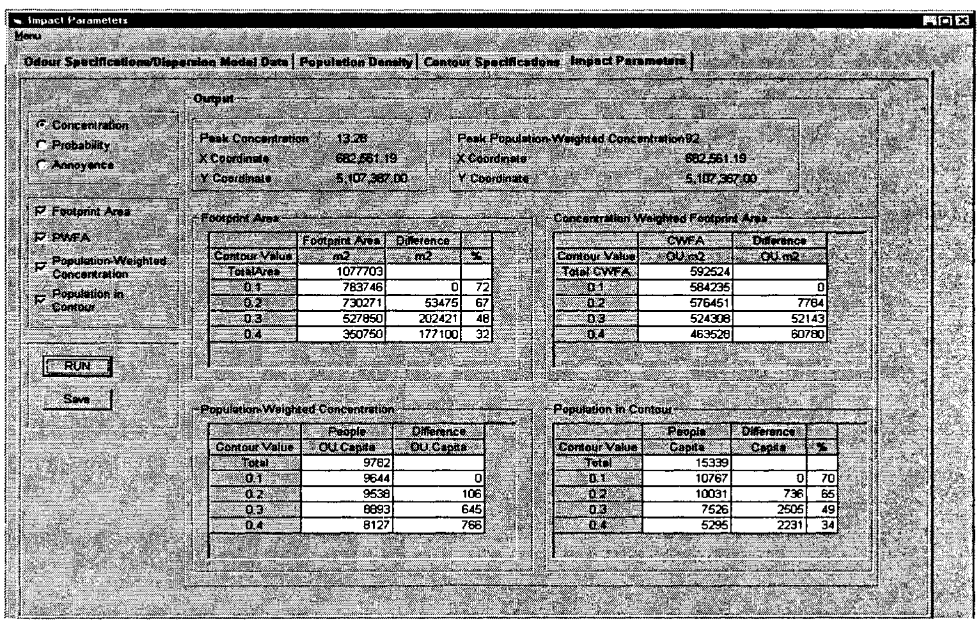


Figure 6.7: An example of ***Impact Parameters*** **window**

In the table of footprint area, the total area of the study area is shown. Also, the differences between two adjacent contours are shown in the third column. The percentage of the footprint area of a contour relative to the total area is shown in the fourth column. These values may be useful to the user in assessing the proportion of contribution of each contour region to the total impact parameter, which would aid them in identifying regions that are most susceptible to odour impact. The table reporting the populations contained within various concentration contours has the same format. The percentage of the population in a contour relative to the total population in the study area is presented in the fourth column and provides the user with the means to identify those regions which contribute most to the total population impact. Similarly, the table of concentration-weighted footprint areas lists the total concentration weighted footprint area, $F_{WC}(T)$, each $F_W(C)$, and the difference between two adjacent contours. Also, the table of population-weighted concentration lists the total population weighted concentration, $N_{WC}(T)$, each $N_W(C)$, and the difference between two adjacent contours.

Upon selection of the item *Gaussian* in the Menu, the interface in Figure 6.8 is revealed. This capability has been included in the software in order to provide the user with the means for generating simple data that can be used for experimenting with the software and its capabilities, prior to using the software in conjunction with more complex dispersion models. For example, the user can experiment with different stack heights, wind velocities, emission rates and stability classes to evaluate their impact on the values of the impact parameters.

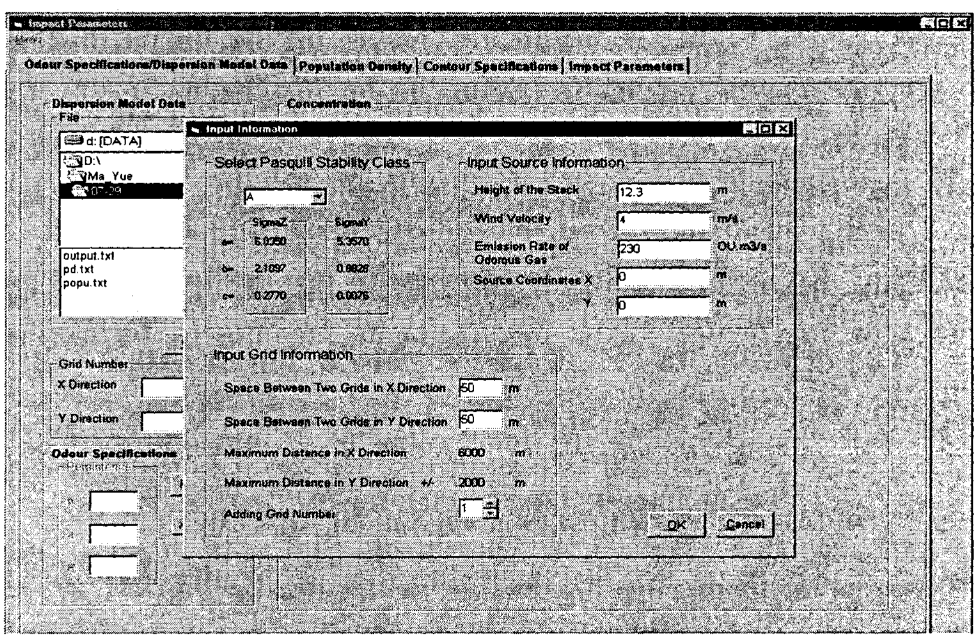


Figure 6.8: *Gaussian* window

There are some differences between using the *Gaussian* model option and reading data from an existing file. Firstly, normally the number of grid lines in an existing file is decided by the software that users choose for dispersion modelling, such as ISC-Aermod. However, under the *Gaussian* option, by default the number of grids is 121 in the *X* direction and 81 in the *Y* direction. Secondly, normally the grid spacing in an existing file that has been generated by a dispersion model is decided upon by the modelling software that user has used. In such cases, the grids may be evenly or unevenly spaced. Under the *Gaussian* option, users can input the grid spacing that they wish in the *Gaussian* model. The grid spacing may be different in *X* and *Y* direction; however, they must be distributed evenly. For the Gaussian model, users can double or triple the number of grids by pressing the *Add Grid Lines* button to increase the range of studying area. The number of grid lines can be increased to an extent that is limited only by the memory of the computer; however, too many grids can dramatically increase the computation time that is required without significantly improving the accuracy of the parameter evaluations. Finally, users need to input the information describing the odour source (e.g., stack height, emission rate, etc.) and choose a *Pasquill Stability Class*. Note that the source emission rate must be specified in $OU \cdot m^3/s$, which may be calculated by multiplying the volumetric emission rate of the source in m^3/s by the source concentration of the odour, expressed in OU. This results in grid concentrations expressed in OU. After pressing the *OK* button, the concentrations at each grid point are calculated and are displayed in the grid. Once these concentrations have been calculated, all options that are normally available for the calculation of odour impact parameters become available. Figure 6.9 shows an example of some contours generated by the *Gaussian* model.

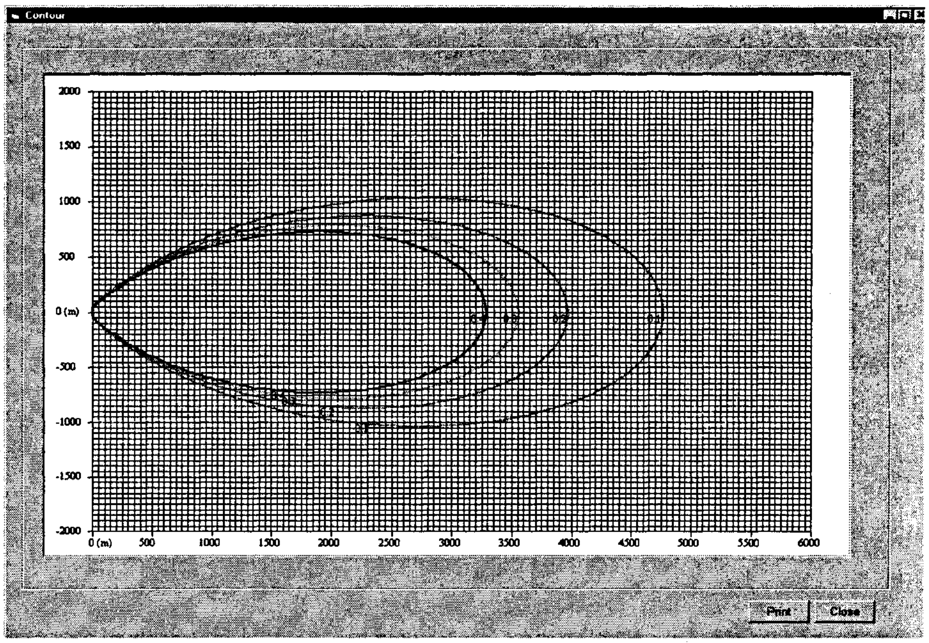


Figure 6.9: Example of contours of Gaussian model generated by OdorImp

6.2 Case studies

6.2.1 Case study 1: localized impact of a hog farm

Gorgy (2003) studied the odour impact of a pig farm located in the Eastern Townships to the East of Montreal, Quebec, Canada. This pig farm consisted of two end-to-end hog barns with a total of 24 fans that ventilated the livestock houses. Gorgy modelled these fans as separate point sources. In one of his series of analyses, Gorgy generated a 50 × 50 grid in a 1091 m × 987 m study area and used ISC to predict the 1-hour time-averaged (short term) concentrations in OU of each grid. This grid covered a region that was immediately adjacent and downwind of the hog barn. The output file containing the *X* and *Y* coordinates of grid receptors and the modelled odour concentrations became the input file for OdorImp. The persistence of response, *p*, of the odour was 0.35 (Gorgy, 2003).

Impact parameters of concentration and probability were chosen for analysis. Table 6.1 gives the peak values and their *X* and *Y* coordinates in the study area. Figure 6.10 shows the contours of concentration. Note that the contours have a shape that reflects the concentrations resulting from the parallel emissions from two adjacent (end to

Table 6.1: Peak values for case study 1

Position (m)		Concentration	Probability
X	Y	(OU)	(%)
682561.1875	5107387	13.3	99.2

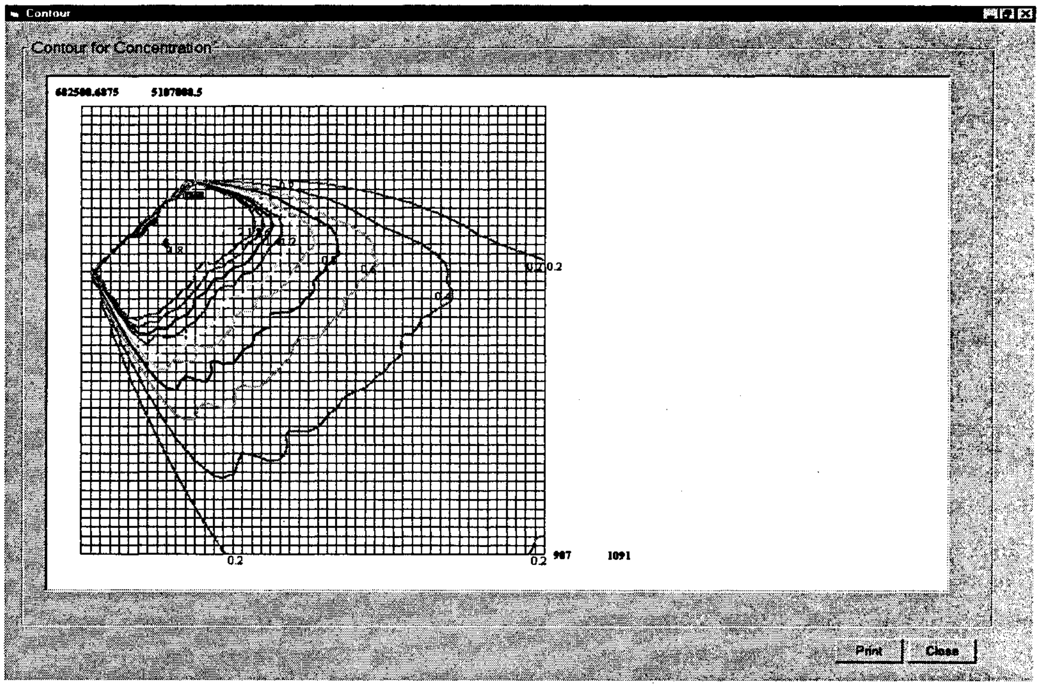


Figure 6.10: Contours of concentration for case study 1

end) barns. The geometry of these contours is less regular than those used in testing in Chapter 5. The total area of the study area was 1,077,703 m^2. Table 6.2 gives the footprint areas that calculated by OdorImp and Surfer and the absolute differences and relative differences between them. Table 6.2 also shows the percentage of each footprint area, *F(C)*, relative to the total study area. Figure 6.11 shows relative differences between footprint areas evaluated by OdorImp and Surfer for selected concentration contours.

Figure 6.12 shows the contours of probability and Table 6.3 gives the footprint areas of probability contours, *F(P)*, calculated by OdorImp and Surfer and the absolute differences and relative differences between them. Figure 6.13 shows the relative difference of *F(P)* between OdorImp and Surfer.

Table 6.2: Comparison of footprint areas of concentration contours, *F(C)*, for case study 1 evaluated by OdorImp and Surfer

Contour (OU)	*F(C)* (m^2) OdorImp	*F(C)* (m^2) Surfer	Difference (m^2)	Relative difference (%)	% of Total study area
0.2	730271	732169	-1898	-0.26	67.8
0.4	350750	352368	-1618	-0.46	32.5
0.6	218856	220336	-1480	-0.67	20.3
0.8	161362	162614	-1252	-0.77	15.0
1.0	127269	128526	-1257	-0.98	11.8
1.2	104447	105514	-1067	-1.01	9.7
1.4	87965	88849	-884	-0.99	8.2
1.6	75734	76538	-804	-1.05	7.0
1.8	65726	66469	-743	-1.12	6.1
2.0	57262	58049	-787	-1.36	5.3
2.2	50310	50909	-599	-1.18	4.7
2.4	43808	44378	-570	-1.28	4.1
2.6	38866	39221	-355	-0.91	3.6
2.8	34649	35023	-374	-1.07	3.2
3.0	31188	31438	-250	-0.80	2.9
3.2	28084	28399	-315	-1.11	2.6
3.4	25517	25865	-348	-1.35	2.4
3.6	23199	23763	-564	-2.37	2.2
3.8	21462	21940	-478	-2.18	2.0
4.0	19923	20321	-398	-1.96	1.8

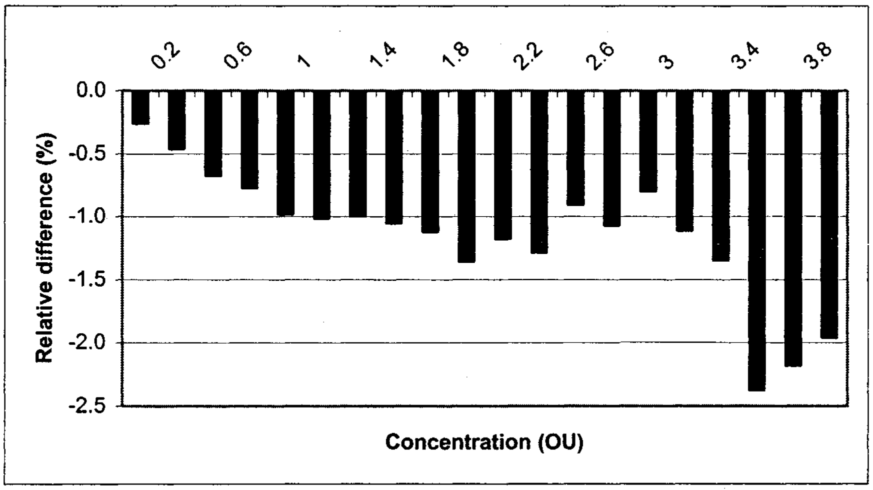


Figure 6.11: Relative differences between OdorImp and Surfer for footprint areas inside concentration contours, *F(C)*, for case study 1

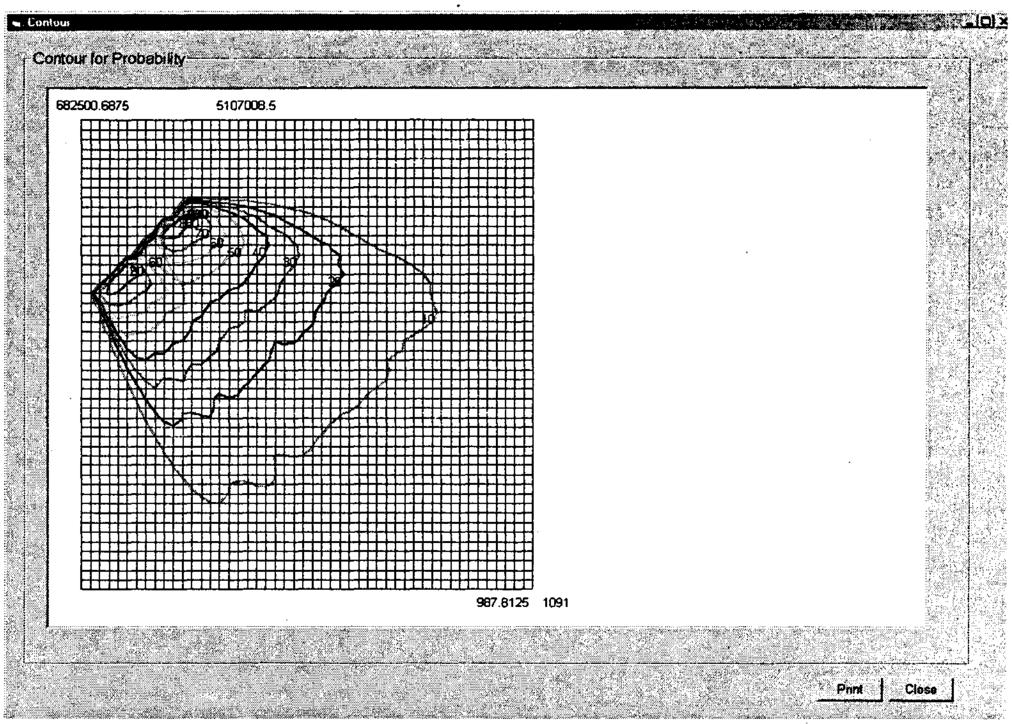


Figure 6.12: Contours of probability for case study 1

Table 6.3: Comparison of footprint areas of probability contours, *F(P)*, for case study 1 evaluated by OdorImp and Surfer

Contour (%)	*F(P)* (m^2) OdorImp	*F(P)* (m^2) Surfer	Difference (m^2)	Relative difference (%)	% of Total study area
10	504455	508130	-3675	-0.72	46.8
20	282133	284566	-2433	-0.85	26.2
30	204718	206203	-1485	-0.72	19.0
40	158609	159466	-857	-0.54	14.7
50	125072	125614	-542	-0.43	11.6
60	97957	97924	33	0.03	9.1
70	74159	73986	173	0.23	6.9
80	50395	50067	328	0.66	4.7
90	24399	23851	548	2.30	2.3

Table 6.4 gives the results of total concentration weighted footprint area, $F_{WC}(T)$ in OU·m^2, and total probability weighted footprint area, $F_{WP}(T)$ in m^2, determined using OdorImp and Surfer and the differences between them.

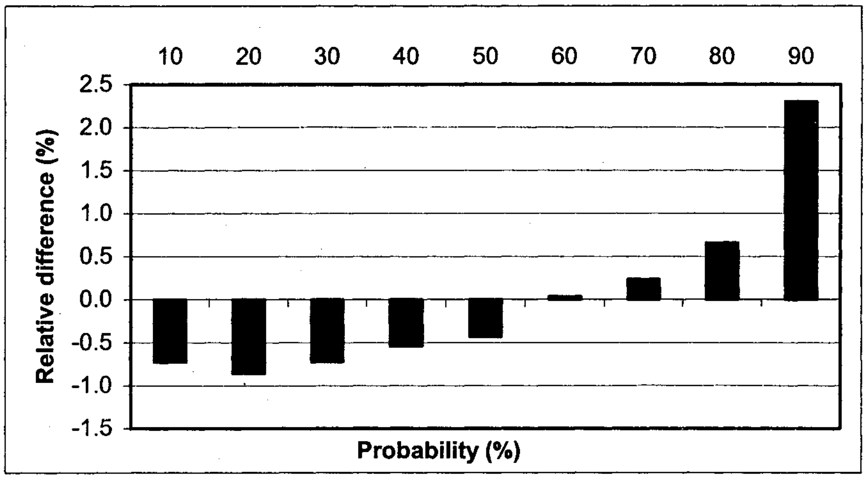


Figure 6.13: Relative differences between OdorImp and Surfer for footprint areas inside probability contours, *F(P)*, for case study 1

Table 6.4: Total concentration-weighted footprint area, $F_{WC}(T)$, and total probability-weighted footprint area, $F_{WP}(T)$, of case study 1 as determined by OdorImp and Surfer

Impact parameter	Footprint evaluated		Difference	Relative difference (%)
	OdorImp	Surfer		
$F_{WC}(T)$ (OU·m^2)	592524	593663	-1139	-0.19
$F_{WP}(T)$ (m^2)	19319800	19414897	-95097	-0.49

Table 6.5 gives the results of concentration-weighted footprint areas, $F_W(C)$ in OU·m^2, determined using OdorImp and Surfer and the differences between them. Figure 6.14 shows the trends in the relative differences between OdorImp and Surfer.

Table 6.6 gives the results of probability-weighted footprint area, $F_W(P)$ in m^2, from OdorImp and Surfer and the differences between them. Figure 6.15 shows the tendencies in the relative differences between OdorImp and Surfer as a function of the size of the contour. Similar to the footprint areas of probability contours.

Table 6.5: Comparison of concentration-weighted footprint areas, $F_W(C)$, for case study 1 evaluated by OdorImp and Surfer

Contour (OU)	$F_W(C)$ ($OU\cdot m^2$)		Difference ($OU\cdot m^2$)	Relative difference (%)
	OdorImp	Surfer		
0.2	576451	581588	-5137	-0.88
0.4	463528	468557	-5029	-1.07
0.6	399862	404875	-5013	-1.24
0.8	359118	365060	-5942	-1.63
1	328897	334652	-5755	-1.72
1.2	303829	309501	-5672	-1.83
1.4	282464	287943	-5479	-1.90
1.6	264133	269530	-5397	-2.00
1.8	247094	252448	-5354	-2.12
2	230997	236466	-5469	-2.31
2.2	216321	221496	-5175	-2.34
2.4	201436	206473	-5037	-2.44
2.6	189148	193601	-4453	-2.30
2.8	177698	182278	-4580	-2.51
3	167659	171889	-4230	-2.46
3.2	158158	162476	-4318	-2.66
3.4	149633	154122	-4489	-2.91
3.6	141674	146769	-5095	-3.47
3.8	135179	140029	-4850	-3.46
4	129129	133718	-4589	-3.43

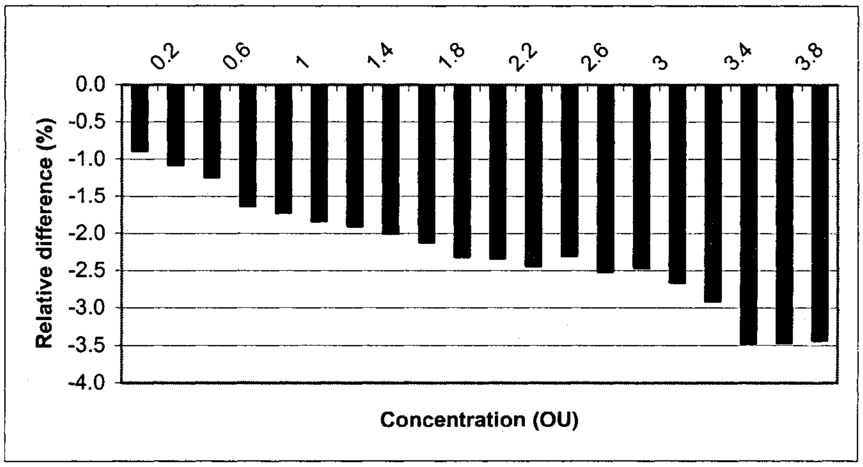


Figure 6.14: Relative differences between OdorImp and Surfer for concentration-weighted footprint area, $F_W(C)$, for case study 1

Table 6.6: Comparison of probability-weighted footprint area, $F_W(P)$, for case study 1 evaluated by OdorImp and Surfer

Contour (%)	$F_W(P)$ (m^2) OdorImp	$F_W(P)$ (m^2) Surfer	Difference (m^2)	Relative difference (%)
10	17358500	17503395	-144895	-0.83
20	14282100	14410069	-127969	-0.89
30	12391900	12496381	-104481	-0.84
40	10793500	10876991	-83491	-0.77
50	9296300	9361944	-65644	-0.70
60	7808400	7843612	-35212	-0.45
70	6271500	6290553	-19053	-0.30
80	4486700	4494029	-7329	-0.16
90	2282200	2267163	15037	0.66

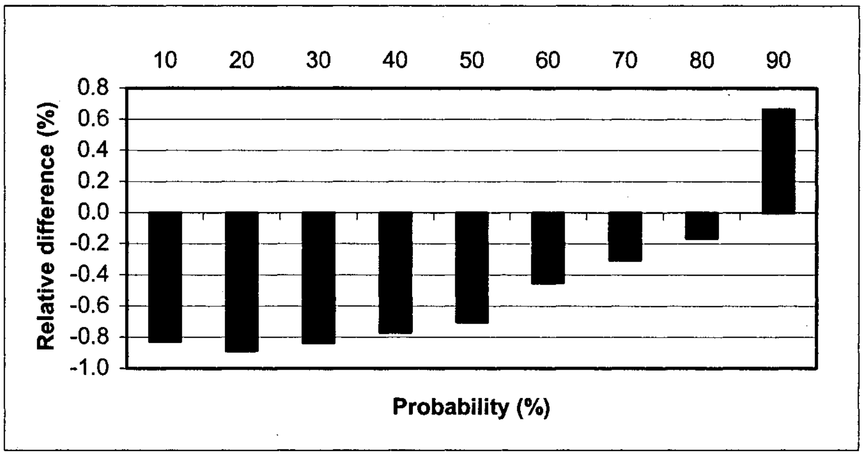
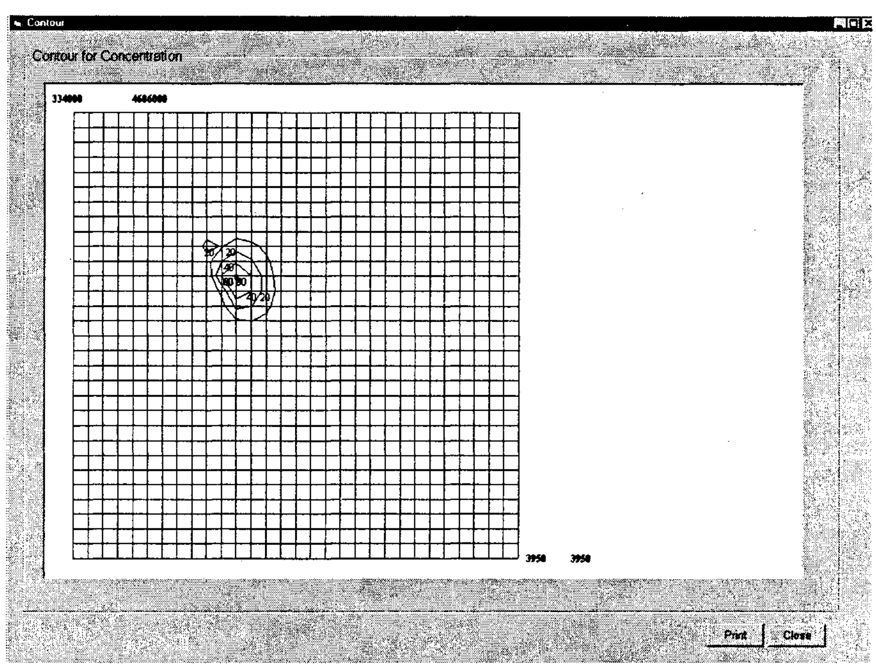


Figure 6.15: Relative differences between OdorImp and Surfer for probability-weighted footprint area, $F_W(P)$, for case study 1

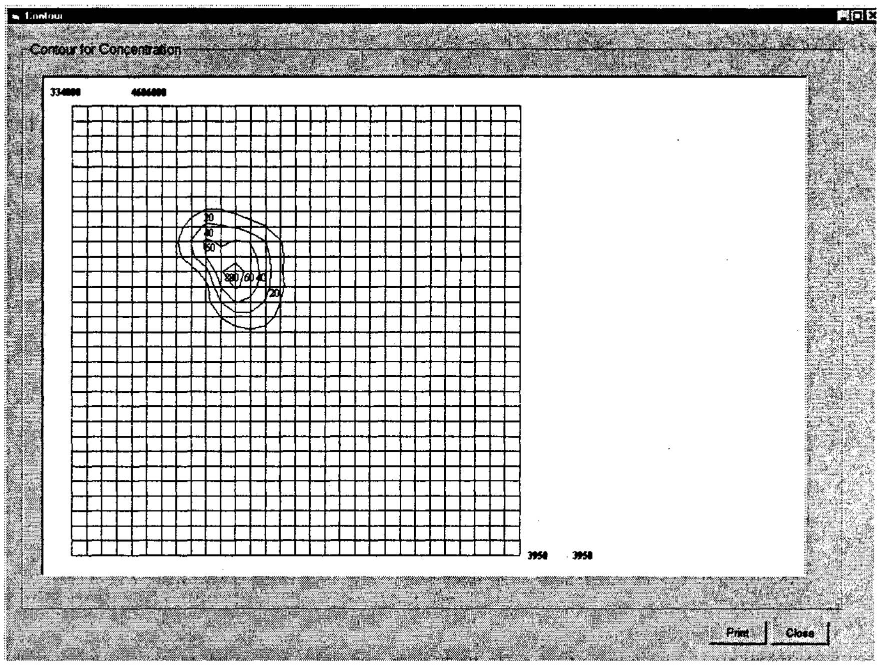
6.2.2 Case study 2: industrial facility in an urban environment

The data for this case study were obtained through Professor P. Henshaw of the University of Windsor who supervised the work of Sikdar (2001). ISC was used to evaluate the odour impact of an industrial facility in a 4 km by 4 km study region, with a grid spacing of 0.05 km. 5-years of hourly meteorological data were used to evaluate the range of impacts that could be experienced at all grid points. The data generated from the model included the peak, 90th, 95th, 99th percentile concentrations at each grid point. The odour had a persistence of 0.30. In this case study, the dispersion modelling results were used in conjunction with OdorImp to predict the peak, 90th, 95th, 99th percentile of probabilities for comparison with the results of Sikdar, who used Surfer to evaluate odour impact parameters. Due to the limitation of data that was available, only parameters such as footprint area of concentration and probability, $F(C)$ and $F(P)$, concentration-weighted footprint area, $F_W(C)$, probability-weighted footprint area, $F_W(P)$, total concentration-weighted footprint area, $F_{WC}(T)$, and the total probability weighted footprint area, $F_{WP}(T)$, could be evaluated for comparison. Parameters related to annoyance could not be analyzed because the ratio of D_{5au} over $D_{50\%}$, R, for the odour emissions from this industrial facility were not measured. In addition, parameters related to population could not be analyzed using OdorImp because the population density values used by Sikdar (2001) were unknown.

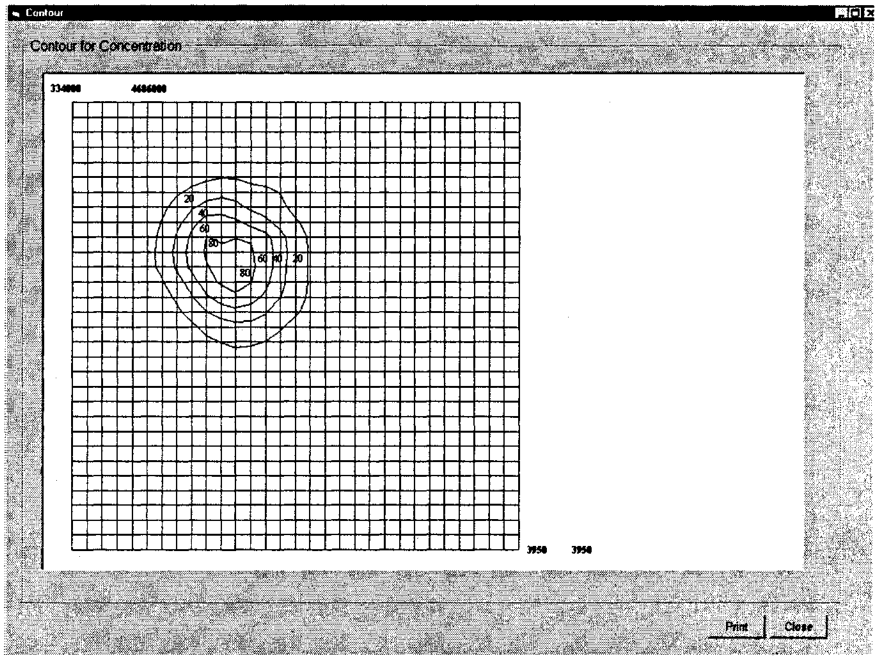
Figure 6.16 shows the contours of 90[th], 95[th] and 99[th] percentiles of probability of response for case study 2 that were drawn by OdorImp. Figures 6.17 and 6.18 show the contours of peak hourly concentration and peak hourly probability of response also drawn by OdorImp. Note that these concentration contours are very irregular in shape, as compared to the highly ideal geometries tested in Chapter 5 and the less ideal geometry seen in case study 1 above. This complex geometry is considered to be the best test of the capability of the algorithms in OdorImp because of the highly distributed nature of the footprints and the volumes under the contours.



(a)



(b)



(c)

Figure 6.16: Contours of (a) 90th, (b) 95th and (c) 99th percentile of probability of response for case study 2

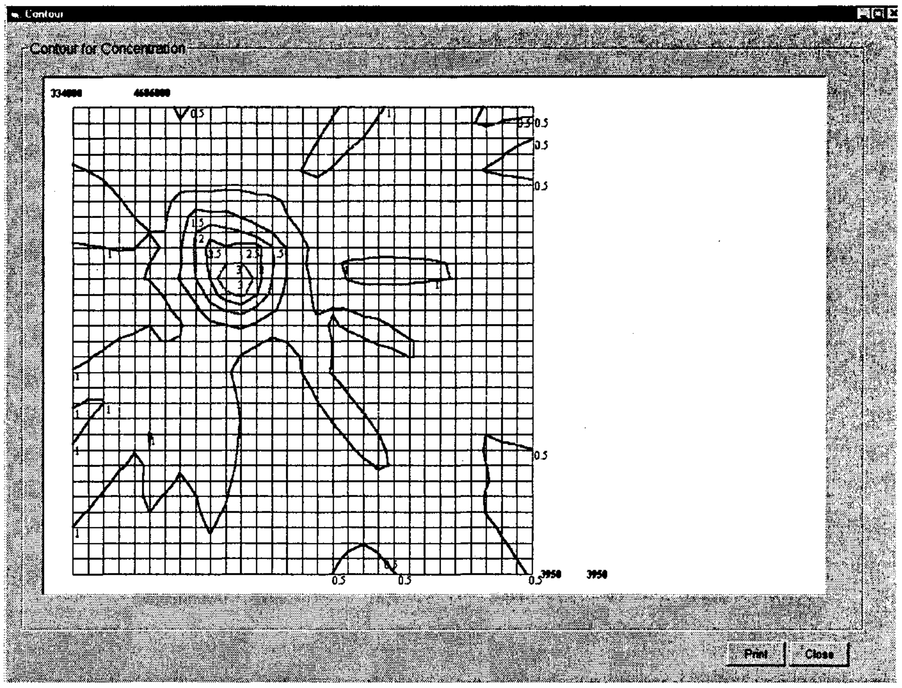


Figure 6.17: Peak hourly concentration contours (in OU) for case study 2

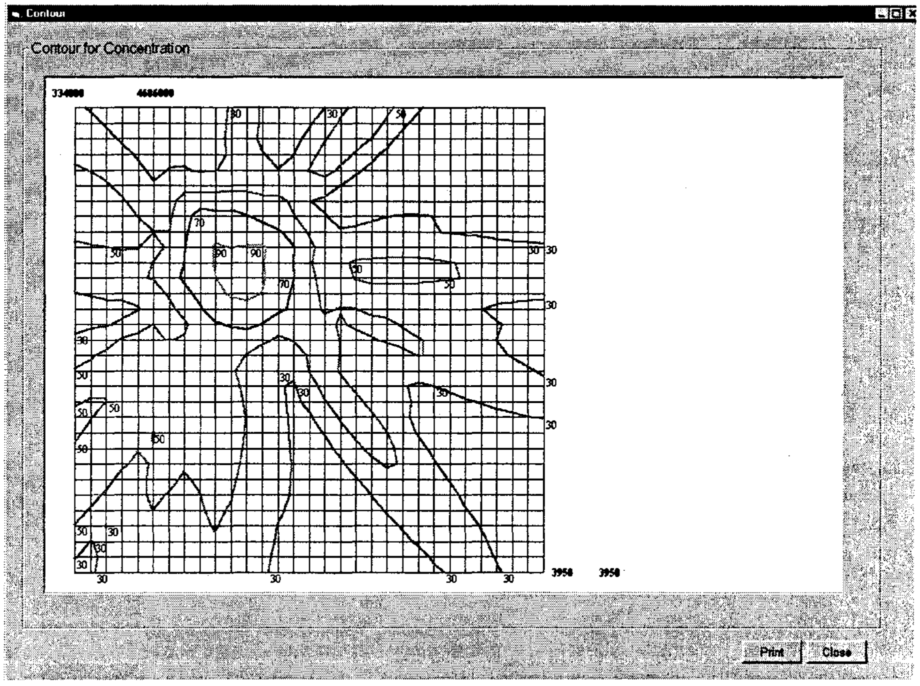


Figure 6.18: Peak hourly probability contours (in %) for case study 2

Table 6.7 lists the footprint areas inside selected probability contours, *F(P)*, that were determined for the 90th, 95th, 99th percentiles and peak hourly probabilities calculated using OdorImp and Surfer and the differences between them.

Table 6.7: Footprint areas inside probability contours, *F(P)*, for 90th, 95th, 99th percentile and peak hourly probabilities for case study 2 as evaluated by OdorImp and Surfer

Contour (%)	Percentiles									Peak hourly		
	90th			95th			99th					
	F(P) (m²)		Relative difference (%)	*F(P)* (m²)		Relative difference (%)	*F(P)* (m²)		Relative difference (%)	*F(P)* (m²)		Relative difference (%)
	OdorImp	Surfer		OdorImp	Surfer		OdorImp	Surfer		OdorImp	Surfer	
10	525141	537919	-2.4	1080465	1100693	-1.85	2462649	2482310	-0.79	15602500	15602500	0.00
20	321390	321874	-0.15	697195	709696	-1.76	1560957	1567837	-0.44	14065145	14095792	-0.22
30	207428	212448	-2.4	515583	523279	-1.47	1113967	1125398	-1.01	10890360	10900165	-0.09
40	138546	143146	-3.2	387403	390896	-0.89	843695	849257	-0.65	7735449	7736664	-0.01
50	89711	90157	-0.49	280769	277996	0.99	641333	644317	-0.46	4396032	4337587	1.35
60	49821	48883	1.92	184667	175703	5.10	481438	481846	-0.08	1538462	1530861	0.50
70	18975	16572	14.50	83289	83669	-0.45	332619	329090	1.07	761741	765237	-0.46
80	1060	888	19.37	23729	21271	11.55	160976	158418	1.61	482476	481493	0.20
90	0	0	0	0	0	0	11073	9842	12.50	175338	174101	0.71

Table 6.8: Probability weighted footprint areas inside probability contours, $F_W(P)$, for 90th, 95th, 99th percentile and peak hourly probabilities for case study 2 as evaluated by OdorImp and Surfer

Contour (%)	Percentiles									Peak hourly		
	90th			95th			99th					
	$F_W(P)$ (m²)		Relative difference (%)	$F_W(P)$ (m²)		Relative difference (%)	$F_W(P)$ (m²)		Relative difference (%)	$F_W(P)$ (m²)		Relative difference (%)
	OdorImp	Surfer		OdorImp	Surfer		OdorImp	Surfer		OdorImp	Surfer	
10	16195249	16160853	0.21	37679377	37779590	-0.26	87479411	87619685	-0.16	634325483	634289670	0.00
20	13176627	13030274	1.12	32148231	32228616	-0.25	74619248	74715544	-0.13	608133493	608550494	-0.07
30	10287993	10361978	-0.71	27655635	27662379	-0.02	63521214	63834761	-0.49	529451996	529505733	-0.01
40	7847140	7956552	-1.37	23135906	23056111	0.35	54110839	54263274	-0.28	418201447	418182126	0.00
50	5642625	5579180	1.14	18299369	17988958	1.72	45001866	45082322	-0.18	267981151	265089864	1.09
60	3433440	3319748	3.42	13022056	12353005	5.42	36227181	36167460	0.16	113014583	112804230	0.19
70	1417932	1232818	15.01	6428600	6423501	0.08	26509743	26227767	1.07	63323745	63530239	-0.32
80	85903	71919	19.44	1969732	1761457	11.82	13646784	13403423	1.82	42397229	42268287	0.30
90							1005552	893686	12.52	16180076	16061164	0.74

Figure 6.19 shows the tendencies in relative differences of footprint area between OdorImp and Surfer as a function of the contour value.

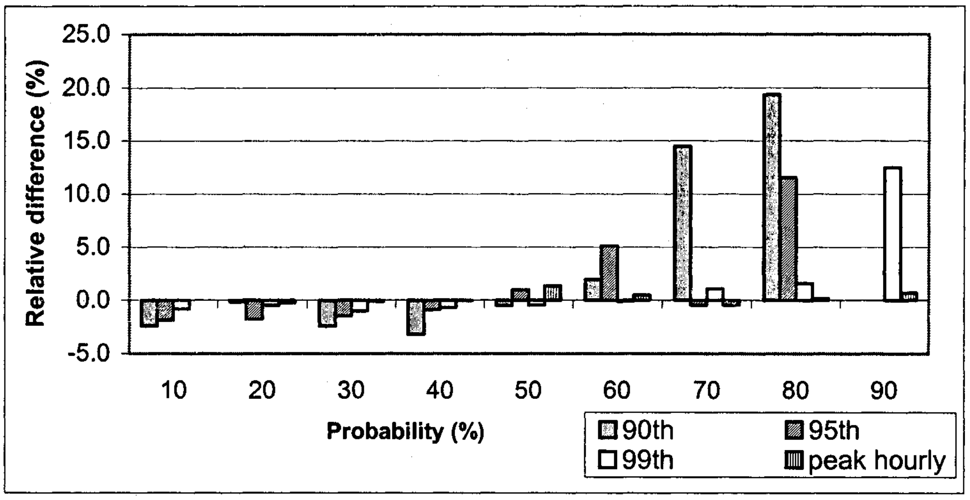


Figure 6.19: Relative differences in footprint areas inside probability contours, *F(P)*, for 90th, 95th, 99th, and peak hourly probabilities for case study 2 evaluated using OdorImp and Surfer

Table 6.8 lists the results of probability-weighted footprint area, $F_W(P)$, for 90th, 95th, 99th percentile and peak hourly probabilities evaluated using OdorImp and Surfer and the relative differences between them. Figure 6.20 shows the tendencies in the relative differences of $F_W(P)$ between OdorImp and Surfer.

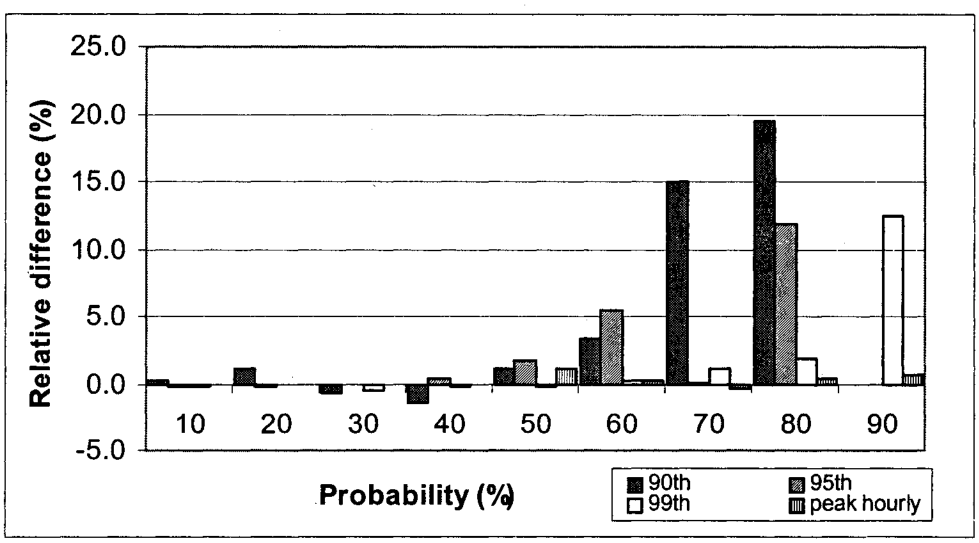


Figure 6.20: Relative differences in probability-weighted footprint areas inside probability contours, $F_W(P)$, for 90th, 95th, 99th, and peak hourly probabilities for case study 2 evaluated using OdorImp and Surfer

Table 6.9 gives the total probability-weighted footprint areas, $F_{WP}(T)$, of 90th, 95th, 99th percentile and peak hourly calculated from OdorImp and Surfer and their relative differences. The results from OdorImp are very close to the ones from Surfer.

Table 6.9: Total probability-weighted footprint area, $F_{WP}(T)$, for 90th, 95th and 99th percentiles and peak hourly probabilities for case study 2 as determined by OdorImp and Surfer

	$F_{WP}(T)$ (m^2)		Relative difference (%)
	OdorImp	Surfer	
90th	21268127	21268081	-0.00022
95th	48858559	48858404	-0.00063
99th	120357944	120378221	0.0168
Peak hourly	634325483	634403905	0.0124

Table 6.10 compares the results of footprint area of probability contour, $F(P)$, calculated from OdorImp and by Sikdar (2001). Note that the results of Sikdar (2001) had been rounded off. The results from OdorImp are very close to those of Sikdar (2001).

Table 6.10: Footprint areas inside selected probability contours, $F(P)$, for 90th, 95th and 99th percentiles and peak hourly probabilities for case study 2 as calculated using OdorImp and reported by Sikdar (2001)

Contour (%)	$F(P)$ (m^2)							
	Percentile						Peak hourly	
	90th		95th		99th			
	OdorImp	Sikdar	OdorImp	Sikdar	OdorImp	Sikdar	OdorImp	Sikdar
10	525141	540000	1080465	1100000	2462649	2500000	15602500	16000000
50	89711	90000	280769	280000	641333	640000	4396032	4300000
90	0	0	0	0	11073	9800	175338	170000

Table 6.11 provides a comparison of the probability-weighted footprint area, $F_W(P)$, from OdorImp with Sikdar (2001). It should be noted that Sikdar (2001) defined this footprint slightly differently than the definition used here. That is, Sikdar (2001) excluded the volume of the cylinder under the bounding contour (as shown in Figure 2.8). Therefore, in order to account for this difference and to allow a comparison of results, $F_W(P)$ values listed in the Table 6.11 were calculated by subtracting the volume of the

cylinder defined by the bounding contour from the value reported by OdorImp. As can be seen, the results from OdorImp and Sikdar are very similar.

Table 6.11: Probability-weighted footprint areas inside selected probability contours, $F_W(P)$, for 90th, 95th and 99th percentiles and peak hourly of probability of response for case study 2 as calculated using OdorImp and reported by Sikdar (2001) Note that in this table, $F_W(P)$ does not include the volume of the cylinder under the contour, as had been calculated by Sikdar (2001). The values reported by Sikdar (2001) have been rounded off.

Contour (%)	$F_W(P)$ (m^2)							
	Percentile						Peak hourly	
	90th		95th		99th			
	OdorImp	Sikdar	OdorImp	Sikdar	OdorImp	Sikdar	OdorImp	Sikdar
10	109438	110000	268747	270000	628529	630000	4783004	4800000
50	11570	11000	42609	41000	129352	130000	481795	480000
90	0	0	0	0	89	79	3996	3900

6.3 Discussions

As can be seen from Table 6.2 and Figure 6.11, the footprint areas in concentration contours, $F(C)$, that OdorImp calculated are consistently close to but smaller than the ones determined using Surfer. Though the absolute differences become smaller with smaller footprint areas, as would be expected, the relative differences don't have the same tendency. The relative differences are somewhat variable with a general tendency to become greater with smaller footprint areas. This would be the consequence of having fewer and fewer grid points with smaller footprint areas. Similar to the footprint areas of concentration contours, the results of concentration-weighted footprint areas, $F_W(C)$, from OdorImp are all smaller than the ones from Surfer. The relative differences of $F_W(C)$ become greater when the contour areas are smaller (see Table 6.5 and Figure 6.14).

As can be seen in Table 6.3 and Figure 6.13, the absolute and relative differences of footprint areas in probability contours, $F(P)$, between OdorImp and Surfer tend from negative to positive with decreasing footprint size, which is different from F(C). The relative differences of $F_W(P)$ between OdorImp and Surfer also follow a trend ranging from negative to positive for decreasing contour size (Table 6.5 and Figure 6.15).

All relative differences between OdorImp and Surfer in case study 1 range from -4 % to 3 % which is quite reasonable. Moreover, the relative differences of the total concentration weighted footprint area, $F_{WC}(T)$, and the total probability weighted footprint area, $F_{WP}(T)$, between OdorImp and Surfer are even better, with relative differences of - 0.19% and - 0.49%, respectively. Thus, the results calculated from OdorImp are very close to those of Surfer and are in a reasonable range.

For case study 2, the differences between OdorImp and Surfer are not as regular as case study 1. From Table 6.7 the maximum relative difference of footprint areas is 19.4% for *F(80%)* of 90th percentile of probability. However, the relevant absolute difference is only 172 m^2, which is small compared to the total area of the study area (i.e., 15,602,500 m^2). Other notable instances of high relative differences include: (1) for *F(70%)* in the 90th percentile evaluation, the relative difference is about 14.5% and its absolute difference is about 2403 m^2; (2) for F(80%) in the 95th percentile evaluation, the relative difference is about 11.6% and its absolute difference is 1459 m^2; and (3) for *F(90%)* in the 99th percentile evaluation, the relative difference is about 12.5%, and its absolute difference is 2230 m^2

The results of probability-weighted footprint area in case study 2, $F_W(P)$, are similar to those of the footprint area, *F(P)*. Even if the relative differences are large, such as $F_W(80\%)$ in 90th percentile evaluation, where the relative difference is a maximum of 19.5%, its absolute difference is only 13,984 m^2 . This is approximately equal to the area of square of 120 m × 120 m (i.e., one that can be represented by less than a 3 by 3 grid of 50 m grid cells). This absolute difference can be considered negligible relative to the study area of 15,602,500 m^2. Among other high relative differences, 15.01% for $F_W(70\%)$ in the 90th percentile evaluation, 11.82% for $F_W(80\%)$ in the 95th percentile evaluation, and 12.52% for $F_W(90\%)$ in the 99th percentile evaluation, the maximum absolute difference is 208,275 m^2. This absolute difference is still not large relative to the study area and is acceptable.

The worst case situations of *F(P)* and $F_W(P)$ occurred in the 90th percentile which have relative small areas. *F(P)* and $F_W(P)$ in peak hourly probabilities which have the biggest area sizes always have the best results. All total probability-weighted footprint areas for the four situations, $F_W(P)$, are fairly small.

Notably all of the particular high relative differences noted above occured when the footprint area was very small relative to the total area being modelled. That is, the smaller the ratios of footprint area to total area, the larger the differences tend to be. Thus, these instances reflect footprints or volumes that have been evaluated using very few grid points. Such evaluations are subject to a great deal of error by both Surfer and OdorImp and, thus, it is not surprising to observe large differences between the results of the two programs. It is recommended that the algorithms used in OdorImp could be improved to provide the user with a warning that under instances where the footprint area is small relative to the larger area, the footprint results are likely to be subject to significant error and a smaller grid spacing should be used to improve the accuracy of the parameter values.

The results of the testing of the algorithms using ideal geometries and a simple Gaussian model (see Chapter 5) has demonstrated that OdorImp is reliable and is at least as accurate as the commercial package Surfer 7.0. In addition, the comparison of the results from the two case studies using OdorImp and Surfer and comparing them with previous results of Sikdar, has shown that OdorImp can also be applied with confidence to situations in which contour geometries are not simple.

OdorImp can be used to calculate all proposed odour impact parameters easily. The interface is quite simple and user-friendly. However, OdorImp currently has some limitations that should be addressed. Firstly, the manner in which population density is handled in the program is somewhat complicated and unsatisfactory Firstly, OdorImp currently requires the population density to be specified in uniform rectangular blocks. However, this is very unlikely in most real situations. Therefore, a better way should be devised to deal with handling heterogeneous distributions of population density. Secondly, the current version of OdorImp has some functions to deal with errors that are created by users' operation. However, due to time limitations, it was not possible to create a program that would respond to all possible errors that could be encountered in the day to day usage of OdorImp. Such errors may lead to the failure of OdorImp, resulting in a loss of data. Additional testing must be conducted to identify potential sources of user errors and to allow OdorImp to automatically compensate for those errors.

7 Conclusions and Recommendations

7.1 Conclusions

Based on the approach taken by previous researchers, a series of potential odour impact parameters were developed. These include point values, footprint areas and volume parameters (i.e., weighted-footprint areas) which in various ways account for impacts as a function of odour concentration, probability of response, degree of annoyance and population density. In order to simplify the calculation of these parameters, algorithms were developed and implemented in Visual Basic computer code. In addition, an algorithm for drawing contours was developed to provide the means for visualizing the data that is used as a basis for parameter evaluation. The software OdorImp was created to implement all of these algorithms together in a user-friendly interface.

In order to demonstrate the ability of the software to reliably calculate odour impact parameters, three sets of synthetic data and two sets of data results from case studies were analysed using OdorImp. Parallel analyses were conducted using a commercial contouring analysis package called Surfer (version 7.0). In addition, in the cases of synthetic data generated from simple geometries (i.e., a half sphere and a taper), theoretical (i.e., exact) values for odour impact parameters were also evaluated. Also, the results of one of the case studies were compared with values that had been published in a previous study. In all cases, it was demonstrated that OdorImp consistently provided reliable estimates of areas within contours and volumes under contours, thereby confirming the ability of the software to evaluate the proposed impact parameters. It was found, however, that the calculation accuracy is strongly related to the grid spacing and that care must be taken to ensure a sufficiently small grid spacing in input data to ensure that the estimated parameter values are sufficiently accurate.

7.2 Recommendations

While it has been demonstrated that the algorithms implemented in OdorImp are reliable and accurate, additional work should be conducted to improve the user interface.

The following issues should be addressed before finalizing the user interface and releasing OdorImp for public use.

1. OdorImp currently has a very simple interface for inputting data about population density. The interface requires that the population density be specified in adjoining rectangular blocks of uniform population density. However, the actual population density in the community would rarely be satisfactorily represented using this type of interface. Further work should focus on improving the manner in which heterogeneous population densities can be input into OdorImp to overcome this limitation.
2. OdorImp provides many messages that give the user reminders or information to handle errors. However, there will be many unpredictable errors may happen when users interact with the software in unpredictable ways. Such errors can cause the program to shut down if they are not trapped by the software. Thus, intensive testing of the interface needs to be conducted and, as needed, more comprehensive methods for error trapping within OdorImp need to be implemented.
3. The accuracy of the values of odour impact parameters produced by OdorImp is a function of the grid spacing used. Methods should be developed in the software to automatically check for problems arising from parameters that are evaluated using small numbers of grid points. Such a system could warn the user of questionable values that are produced by the program and could advise them to increase the density of the grid points that are input into the program.
4. OdorImp currently labels all contours automatically. However, sometimes it is unnecessary or undesirable to label all contours. Thus, the interface could be improved by allowing the users to decide which contours they prefer labelling.

OdorImp has been designed to facilitate the evaluation of numerous parameters that could be used when conducting odour impact assessments. However, most of the odour impact parameters proposed in this study and implemented in OdorImp have never been applied before. Thus, it is recommended that the relative usefulness of these parameters be evaluated by applying OdorImp to actual odour impact situations. Such research would form the basis for determining which parameters correlate best with actual odour impacts observed in communities.

REFERENCES

Aramini, M.J. "Implementation of an Improved Contour Plotting Algorithm", Masters Thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, 1981 (Available online at http://members.bellatlantic.net/~vze2vrva/thesis.html)

Artis, D. "Odour Nuisances and their Control", Shaw & Sons Ltd., London, UK, 1984.

Beychok, M.R., "Fundamental of Gas Dispersion," 3rd ed., Published by author. Irvine, CA, 1994.

Chao, Y.L., "Kriging Interpolation". Online article, Department of Computer Science, Cornell University, 2002. (Available online at http://www.nbb.cornell.edu/neurobio/land/OldStudentProjects/cs490-94to95/clang/kriging.html)

Cheremisinoff, P.N. and Young, R.A. "Industrial Odor Technology Assessment". Ann Arbor, Michigan, Ann Arbor Science Publishers Inc., 1988.

Cottafava, G. and Le Moli, G. "Automatic Contour Plotting", Comm. ACM Vol. 12, No. 7, pp. 386-391, 1969.

Duffy, R. "Introduction to Programming with Visual Basic". Que Corporation, 1995.

Gorgy, T.G.A. "Validation of an Air Dispersion Model for Odour Impact Assessment". Master of Applied Science Thesis, Department of Civil Engineering & Applied Mechanics, McGill University, Montreal, QC, 2003.

Gostelow, P., Parsons, S.A. and Stuetz, R.M. "Odour Measurements for Sewage Treatment Works". *Water Research*, Vol. 35, No. 3, pp. 579-597, 2001.

Government of Ontario "Regulation 308, Environmental Protection Act". Revised Regulation of Ontario, 1980 as amended to O.Reg, 90/90, Queen's Printer for Ontario, ON, Canada, 1990.

Harreveld, A.P., Heeres, P., and Harssema, H. "A Review of 20 Years of Standardization of Odor Concentration Measurement by Dynamic Olfactometry in Europe". *J. Air. Waste. Manage. Assoc.,* Vol. 49, (June), pp 705-715, 1999.

Henshaw, P., Nicell, J.A, Sikdar, A. "A New Method for Odour Impact Assessment Based on Spatial and Temporal Analyses of Community Response". In Proceedings of 2002 Joint CSCE-EWRI International Conference on Environment Engineering, July 21-24, Niagara Falls, ON (on CD ROM), 2002.

Leonardos, G. "Review of Odor Control Regulations in the USA." In Odors: Indoor and Environmental Air, Proceedings of an International Specialty Conference Sponsored by the Air and Waste Management Association, (Bloomington, MN, Sept 13-15, 1995), Air and Waste Management Association, Pittsburgh, PA, pp.74-84, 1996.

National Research Council Committee on Odors "Odors from Stationary and Mobile Sources," Board on Toxicology and Environmental Hazards, Assembly of Life Sciences, National Academy of Sciences, Washington, D.C., 1979.

Nagy, G.Z. "The Odor Impact Model", *J. Air Waste Manage. Assoc.*, Vol. 41, No.10, 1991, pp. 1360-1362.

Nicell, J.A. "Preliminary Assessment of the Odour Impact Model as a Regulatory Strategy". Master of Applied Science Thesis, Department of Chemical Engineering, University of Windsor, Windsor, ON, 1986.

Nicell, J.A. "Development of the Odour Impact Model as a Regulatory Strategy," *Int. J. Env. Poll.* Vol. 4, No. 1/2, pp. 124-138, 1994.

Nicell, J.A. and Tsakaloyannis M. "A Proposed Regulatory Method for the Assessment of the Impact of Odorous Emissions from Stationary Sources," In Proceedings of the 3rd International Interdisciplinary Conference on the Environment, June 25-28, Cambridge, MA, pp.188-201, 1997.

Nicell, J.A. "Extension of Odor Impact Assessments Beyond Nuisance Evaluations and the Dilution-to-Threshold Principle", *Interdis. Env. Rev.,* Vol. 1:, No.1, pp. 45-62, 1999.

Nicell, J.A. (in press), "Expressions to Relate Population Responses to Odour Concentration". *Atm. Env.*, 2003.

Ontario Ministry of Environment. "Updating Ontario's Air Dispersion Models: A Discussion Paper". 2001.

Pootschi, E.B.M. "Development of a Strategy for Quantifying the Impact of Odorous Emissions from Stationary Sources on the Surrounding Communities," Ph.D. Dissertation, Department of Chemical Engineering, University of Windsor, Windsor, ON, 1985.

Rand, D., "Contour Plotting in Java", *Machtech*, Vol. 13, No. 9, 2002. Available online at http://www.mactech.com/articles/mactech/vol.13/13.09/ContourPlottinginJava

Schneider, D.I. "An Introduction to Programming with Visual Basic 6.0 (Fourth Edition)" Prentice-Hall. Inc., 1995-1999. Available online at http://cwx.prenhall.com/bookbind/pubbooks/schneider/chapter1/deluxe.html

Schulz, T.J., Harreveld, A.P., "International Moves Towards Standardisation of Odour Measurement Using Olfactometry", Wat. Sci. Tech., Vol. 34, No. 3-4, 1996, pp. 541-547.

Sikdar, A., "An Objective Method for the Assessment of the Impacts of Odourous Emissions from Stationary Sources", Faculty of Graduate Studies and Research, University of Windsor, Windsor, ON, 2001.

Snyder, W.V., "Algorithm 531, Contour plotting [J6]", ACM Trans. Math. Softw. Vol. 4, No. 3, 290-294, 1978.

Lakes Environmental, Inc., "User's Guide ISC-Aermod for Window", Waterloo, Ontario: Lakes Environmental Consultants, Inc., 2000.

Golden Software, Inc., "User's Guide Surfer 7.0", Golden, Colorado: Golden Software Consultants, Inc. 1999.

APPENDIX 1

Computer Code for Generating Synthetic Data

1.1 Code for generating Gaussian dispersion model data

XnumberB	Number of grids in X direction for a simple Gaussian dispersion model
YnumberB	Number of grids in Y direction for a simple Gaussian dispersion model
Xdistance	Distance between grid and source in X direction
Ydistance	Distance between grid and source in Y direction
Sourcex, SourceY	Coordination of source in X and Y direction
Concentration()	One dimensional array containing concentrations of grids
sigmaY, sigmaZ	Coefficients in Beychok equation σ_y, σ_z
S_ay, S_by, S_cy	Coefficients a, b, c for σ_y in Beychok equation
S_az, S_bz, S_cy	Coefficients a, b, c for σ_z in Beychok equation

```
For j = 1 To YnumberB
  For i = 1 To XnumberB
    Xdistance = (i - 1) * GridspaceX / 1000 - Sourcex / 1000
    Ydistance = Abs(((YnumberB - 1) / 2 + 1 - j) * GridspaceY - Sourcey)
    Concentration(i + XnumberB * (j - 1)) = 0
    'Grid(i + XnumberB * (j - 1)) = 10000
    If Xdistance > 0 Then
      sigmaY = Exp(S_ay + S_by * Log(Xdistance) + S_cy * Log(Xdistance) *
Log(Xdistance))
      sigmaZ = Exp(S_az + S_bz * Log(Xdistance) + S_cz * Log(Xdistance) *
Log(Xdistance))
      If (Ydistance / sigmaY) < 10 Then
      Concentration(i + XnumberB * (j - 1)) = Flowrate / PI / Velocity / sigmaZ /
sigmaY / Exp(Ydistance * Ydistance / 2 / sigmaY / sigmaY) / Exp(SourceHeight *
SourceHeight / 2 / sigmaZ / sigmaZ)
      End If
    End If
  Next i
Next j
```

1.2 Code for generating half-sphere data

```
For j = 1 To YnumberB
   For i = 1 To XnumberB
      Xdistance = (i - 1) * GridspaceX - Sourcex
      Ydistance = Abs(((YnumberB - 1) / 2 + 1 - j) * GridspaceY - Sourcey)
      If (Xdistance * Xdistance + Ydistance * Ydistance) >= 1000000 Then
         Concentration(i + XnumberB * (j - 1)) = 0
      Else
         Concentration(i + XnumberB * (j - 1)) = Sqr(1000000 - Xdistance * Xdistance - Ydistance * Ydistance)
      End If
   Next i
Next j
```

1.3 Code for generating taper data

```
For j = 1 To YnumberbB
   For i = 1 To XnumberB
      Xdistance = (i - 1) * GridspaceX - Sourcex
      Ydistance = Abs(((YnumberB - 1) / 2 + 1 - j) * GridspaceY - Sourcey)
      If Sqr(Xdistance * Xdistance + Ydistance * Ydistance) >= 1000 Then
      Concentration(i + XnumberB * (j - 1)) = 1000
      Else
      Concentration(i + XnumberB * (j - 1)) = Sqr(Xdistance * Xdistance + Ydistance *
Ydistance)
      End If
      Concentration(i + XnumberB * (j - 1)) = 1000 - Grid(i + Xnumberb * (j - 1))
   Next i
Next j
```

APPENDIX 2

Computer Code for Algorithms used in OdorImp

2.1 Code for drawing contours

xyl()	A two-dimensional array containing the coordinates of X and Y direction
Grid()	A one-dimensional array containing the Z values of each grid
contourvalue	A variable containing the value of a contour
FactorX	Scale in X direction
FactorY	Scale in Y direction

Public Sub contour2 (xyl, Grid, contourvalue, FactorX, FactorY)

```
Dim dX, dY As Double 'record distance in the drawing area
Dim i, j, k, m As Long
Dim v1, v2, v3, v4 As Integer
Dim element() As Long
Dim Side() As Long
Dim XYC() As Double
Dim maxX, maxY, Xdistance, Ydistance As Double
Dim X1, X2, Y1, Y2, D1, D2 As Double
Dim NumE, Boundary, NumS, NumStart, NumEnd As Long

frmgridline.picdrawingarea.DrawWidth = 1
contourvalue = contourvalue / PA

ReDim element(1 To (Xnumber - 1) * (Ynumber - 1), 1 To 4) As Long
ReDim Side(1 To Xnumber * (Ynumber - 1) + (Xnumber - 1) * Ynumber) As Long
ReDim XYC(1 To (Xnumber * (Ynumber - 1) + (Xnumber - 1) * Ynumber), 1 To 2) As
Double

For i = 1 To Xnumber * (Ynumber - 1) + (Xnumber - 1) * Ynumber 'set initial value
   Side(i) = 0    XYC(i, 1) = 0    XYC(i, 2) = 0
Next i

'Find the points on the horizontal sides
For j = 1 To Ynumber
   For i = 1 To Xnumber - 1
      m = i + (j - 1) * Xnumber
      If Grid(m) = contourvalue Then          Grid(m) = contourvalue + 0.01
      End If
      If Grid(m + 1) = contourvalue Then          Grid(m + 1) = contourvalue + 0.01
      End If

      If (Grid(m) < contourvalue And Grid(m + 1) > contourvalue) Or (Grid(m) >
contourvalue And Grid(m + 1) < contourvalue) Then
         n = i + (Xnumber - 1) * (j - 1)
         Side(n) = 1          D1 = Grid(m)          D2 = Grid(m + 1)
```

```
          X1 = xyl(m, 1)          X2 = xyl(m + 1, 1)
          XYC(n, 1) = X1 + (contourvalue - D1) * (X2 - X1) / (D2 - D1)
          XYC(n, 2) = xyl(m, 2)
          If XYC(n, 1) = xyl(Xnumber * Ynumber, 1) Or XYC(n, 1) = 0 Or XYC(n, 2) = 0
Or XYC(n, 2) = xyl(Xnumber * Ynumber, 2) Then
          Range = 1
          End If
      End If
  Next i
Next j

'Find the points on the vertical sides
For j = 1 To Ynumber - 1
   For i = 1 To Xnumber
      m = i + (j - 1) * Xnumber
      If Grid(m) = contourvalue Then
         Grid(m) = contourvalue + 0.01
      End If
      If Grid(m + Xnumber) = contourvalue Then
         Grid(m + Xnumber) = contourvalue + 0.01
      End If

      If (Grid(m) < contourvalue And Grid(m + Xnumber) > contourvalue) Or (Grid(m) >
contourvalue And Grid(m + Xnumber) < contourvalue) Then
         n = (Xnumber - 1) * Ynumber + Xnumber * (j - 1) + i
         Side(n) = 1          D1 = Grid(m)          D2 = Grid(m + Xnumber)
         Y1 = xyl(m, 2)          Y2 = xyl(m + Xnumber, 2)
         XYC(n, 1) = xyl(m, 1)
         XYC(n, 2) = Y2 + (contourvalue - D2) * (Y2 - Y1) / (D2 - D1)
         If XYC(n, 1) = xyl(Xnumber * Ynumber, 1) Or XYC(n, 1) = 0 Or XYC(n, 2) = 0
Or XYC(n, 2) = xyl(Xnumber * Ynumber, 2) Then
         Range = 1
         End If
      End If
   Next i
Next j
'Form the element using the series number of edges
For i = 1 To (Xnumber - 1) * (Ynumber - 1)
   element(i, 1) = i
   element(i, 2) = i + (Xnumber - 1) * Ynumber + 1 + Fix((i - 1) / (Xnumber - 1))
   element(i, 3) = element(i, 1) + Xnumber - 1
   element(i, 4) = element(i, 2) - 1
Next i

'Draw the contour
R% = Int(255 * Rnd) g% = Int(255 * Rnd) b% = Int(255 * Rnd)
```

```
'Looking for the start point
For k = 1 To (Xnumber - 1) * Ynumber + Xnumber * (Ynumber - 1)
If Side(k) = 1 Then
  For i = 1 To (Xnumber - 1) * (Ynumber - 1)
    For j = 1 To 4
      If element(i, j) = k Then
      NumE = I          NumS = j          Boundary = 0
      If XYC(element(i, j), 2) = xy1(1, 2) Or XYC(element(i, j), 2) = xy1(Xnumber *
Ynumber, 2) Or XYC(element(i, j), 1) = xy1(1, 1) Or XYC(element(i, j), 1) =
xy1(Xnumber * Ynumber, 1) Then
        Boundary = 1
        Call Draw(NumE, NumS, Boundary, element, Side, XYC, R, g, b,
contourvalue, xy1)
        GoTo R
      End If
      For m = 1 To 4
        If m <> j Then
        If Side(element(i, m)) = 1 Then
          If XYC(element(i, m), 2) = xy1(1, 2) Or XYC(element(i, m), 2) =
xy1(Xnumber * Ynumber, 2) Or XYC(element(i, m), 1) = xy1(1, 1) Or XYC(element(i,
m), 1) = xy1(Xnumber * Ynumber, 1) Then
            NumS = m
            Boundary = 1
            Call Draw(NumE, NumS, Boundary, element, Side, XYC, R, g, b,
contourvalue, xy1)
            GoTo R
          End If
        End If
        End If
      Next m
      Call Draw(NumE, NumS, Boundary, element, Side, XYC, R, g, b, contourvalue,
xy1)
      GoTo R
      End If
    Next j
  Next i
End If
R: Next k

'Close the contour
For i = 1 To (Xnumber - 1) * (Ynumber - 1)
  For j = 1 To 3
    If Side(element(i, j)) = 1 Then
    For k = j + 1 To 4
      If Side(element(i, k)) = 1 Then
```

```
            frmgridline.picdrawingarea.Line (XYC(element(i, j), 1), XYC(element(i, j), 2))-(XYC(element(i, k), 1), XYC(element(i, k), 2)), RGB(R, g, b)
            Side(element(i, j)) = 0
            Side(element(i, k)) = 0
          End If
        Next k
        End If
    Next j
Next i

For i = 1 To (Xnumber - 1) * Ynumber + (Ynumber - 1) * Xnumber
If Side(i) = 1 Then Debug.Print "i", i
Next i

maxX = 0
For i = 1 To (Xnumber - 1) * Ynumber + (Ynumber - 1) * Xnumber
    If XYC(i, 1) > maxX Then
    maxX = XYC(i, 1)
    maxY = XYC(i, 2)
    End If
Next i

frmgridline.picdrawingarea.CurrentX = maxX - 300
frmgridline.picdrawingarea.CurrentY = maxY
frmgridline.picdrawingarea.Print contourvalue

End Sub
```

Public Sub Draw(NumE, NumS, Boundary, element, Side, XYC, R, g, b, contourvalue, xy1)

```
Dim StartE, StartS As Long

frmgridline.picdrawingarea.CurrentX = XYC(element(NumE, NumS), 1)
frmgridline.picdrawingarea.CurrentY = XYC(element(NumE, NumS), 2)
StartE = NumE
StartS = NumS
Draw:  If Boundary = 0 Then
        If NumS = 1 Then
            If NumE - Xnumber + 1 > 0 Then
               If Side(element(NumE - Xnumber + 1, 4)) = 1 Then
                  If XYC(element(NumE - Xnumber + 1, 4), 1) - xy1(1, 1) <> 0 Then
                     frmgridline.picdrawingarea.Line -(XYC(element(NumE - Xnumber + 1,
4), 1), XYC(element(NumE - Xnumber + 1, 4), 2)), RGB(R, g, b)
                     NumE = NumE - Xnumber + 1
                     NumS = 4
                     Side(element(NumE, 4)) = 0
                     If (StartE - NumE = 0 And StartS - NumS = 0) Then GoTo W Else
GoTo Draw
                  Else
                     frmgridline.picdrawingarea.Line -(XYC(element(NumE - Xnumber + 1,
4), 1), XYC(element(NumE - Xnumber + 1, 4), 2)), RGB(R, g, b)
                     Side(element(NumE - Xnumber + 1, 4)) = 0
                     GoTo W
                  End If
               End If
               If Side(element(NumE - Xnumber + 1, 1)) = 1 Then
                  If XYC(element(NumE - Xnumber + 1, 1), 2) - xy1(1, 2) <> 0 Then
                     frmgridline.picdrawingarea.Line -(XYC(element(NumE - Xnumber + 1,
1), 1), XYC(element(NumE - Xnumber + 1, 1), 2)), RGB(R, g, b)
                     NumE = NumE - Xnumber + 1
                     NumS = 1
                     Side(element(NumE, 1)) = 0
                  If (StartE - NumE = 0 And StartS - NumS = 0) Then GoTo W Else GoTo
Draw
                  Else
                     frmgridline.picdrawingarea.Line -(XYC(element(NumE - Xnumber + 1,
1), 1), XYC(element(NumE - Xnumber + 1, 1), 2)), RGB(R, g, b)
                     Side(element(NumE - Xnumber + 1, 1)) = 0
                     GoTo W
                  End If
               End If
               If Side(element(NumE - Xnumber + 1, 2)) = 1 Then
```

```
            If XYC(element(NumE - Xnumber + 1, 2), 2) - xy1(Xnumber * Ynumber,
2) <> 0 Then
              frmgridline.picdrawingarea.Line -(XYC(element(NumE - Xnumber + 1,
2), 1), XYC(element(NumE - Xnumber + 1, 2), 2)), RGB(R, g, b)
              NumE = NumE - Xnumber + 1
              NumS = 2
              Side(element(NumE, 2)) = 0
              If (StartE - NumE = 0 And StartS - NumS = 0) Then GoTo W Else
GoTo Draw
            Else
              frmgridline.picdrawingarea.Line -(XYC(element(NumE - Xnumber + 1,
2), 1), XYC(element(NumE - Xnumber + 1, 2), 2)), RGB(R, g, b)
              Side(element(NumE - Xnumber + 1, 2)) = 0
              GoTo W
            End If
          End If
        End If
      End If
      If NumS = 2 Then
        For i = 1 To 3
          If (NumE + 1) <= (Xnumber - 1) * (Ynumber - 1) Then
          If Side(element(NumE + 1, i)) = 1 Then
            If XYC(element(NumE + 1, i), 1) - xy1(1, 1) <> 0 And
XYC(element(NumE + 1, i), 2) - xy1(1, 2) <> 0 And XYC(element(NumE + 1, i), 1) -
xy1(Xnumber * Ynumber, 1) <> 0 And XYC(element(NumE + 1, i), 2) - xy1(Xnumber *
Ynumber, 2) <> 0 Then
              frmgridline.picdrawingarea.Line -(XYC(element(NumE + 1, i), 1),
XYC(element(NumE + 1, i), 2)), RGB(R, g, b)
              NumE = NumE + 1
              NumS = i
              Side(element(NumE, i)) = 0
              If (StartE - NumE = 0 And StartS - NumS = 0) Then GoTo W Else
GoTo Draw
            Else
              frmgridline.picdrawingarea.Line -(XYC(element(NumE + 1, i), 1),
XYC(element(NumE + 1, i), 2)), RGB(R, g, b)
              Side(element(NumE + 1, i)) = 0
              GoTo W
            End If
          End If
          End If
        Next i
      End If
      If NumS = 3 Then
      For i = 2 To 4
        If (NumE + Xnumber - 1) <= (Xnumber - 1) * (Ynumber - 1) Then
```

```
            If Side(element(NumE + Xnumber - 1, i)) = 1 Then
            If XYC(element(NumE + Xnumber - 1, i), 1) <> xy1(1, 1) And
XYC(element(NumE + Xnumber - 1, i), 2) <> xy1(1, 2) And XYC(element(NumE +
Xnumber - 1, i), 1) <> xy1(Xnumber * Ynumber, 1) And XYC(element(NumE +
Xnumber - 1, i), 2) <> xy1(Xnumber * Ynumber, 2) Then
              frmgridline.picdrawingarea.Line -(XYC(element(NumE + Xnumber - 1,
i), 1), XYC(element(NumE + Xnumber - 1, i), 2)), RGB(R, g, b)
              NumE = NumE + Xnumber - 1
              NumS = i
              Side(element(NumE, i)) = 0
              If (StartE - NumE = 0 And StartS - NumS = 0) Then GoTo W Else GoTo
Draw
            Else
              frmgridline.picdrawingarea.Line -(XYC(element(NumE + Xnumber - 1,
i), 1), XYC(element(NumE + Xnumber - 1, i), 2)), RGB(R, g, b)
              Side(element(NumE + Xnumber - 1, i)) = 0
              GoTo W
            End If
            End If
          End If
          Next i
        End If
        If NumS = 4 Then
        If NumE - 1 > 0 Then
          If Side(element(NumE - 1, 3)) = 1 Then
            If XYC(element(NumE - 1, 3), 1) - xy1(Xnumber * Ynumber, 1) <> 0 Then
              frmgridline.picdrawingarea.Line -(XYC(element(NumE - 1, 3), 1),
XYC(element(NumE - 1, 3), 2)), RGB(R, g, b)
              NumE = NumE - 1
              NumS = 3
              Side(element(NumE, 3)) = 0
             If (StartE - NumE = 0 And StartS - NumS = 0) Then GoTo W Else GoTo
Draw
            Else
              frmgridline.picdrawingarea.Line -(XYC(element(NumE - 1, 3), 1),
XYC(element(NumE - 1, 3), 2)), RGB(R, g, b)
              Side(element(NumE - 1, 3)) = 0
              GoTo W
            End If
          End If
          If Side(element(NumE - 1, 4)) = 1 Then
            If XYC(element(NumE - 1, 4), 1) - xy1(1, 1) <> 0 Then
              frmgridline.picdrawingarea.Line -(XYC(element(NumE - 1, 4), 1),
XYC(element(NumE - 1, 4), 2)), RGB(R, g, b)
              NumE = NumE - 1
              NumS = 4
```

```
                Side(element(NumE, 4)) = 0
                If (StartE - NumE = 0 And StartS - NumS = 0) Then GoTo W Else GoTo
Draw
              Else
                frmgridline.picdrawingarea.Line -(XYC(element(NumE - 1, 4), 1),
XYC(element(NumE - 1, 4), 2)), RGB(R, g, b)
                Side(element(NumE - 1, 4)) = 0
                GoTo W
              End If
            End If
            If Side(element(NumE - 1, 1)) = 1 Then
              If XYC(element(NumE - 1, 1), 2) - xy1(1, 2) <> 0 Then
                frmgridline.picdrawingarea.Line -(XYC(element(NumE - 1, 1), 1),
XYC(element(NumE - 1, 1), 2)), RGB(R, g, b)
                NumE = NumE - 1
                NumS = 1
                Side(element(NumE, 1)) = 0
                If (StartE - NumE = 0 And StartS - NumS = 0) Then GoTo W Else GoTo
Draw
              Else
                frmgridline.picdrawingarea.Line -(XYC(element(NumE - 1, 1), 1),
XYC(element(NumE - 1, 1), 2)), RGB(R, g, b)
                Side(element(NumE - 1, 1)) = 0
                GoTo W
              End If
            End If
          End If
          End If
        End If
      If Boundary = 1 Then
        Boundary = 0
        If NumS = 1 Then
          For i = 2 To 4
            If Side(element(NumE, i)) = 1 Then
              frmgridline.picdrawingarea.Line -(XYC(element(NumE, i), 1),
XYC(element(NumE, i), 2)), RGB(R, g, b)
              Side(element(NumE, 1)) = 0
              Side(element(NumE, i)) = 0
              NumS = i
              GoTo Draw
            End If
          Next i
        End If
        If NumS = 2 Then
          If Side(element(NumE, 3)) = 1 Then
```

```
frmgridline.picdrawingarea.Line -(XYC(element(NumE, 3), 1), XYC(element(NumE, 3), 2)), RGB(R, g, b)
            Side(element(NumE, 2)) = 0
            Side(element(NumE, 3)) = 0
            NumS = 3
            GoTo Draw
        End If
        If Side(element(NumE, 4)) = 1 Then
            frmgridline.picdrawingarea.Line -(XYC(element(NumE, 4), 1), XYC(element(NumE, 4), 2)), RGB(R, g, b)
            Side(element(NumE, 2)) = 0
            Side(element(NumE, 4)) = 0
            NumS = 4
            GoTo Draw
        End If
        If Side(element(NumE, 1)) = 1 Then
            frmgridline.picdrawingarea.Line -(XYC(element(NumE, 1), 1), XYC(element(NumE, 1), 2)), RGB(R, g, b)
            Side(element(NumE, 2)) = 0
            Side(element(NumE, 1)) = 0
            NumS = 1
            GoTo Draw
        End If
    End If
    If NumS = 3 Then
        If Side(element(NumE, 4)) = 1 Then
            frmgridline.picdrawingarea.Line -(XYC(element(NumE, 4), 1), XYC(element(NumE, 4), 2)), RGB(R, g, b)
            Side(element(NumE, 3)) = 0
            Side(element(NumE, 4)) = 0
            NumS = 4
            GoTo Draw
        End If
        If Side(element(NumE, 1)) = 1 Then
            frmgridline.picdrawingarea.Line -(XYC(element(NumE, 1), 1), XYC(element(NumE, 1), 2)), RGB(R, g, b)
            Side(element(NumE, 3)) = 0
            Side(element(NumE, 1)) = 0
            NumS = 1
            GoTo Draw
        End If
        If Side(element(NumE, 2)) = 1 Then
            frmgridline.picdrawingarea.Line -(XYC(element(NumE, 2), 1), XYC(element(NumE, 2), 2)), RGB(R, g, b)
            Side(element(NumE, 3)) = 0
            Side(element(NumE, 2)) = 0
```

```
        NumS = 2
        GoTo Draw
      End If
    End If
    If NumS = 4 Then
      For i = 1 To 3
        If Side(element(NumE, i)) = 1 Then
          frmgridline.picdrawingarea.Line        -(XYC(element(NumE,        i),        1),
XYC(element(NumE, i), 2)), RGB(R, g, b)
          Side(element(NumE, 4)) = 0
          Side(element(NumE, i)) = 0
          NumS = i
          GoTo Draw
        End If
      Next i
    End If
  End If

W:  frmgridline.picdrawingarea.Print contourvalue

End Sub
```

2.2 Code for calculating footprint areas *F(A)*, *F(C)*, or *F(P)* and annoyance-, concentration-, or probability-weighted footprint areas, *Fw(A)*, *Fw(C)*, and *Fw(P)*

Grid()	A one-dimensional array containing the Z values of each grid
contourvalue	A specific value of contours
Parameter	Set as "pwfa" when calculating annoyance-, concentration-, or probability-weighted footprint area, $F_W(A)$, $F_W(C)$, $F_W(P)$; Set as "fa" when calculating footprint area F(A), F(C), or F(P)

Public Function PWFA(Grid, contourvalue, Parameter)

```
Dim i As Long
Dim v1, v2, v3, v4 As Long
Dim X1, Y2, X3, Y4, EVolume As Double
Dim aX, aY, aZ, bX, bY, bZ, cX, cY, cZ, dX, dY, dZ, CV As Double

PWFA = 0

If Model = "Beychok" Then
For i = 1 To (XnumberB - 1) * (YnumberB - 1) 'scan every element
   'calculate the vertix number of each element
   v1 = i + Fix((i - 1) / (XnumberB - 1))
   v2 = v1 + 1    v4 = v1 + XnumberB    v3 = v4 + 1

   aX = ((v1 - 1) Mod XnumberB) * GridspaceX
   aY = ((YnumberB - 1) / 2 - Fix(v1 / XnumberB)) * GridspaceY
   aZ = Grid(v1)

   bX = aX + GridspaceX    bY = aY    bZ = Grid(v2)
   cX = bX    cY = bY - GridspaceY    cZ = Grid(v3)
   dX = aX    dY = cY    dZ = Grid(v4)

'Interpolation
If (aZ > contourvalue And bZ < contourvalue) Or (aZ < contourvalue And bZ >
contourvalue) Then    X1 = aX + (contourvalue - aZ) / (bZ - aZ) * GridspaceX
End If
If (bZ > contourvalue And cZ < contourvalue) Or (bZ < contourvalue And cZ >
contourvalue) Then    Y2 = bY - (contourvalue - bZ) / (cZ - bZ) * GridspaceY
End If
If (cZ > contourvalue And dZ < contourvalue) Or (cZ < contourvalue And dZ >
contourvalue) Then    X3 = dX + (contourvalue - dZ) / (cZ - dZ) * GridspaceX
End If
If (dZ > contourvalue And aZ < contourvalue) Or (dZ < contourvalue And aZ >
contourvalue) Then    Y4 = aY - (contourvalue - aZ) / (dZ - aZ) * GridspaceY
End If
```

```
'set casevalue
If aZ < contourvalue And bZ < contourvalue And cZ < contourvalue And dZ <
contourvalue Then    Casevalue = 1
End If
If aZ < contourvalue And bZ > contourvalue And cZ > contourvalue And dZ >
contourvalue Then    Casevalue = 21
End If
If aZ > contourvalue And bZ < contourvalue And cZ > contourvalue And dZ >
contourvalue Then    Casevalue = 22
End If
If aZ > contourvalue And bZ > contourvalue And cZ < contourvalue And dZ >
contourvalue Then    Casevalue = 23
End If
If aZ > contourvalue And bZ > contourvalue And cZ > contourvalue And dZ <
contourvalue Then    Casevalue = 24
End If
If aZ < contourvalue And bZ < contourvalue And cZ > contourvalue And dZ >
contourvalue Then    Casevalue = 31
End If
If aZ > contourvalue And bZ < contourvalue And cZ < contourvalue And dZ >
contourvalue Then    Casevalue = 32
End If
If aZ > contourvalue And bZ > contourvalue And cZ < contourvalue And dZ <
contourvalue Then    Casevalue = 33
End If
If aZ < contourvalue And bZ > contourvalue And cZ > contourvalue And dZ <
contourvalue Then    Casevalue = 34
End If
If aZ > contourvalue And bZ < contourvalue And cZ < contourvalue And dZ <
contourvalue Then    Casevalue = 41
End If
If aZ < contourvalue And bZ > contourvalue And cZ < contourvalue And dZ <
contourvalue Then    Casevalue = 42
End If
If aZ < contourvalue And bZ < contourvalue And cZ > contourvalue And dZ <
contourvalue Then    Casevalue = 43
End If
If aZ < contourvalue And bZ < contourvalue And cZ < contourvalue And dZ >
contourvalue Then    Casevalue = 44
End If
If aZ < contourvalue And bZ > contourvalue And cZ < contourvalue And dZ >
contourvalue Then    Average = (aZ + bZ + cZ + dZ) / 4
   If Average > contourvalue Then
      Casevalue = 511
   Else
      Casevalue = 512
```

```
  End If
End If
If aZ > contourvalue And bZ < contourvalue And cZ > contourvalue And dZ <
contourvalue Then   Average = (aZ + bZ + cZ + dZ) / 4
  If Average > contourvalue Then      Casevalue = 521
  Else      Casevalue = 522
  End If
End If
If aZ > contourvalue And bZ > contourvalue And cZ > contourvalue And dZ >
contourvalue Then   Casevalue = 6
End If

If Parameter = "fa" Then
aZ = 1 bZ = 1 cZ = 1 dZ = 1 CV = 1
Else
CV = contourvalue
End If

Select Case Casevalue
Case 1
  EVolume = 0
Case 21
  EVolume = Volume(aX, aY, aZ, bX, bY, bZ, cX, cY, cZ) + Volume(cX, cY, cZ, dX,
dY, dZ, aX, aY, aZ) - Volume(aX, Y4, CV, aX, aY, aZ, X1, aY, CV)
Case 22
  EVolume = Volume(aX, aY, aZ, bX, bY, bZ, cX, cY, cZ) + Volume(cX, cY, cZ, dX,
dY, dZ, aX, aY, aZ) - Volume(X1, bY, CV, bX, bY, bZ, bX, Y2, CV)
Case 23
  EVolume = Volume(aX, aY, aZ, bX, bY, bZ, cX, cY, cZ) + Volume(cX, cY, cZ, dX,
dY, dZ, aX, aY, aZ) - Volume(cX, Y2, CV, cX, cY, cZ, X3, cY, CV)
Case 24
  EVolume = Volume(aX, aY, aZ, bX, bY, bZ, cX, cY, cZ) + Volume(cX, cY, cZ, dX,
dY, dZ, aX, aY, aZ) - Volume(X3, dY, CV, dX, dY, dZ, dX, Y4, CV)
Case 31
  EVolume = Volume(cX, Y2, CV, cX, cY, cZ, dX, dY, dZ) + Volume(dX, dY, dZ, dX,
Y4, CV, cX, Y2, CV)
Case 32
  EVolume = Volume(aX, aY, aZ, X1, aY, CV, X3, dY, CV) + Volume(X3, dY, CV,
dX, dY, dZ, aX, aY, aZ)
Case 33
  EVolume = Volume(aX, aY, aZ, bX, bY, bZ, bX, Y2, CV) + Volume(bX, Y2, CV, aX,
Y4, CV, aX, aY, aZ)
Case 34
  EVolume = Volume(X1, bY, CV, bX, bY, bZ, cX, cY, cZ) + Volume(cX, cY, cZ, X3,
cY, CV, X1, bY, CV)
Case 41
```

```
   EVolume = Volume(aX, Y4, CV, aX, aY, aZ, X1, aY, CV)
Case 42
   EVolume = Volume(X1, bY, CV, bX, bY, bZ, bX, Y2, CV)
Case 43
    EVolume = Volume(cX, Y2, CV, cX, cY, cZ, X3, cY, CV)
Case 44
   EVolume = Volume(X3, dY, CV, dX, dY, dZ, dX, Y4, CV)
Case 511
   EVolume = Volume(aX, aY, aZ, bX, bY, bZ, cX, cY, cZ) + Volume(cX, cY, cZ, dX,
dY, dZ, aX, aY, aZ) - Volume(aX, Y4, CV, aX, aY, aZ, X1, aY, CV) - Volume(cX, Y2,
CV, cX, cY, cZ, X3, cY, CV)
Case 512
   EVolume = Volume(X1, bY, CV, bX, bY, bZ, bX, Y2, CV) + Volume(X3, dY, CV,
dX, dY, dZ, dX, Y4, CV)
Case 521
   EVolume = Volume(aX, aY, aZ, bX, bY, bZ, cX, cY, cZ) + Volume(cX, cY, cZ, dX,
dY, dZ, aX, aY, aZ) - Volume(X1, bY, CV, bX, bY, bZ, bX, Y2, CV) - Volume(X3, dY,
CV, dX, dY, dZ, dX, Y4, CV)
Case 522
   EVolume = Volume(aX, Y4, CV, aX, aY, aZ, X1, aY, CV) + Volume(cX, Y2, CV,
cX, cY, cZ, X3, cY, CV)
Case 6
   EVolume = Volume(aX, aY, aZ, bX, bY, bZ, cX, cY, cZ) + Volume(cX, cY, cZ, dX,
dY, dZ, aX, aY, aZ)
End Select

PWFA = PWFA + EVolume

Next i

End If

If Model = "ReadFile" Then

For i = 1 To (Xnumber - 1) * (Ynumber - 1) 'scan every element
   'calculate the vertix number of each element
   v1 = i + Fix((i - 1) / (Xnumber - 1)) v2 = v1 + 1  v4 = v1 + Xnumber v3 = v4 + 1

   aX = XY(v1, 1)    aY = XY(v1, 2)    aZ = Grid(v1)
   bX = XY(v2, 1)    bY = XY(v2, 2)    bZ = Grid(v2)
   cX = XY(v3, 1)    cY = XY(v3, 2)    cZ = Grid(v3)
   dX = XY(v4, 1)    dY = XY(v4, 2)    dZ = Grid(v4)
   GridspaceX = Abs(aX - bX)    GridspaceY = Abs(cY - bY)

'Interpolation
```

```
If (aZ > contourvalue And bZ < contourvalue) Or (aZ < contourvalue And bZ >
contourvalue) Then    X1 = aX + (contourvalue - aZ) / (bZ - aZ) * GridspaceX
End If
If (bZ > contourvalue And cZ < contourvalue) Or (bZ < contourvalue And cZ >
contourvalue) Then    Y2 = bY + (contourvalue - bZ) / (cZ - bZ) * GridspaceY
End If
If (cZ > contourvalue And dZ < contourvalue) Or (cZ < contourvalue And dZ >
contourvalue) Then    X3 = dX + (contourvalue - dZ) / (cZ - dZ) * GridspaceX
End If
If (dZ > contourvalue And aZ < contourvalue) Or (dZ < contourvalue And aZ >
contourvalue) Then    Y4 = aY + (contourvalue - aZ) / (dZ - aZ) * GridspaceY
End If

'set casevalue
If aZ < contourvalue And bZ < contourvalue And cZ < contourvalue And dZ <
contourvalue Then    Casevalue = 1
End If
If aZ < contourvalue And bZ > contourvalue And cZ > contourvalue And dZ >
contourvalue Then    Casevalue = 21
End If
If aZ > contourvalue And bZ < contourvalue And cZ > contourvalue And dZ >
contourvalue Then    Casevalue = 22
End If
If aZ > contourvalue And bZ > contourvalue And cZ < contourvalue And dZ >
contourvalue Then    Casevalue = 23
End If
If aZ > contourvalue And bZ > contourvalue And cZ > contourvalue And dZ <
contourvalue Then    Casevalue = 24
End If
If aZ < contourvalue And bZ < contourvalue And cZ > contourvalue And dZ >
contourvalue Then    Casevalue = 31
End If
If aZ > contourvalue And bZ < contourvalue And cZ < contourvalue And dZ >
contourvalue Then    Casevalue = 32
End If
If aZ > contourvalue And bZ > contourvalue And cZ < contourvalue And dZ <
contourvalue Then    Casevalue = 33
End If
If aZ < contourvalue And bZ > contourvalue And cZ > contourvalue And dZ <
contourvalue Then    Casevalue = 34
End If
If aZ > contourvalue And bZ < contourvalue And cZ < contourvalue And dZ <
contourvalue Then    Casevalue = 41
End If
If aZ < contourvalue And bZ > contourvalue And cZ < contourvalue And dZ <
contourvalue Then    Casevalue = 42
```

```
End If
If aZ < contourvalue And bZ < contourvalue And cZ > contourvalue And dZ <
contourvalue Then    Casevalue = 43
End If
If aZ < contourvalue And bZ < contourvalue And cZ < contourvalue And dZ >
contourvalue Then    Casevalue = 44
End If
If aZ < contourvalue And bZ > contourvalue And cZ < contourvalue And dZ >
contourvalue Then    Average = (aZ + bZ + cZ + dZ) / 4
   If Average > contourvalue Then       Casevalue = 511
   Else       Casevalue = 512
   End If
End If
If aZ > contourvalue And bZ < contourvalue And cZ > contourvalue And dZ <
contourvalue Then    Average = (aZ + bZ + cZ + dZ) / 4
   If Average > contourvalue Then       Casevalue = 521
   Else       Casevalue = 522
   End If
End If
If aZ > contourvalue And bZ > contourvalue And cZ > contourvalue And dZ >
contourvalue Then    Casevalue = 6
End If

If Parameter = "fa" Then
aZ = 1 bZ = 1 cZ = 1 dZ = 1 CV = 1
Else
CV = contourvalue
End If

Select Case Casevalue
Case 1
   EVolume = 0
Case 21
   EVolume = Volume(aX, aY, aZ, bX, bY, bZ, cX, cY, cZ) + Volume(cX, cY, cZ, dX,
dY, dZ, aX, aY, aZ) - Volume(aX, Y4, CV, aX, aY, aZ, X1, aY, CV)
Case 22
   EVolume = Volume(aX, aY, aZ, bX, bY, bZ, cX, cY, cZ) + Volume(cX, cY, cZ, dX,
dY, dZ, aX, aY, aZ) - Volume(X1, bY, CV, bX, bY, bZ, bX, Y2, CV)
Case 23
   EVolume = Volume(aX, aY, aZ, bX, bY, bZ, cX, cY, cZ) + Volume(cX, cY, cZ, dX,
dY, dZ, aX, aY, aZ) - Volume(cX, Y2, CV, cX, cY, cZ, X3, cY, CV)
Case 24
   EVolume = Volume(aX, aY, aZ, bX, bY, bZ, cX, cY, cZ) + Volume(cX, cY, cZ, dX,
dY, dZ, aX, aY, aZ) - Volume(X3, dY, CV, dX, dY, dZ, dX, Y4, CV)
Case 31
```

```
  EVolume = Volume(cX, Y2, CV, cX, cY, cZ, dX, dY, dZ) + Volume(dX, dY, dZ, dX,
Y4, CV, cX, Y2, CV)
Case 32
  EVolume = Volume(aX, aY, aZ, X1, aY, CV, X3, dY, CV) + Volume(X3, dY, CV,
dX, dY, dZ, aX, aY, aZ)
Case 33
  EVolume = Volume(aX, aY, aZ, bX, bY, bZ, bX, Y2, CV) + Volume(bX, Y2, CV, aX,
Y4, CV, aX, aY, aZ)
Case 34
  EVolume = Volume(X1, bY, CV, bX, bY, bZ, cX, cY, cZ) + Volume(cX, cY, cZ, X3,
cY, CV, X1, bY, CV)
Case 41
  EVolume = Volume(aX, Y4, CV, aX, aY, aZ, X1, aY, CV)
Case 42
  EVolume = Volume(X1, bY, CV, bX, bY, bZ, bX, Y2, CV)
Case 43
  EVolume = Volume(cX, Y2, CV, cX, cY, cZ, X3, cY, CV)
Case 44
  EVolume = Volume(X3, dY, CV, dX, dY, dZ, dX, Y4, CV)
Case 511
  EVolume = Volume(aX, aY, aZ, bX, bY, bZ, cX, cY, cZ) + Volume(cX, cY, cZ, dX,
dY, dZ, aX, aY, aZ) - Volume(aX, Y4, CV, aX, aY, aZ, X1, aY, CV) - Volume(cX, Y2,
CV, cX, cY, cZ, X3, cY, CV)
Case 512
  EVolume = Volume(X1, bY, CV, bX, bY, bZ, bX, Y2, CV) + Volume(X3, dY, CV,
dX, dY, dZ, dX, Y4, CV)
Case 521
  EVolume = Volume(aX, aY, aZ, bX, bY, bZ, cX, cY, cZ) + Volume(cX, cY, cZ, dX,
dY, dZ, aX, aY, aZ) - Volume(X1, bY, CV, bX, bY, bZ, bX, Y2, CV) - Volume(X3, dY,
CV, dX, dY, dZ, dX, Y4, CV)
Case 522
  EVolume = Volume(aX, Y4, CV, aX, aY, aZ, X1, aY, CV) + Volume(cX, Y2, CV,
cX, cY, cZ, X3, cY, CV)
Case 6
  EVolume = Volume(aX, aY, aZ, bX, bY, bZ, cX, cY, cZ) + Volume(cX, cY, cZ, dX,
dY, dZ, aX, aY, aZ)
End Select

PWFA = PWFA + EVolume

Next i

End If
End Function
```

Public Function Volume (aX, aY, aZ, bX, bY, bZ, cX, cY, cZ)

```
Dim Volume1, Area1, Volume2, Area2, ab, bc, ac As Double
Dim A_a, A_b, A_c, Height As Double

ab = Sqr((aX - bX) ^ 2 + (aY - bY) ^ 2)
bc = Sqr((bX - cX) ^ 2 + (bY - cY) ^ 2)
ac = Sqr((aX - cX) ^ 2 + (aY - cY) ^ 2)
A_a = Arccos((ab ^ 2 + ac ^ 2 - bc ^ 2) / (2 * ab * ac))
A_b = Arccos((ab ^ 2 + bc ^ 2 - ac ^ 2) / (2 * ab * bc))
A_c = Arccos((bc ^ 2 + ac ^ 2 - ab ^ 2) / (2 * bc * ac))

Area1 = 0.5 * ab * ac * Sin(A_a)
If aZ <= bZ And aZ <= cZ Then
   Volume1 = Area1 * aZ
   Height = ac * Sin(A_c)
   Area2 = 0.5 * (cZ + bZ) * bc - bc * aZ
End If
If bZ <= aZ And bZ <= cZ Then
   Volume1 = Area1 * bZ
   Height = ab * Sin(A_a)
   Area2 = 0.5 * (aZ + cZ) * ac - (ac * bZ)
End If
If cZ <= bZ And cZ <= aZ Then
   Volume1 = Area1 * cZ
   Height = ac * Sin(A_a)
   Area2 = 0.5 * (aZ + bZ) * ab - ab * cZ
End If
Volume2 = Area2 * Height / 3
Volume = Volume1 + Volume2
End Function
```

Public Function Arccos(X)

```
        Arccos = Atn(-X / Sqr(-X * X + 1)) + 2 * Atn(1)
End Function
```

2.3 Code for calculating the total annoyance-, concentration- and probability-weighted footprint areas: $F_{WA}(T)$, $F_{WC}(T)$, $F_{WP}(T)$

Grid() A one-dimensional array containing the Z values of each grid

```
Dim i, j As Long
Dim v1, v2, v3, v4 As Long
Dim aX, aY, aZ, bX, bY, bZ, cX, cY, cZ, dX, dY, dZ As Double
TotalVolume = 0
If Model = "Beychok" Then
For i = 1 To (XnumberB - 1) * (YnumberB - 1)  'scan every element

   'calculate the vertix number of each element
   v1 = i + Fix((i - 1) / (XnumberB - 1)) v2 = v1 + 1 v4 = v1 + XnumberB v3 = v4 + 1

   aX = ((v1 - 1) Mod XnumberB) * GridspaceX
   aY = ((YnumberB - 1) / 2 - Fix(v1 / XnumberB)) * GridspaceY
   aZ = Grid(v1)
   bX = aX + GridspaceX    bY = aY    bZ = Grid(v2)
   cX = bX    cY = bY - GridspaceY    cZ = Grid(v3)
   dX = aX    dY = cY    dZ = Grid(v4)

   TotalVolume = TotalVolume + Volume(aX, aY, aZ, bX, bY, bZ, cX, cY, cZ)
   TotalVolume = TotalVolume + Volume(cX, cY, cZ, dX, dY, dZ, aX, aY, aZ)

Next i
End If

If Model = "ReadFile" Then
For i = 1 To (Xnumber - 1) * (Ynumber - 1)  'scan every element

   'calculate the vertix number of each element
   v1 = i + Fix((i - 1) / (Xnumber - 1))   v2 = v1 + 1  v4 = v1 + Xnumber v3 = v4 + 1
   aX = XY(v1, 1)   aY = XY(v1, 2)   aZ = Grid(v1)
   bX = XY(v2, 1)   bY = XY(v2, 2)   bZ = Grid(v2)
   cX = XY(v3, 1)   cY = XY(v3, 2)   cZ = Grid(v3)
   dX = XY(v4, 1)   dY = XY(v4, 2)   dZ = Grid(v4)

   TotalVolume = TotalVolume + Volume(aX, aY, aZ, bX, bY, bZ, cX, cY, cZ)
   TotalVolume = TotalVolume + Volume(cX, cY, cZ, dX, dY, dZ, aX, aY, aZ)

Next i
End If

End Function
```

2.4 An alternative algorithm for calculating footprint areas: *F(A)*, *F(C)*, or *F(P)*

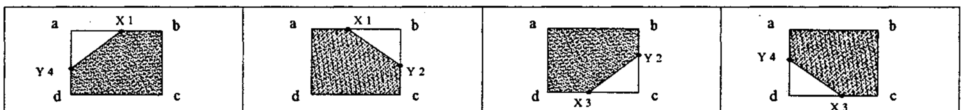
1. Algorithm

Table A2-1: Algorithm for calculating *F(C)*, *F(P)*, and *F(A)*

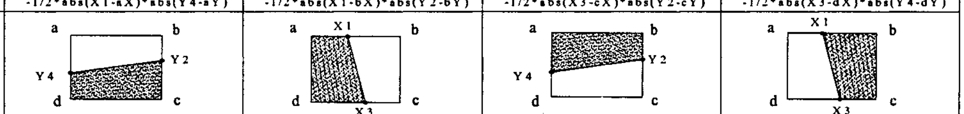
Set FA=0
Do a loop for all elements: 1. Find the series numbers of four vertices of each element. 2. Get X, Y coordinates and Z values of vertices. 3. Calculate intersected points of four edges respectively. X1, Y2, X3, Y4 are the coordinates of intersected points for North, East, South, West edge respectively. 4. Get case value for each element and calculate the area of each element according to table shown below. 5. Accumulate FA.

2. Case values, conditions and footprint area equations for all situation

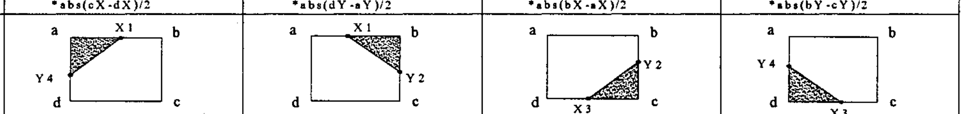
Table A2-2: Case values, conditions and footprint area equations for all situations



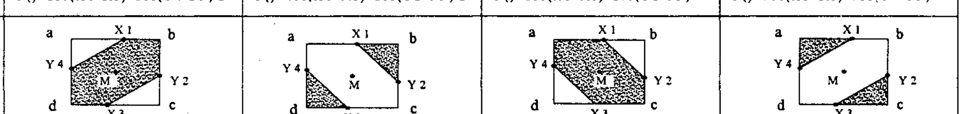
Case Value	21	22	23	24
Conditions	aZ<CV,bZ>CV,cZ>CV,dZ>CV	aZ>CV,bZ<CV,cZ>CV,dZ>CV	aZ>CV,bZ>CV,cZ<CV,dZ>CV	aZ>CV,bZ>CV,cZ>CV,dZ<CV
Equations	F()=abs(bX-aX)*abs(cY-bY) -1/2*abs(X1-aX)*abs(Y4-aY)	F()=abs(bX-aX)*abs(cY-bY) -1/2*abs(X1-bX)*abs(Y2-bY)	F()=abs(bX-aX)*abs(cY-bY) -1/2*abs(X3-cX)*abs(Y2-cY)	F()=abs(bX-aX)*abs(cY-bY) -1/2*abs(X3-dX)*abs(Y4-dY)



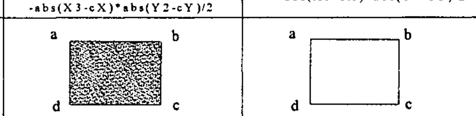
Case Value	31	32	33	34
Conditions	aZ<CV,bZ<CV,cZ>CV,dZ>CV	aZ>CV,bZ<CV,cZ<CV,dZ>CV	aZ>CV,bZ>CV,cZ<CV,dZ<CV	aZ<CV,bZ>CV,cZ>CV,dZ<CV
Equations	F()=(abs(Y4-dY)+abs(Y2-cY)) *abs(cX-dX)/2	F()=(abs(X3-dX)+abs(X1-aX)) *abs(dY-aY)/2	F()=(abs(Y2-bY)+abs(Y4-aY)) *abs(bX-aX)/2	F()=(abs(X1-bX)+abs(X3-cX)) *abs(bY-cY)/2



Case Value	41	42	43	44
Conditions	aZ>CV,bZ<CV,cZ<CV,dZ<CV	aZ<CV,bZ>CV,cZ<CV,dZ<CV	aZ<CV,bZ<CV,cZ>CV,dZ<CV	aZ<CV,bZ<CV,cZ<CV,dZ>CV
Equations	F()=abs(X1-aX)*abs(Y4-aY)/2	F()=abs(X1-bX)*abs(Y2-bY)/2	F()=abs(X3-cX)*abs(Y2-cY)	F()=abs(X3-dX)*abs(Y4-dY)



Case Value	511	512	521	522
Conditions	aZ<CV,bZ>CV,cZ<CV,dZ>CV M>CV	aZ<CV,bZ>CV,cZ<CV,dZ>CV M<CV	aZ>CV,bZ<CV,cZ>CV,dZ<CV M>CV	aZ>CV,bZ<CV,cZ>CV,dZ<CV M<CV
Equations	F()=abs(bX-aX)*abs(cY-bY) -abs(X1-aX)*abs(Y4-aY)/2 -abs(X3-cX)*abs(Y2-cY)/2	F()=abs(X1-bX)*abs(Y2-bY)/2 +abs(X3-dX)*abs(Y4-dY)/2	F()=abs(bX-aX)*abs(cY-bY) -abs(X3-dX)*abs(Y4-dY)/2 -abs(X1-bX)*abs(Y2-bY)/2	F()=abs(X1-aX)*abs(Y4-aY)/2 +abs(X3-cX)*abs(Y2-cY)



Case Value	1	6
Conditions	aZ>CV,bZ>CV,cZ>CV,dZ>CV	aZ<CV,bZ<CV,cZ<CV,dZ<CV
Equations	F()=abs(bX-aX)*abs(cY-bY)	F()=0

aX, bX, cX, dX: Coordination in X direction for vertice a, b, c, d;
aY, bY, cY, dY: Coordination in Y direction for vertice a, b, c, d;
aZ, bZ, cZ, dZ: Z values for vertice a, b, c, d;
X1, Y2, X3, Y4: intersections in the four edges;
CV: Contour value;
M: Mean value of Z of vertice a, b, c, d.

3. Code for the alternative algorithm

Grid() A one-dimensional array containing the Z values of each grid
contourvalue A variable containing the value of a contour

```
Public Function FootprintArea(Grid, contourvalue)
Dim i, Casevalue As Integer
Dim v1, v2, v3, v4 As Long ' record four vertices
Dim aZ, bZ, cZ, dZ, Average As Double ' record grids at four vertices of each element
Dim X1, Y2, X3, Y4 As Double
Dim aX, aY, bX, bY, cX, cY, dX, dY As Double
Dim Area As Double

Area = 0
If Model = "Beychok" Then

For i = 1 To (XnumberB - 1) * (YnumberB - 1) 'scan every element

'calculate the vertix number of each element
v1 = i + Fix((i - 1) / (XnumberB - 1)) v2 = v1 + 1 v4 = v1 + XnumberB v3 = v4 + 1

aX = ((v1 - 1) Mod XnumberB) * GridspaceX
aY = ((YnumberB - 1) / 2 - Fix(v1 / XnumberB)) * GridspaceY
bX = aX + GridspaceX bY = aY cX = bX cY = bY - GridspaceY
dX = aX dY = cY aZ = Grid(v1) bZ = Grid(v2) cZ = Grid(v3) dZ = Grid(v4)

'Interpolation
If (aZ > contourvalue And bZ < contourvalue) Or (aZ < contourvalue And bZ >
contourvalue) Then X1 = aX + (contourvalue - aZ) / (bZ - aZ) * GridspaceX
End If
If (bZ > contourvalue And cZ < contourvalue) Or (bZ < contourvalue And cZ >
contourvalue) Then Y2 = bY - (contourvalue - bZ) / (cZ - bZ) * GridspaceY
End If
If (cZ > contourvalue And dZ < contourvalue) Or (cZ < contourvalue And dZ >
contourvalue) Then X3 = dX + (contourvalue - dZ) / (cZ - dZ) * GridspaceX
End If
If (dZ > contourvalue And aZ < contourvalue) Or (dZ < contourvalue And aZ >
contourvalue) Then Y4 = aY - (contourvalue - aZ) / (dZ - aZ) * GridspaceY
End If

'set casevalue
If aZ < contourvalue And bZ < contourvalue And cZ < contourvalue And dZ <
contourvalue Then Casevalue = 1
End If
If aZ < contourvalue And bZ > contourvalue And cZ > contourvalue And dZ >
contourvalue Then Casevalue = 21
```

```
End If
If aZ > contourvalue And bZ < contourvalue And cZ > contourvalue And dZ >
contourvalue Then    Casevalue = 22
End If
If aZ > contourvalue And bZ > contourvalue And cZ < contourvalue And dZ >
contourvalue Then    Casevalue = 23
End If
If aZ > contourvalue And bZ > contourvalue And cZ > contourvalue And dZ <
contourvalue Then    Casevalue = 24
End If
If aZ < contourvalue And bZ < contourvalue And cZ > contourvalue And dZ >
contourvalue Then    Casevalue = 31
End If
If aZ > contourvalue And bZ < contourvalue And cZ < contourvalue And dZ >
contourvalue Then    Casevalue = 32
End If
If aZ > contourvalue And bZ > contourvalue And cZ < contourvalue And dZ <
contourvalue Then    Casevalue = 33
End If
If aZ < contourvalue And bZ > contourvalue And cZ > contourvalue And dZ <
contourvalue Then    Casevalue = 34
End If
If aZ > contourvalue And bZ < contourvalue And cZ < contourvalue And dZ <
contourvalue Then    Casevalue = 41
End If
If aZ < contourvalue And bZ > contourvalue And cZ < contourvalue And dZ <
contourvalue Then    Casevalue = 42
End If
If aZ < contourvalue And bZ < contourvalue And cZ > contourvalue And dZ <
contourvalue Then    Casevalue = 43
End If
If aZ < contourvalue And bZ < contourvalue And cZ < contourvalue And dZ >
contourvalue Then    Casevalue = 44
End If
If aZ < contourvalue And bZ > contourvalue And cZ < contourvalue And dZ >
contourvalue Then    Average = (aZ + bZ + cZ + dZ) / 4
  If Average > contourvalue Then       Casevalue = 511
  Else       Casevalue = 512
  End If
End If
If aZ > contourvalue And bZ < contourvalue And cZ > contourvalue And dZ <
contourvalue Then    Average = (aZ + bZ + cZ + dZ) / 4
  If Average > contourvalue Then       Casevalue = 521
  Else       Casevalue = 522
  End If
End If
```

```
If aZ > contourvalue And bZ > contourvalue And cZ > contourvalue And dZ >
contourvalue Then    Casevalue = 6
End If

Select Case Casevalue
Case 1
   Area = 0
Case 21
   Area = GridspaceX * GridspaceY - Abs(Y4 - aY) * Abs(X1 - aX) / 2
Case 22
   Area = GridspaceX * GridspaceY - Abs(Y2 - bY) * Abs(X1 - bX) / 2
Case 23
   Area = GridspaceX * GridspaceY - Abs(Y2 - cY) * Abs(X3 - cX) / 2
Case 24
   Area = GridspaceX * GridspaceY - Abs(Y4 - dY) * Abs(X3 - dX) / 2
Case 31
   Area = (Abs(Y2 - cY) + Abs(Y4 - dY)) * GridspaceX / 2
Case 32
   Area = (Abs(X1 - aX) + Abs(X3 - dX)) * GridspaceY / 2
Case 33
   Area = (Abs(Y2 - bY) + Abs(Y4 - aY)) * GridspaceX / 2
Case 34
   Area = (Abs(X1 - bX) + Abs(X3 - cX)) * GridspaceY / 2
Case 41
   Area = Abs(Y4 - aY) * Abs(X1 - aX) / 2
Case 42
   Area = Abs(Y2 - bY) * Abs(X1 - bX) / 2
Case 43
   Area = Abs(Y2 - cY) * Abs(X3 - cX) / 2
Case 44
   Area = Abs(Y4 - dY) * Abs(X3 - dX) / 2
Case 511
   Area = GridspaceX * GridspaceY - Abs(Y4 - aY) * Abs(X1 - aX) / 2 - Abs(Y2 - cY) *
Abs(X3 - cX) / 2
Case 512
   Area = Abs(Y2 - bY) * Abs(X1 - bX) / 2 + Abs(Y4 - dY) * Abs(X3 - dX) / 2
Case 521
   Area = GridspaceX * GridspaceY - Abs(X1 - bX) * Abs(Y2 - bY) / 2 - Abs(Y4 - dY) *
Abs(X3 - dX) / 2
Case 522
   Area = Abs(X1 - aX) * Abs(Y4 - aY) / 2 + Abs(Y2 - cY) * Abs(X3 - cX) / 2
Case 6
   Area = GridspaceX * GridspaceY
End Select
FootprintArea = FootprintArea + Area
Next i
```

```
End If

If Model = "ReadFile" Then
For i = 1 To (Xnumber - 1) * (Ynumber - 1)  'scan every element
'calculate the vertix number of each element
v1 = i + Fix((i - 1) / (Xnumber - 1)) v2 = v1 + 1 v4 = v1 + Xnumber v3 = v4 + 1

aX = XY(v1, 1) aY = XY(v1, 2) bX = XY(v2, 1) bY = XY(v2, 2)
cX = XY(v3, 1) cY = XY(v3, 2) dX = XY(v4, 1) dY = XY(v4, 2)
aZ = Grid(v1) bZ = Grid(v2) cZ = Grid(v3) dZ = Grid(v4)
GridspaceX = Abs(aX - bX) GridspaceY = Abs(bY - cY)

'Interpolation
If (aZ > contourvalue And bZ < contourvalue) Or (aZ < contourvalue And bZ > contourvalue) Then   X1 = aX + (contourvalue - aZ) / (bZ - aZ) * GridspaceX
End If
If (bZ > contourvalue And cZ < contourvalue) Or (bZ < contourvalue And cZ > contourvalue) Then   Y2 = bY + (contourvalue - bZ) / (cZ - bZ) * GridspaceY
End If
If (cZ > contourvalue And dZ < contourvalue) Or (cZ < contourvalue And dZ > contourvalue) Then   X3 = dX + (contourvalue - dZ) / (cZ - dZ) * GridspaceX
End If
If (dZ > contourvalue And aZ < contourvalue) Or (dZ < contourvalue And aZ > contourvalue) Then   Y4 = aY + (contourvalue - aZ) / (dZ - aZ) * GridspaceY
End If

'set casevalue
If aZ < contourvalue And bZ < contourvalue And cZ < contourvalue And dZ < contourvalue Then   Casevalue = 1
End If
If aZ < contourvalue And bZ > contourvalue And cZ > contourvalue And dZ > contourvalue Then   Casevalue = 21
End If
If aZ > contourvalue And bZ < contourvalue And cZ > contourvalue And dZ > contourvalue Then   Casevalue = 22
End If
If aZ > contourvalue And bZ > contourvalue And cZ < contourvalue And dZ > contourvalue Then   Casevalue = 23
End If
If aZ > contourvalue And bZ > contourvalue And cZ > contourvalue And dZ < contourvalue Then   Casevalue = 24
End If
If aZ < contourvalue And bZ < contourvalue And cZ > contourvalue And dZ > contourvalue Then   Casevalue = 31
End If
```

```
If aZ > contourvalue And bZ < contourvalue And cZ < contourvalue And dZ >
contourvalue Then   Casevalue = 32
End If
If aZ > contourvalue And bZ > contourvalue And cZ < contourvalue And dZ <
contourvalue Then   Casevalue = 33
End If
If aZ < contourvalue And bZ > contourvalue And cZ > contourvalue And dZ <
contourvalue Then   Casevalue = 34
End If
If aZ > contourvalue And bZ < contourvalue And cZ < contourvalue And dZ <
contourvalue Then   Casevalue = 41
End If
If aZ < contourvalue And bZ > contourvalue And cZ < contourvalue And dZ <
contourvalue Then   Casevalue = 42
End If
If aZ < contourvalue And bZ < contourvalue And cZ > contourvalue And dZ <
contourvalue Then   Casevalue = 43
End If
If aZ < contourvalue And bZ < contourvalue And cZ < contourvalue And dZ >
contourvalue Then   Casevalue = 44
End If
If aZ < contourvalue And bZ > contourvalue And cZ < contourvalue And dZ >
contourvalue Then   Average = (aZ + bZ + cZ + dZ) / 4
  If Average > contourvalue Then      Casevalue = 511
  Else      Casevalue = 512
  End If
End If
If aZ > contourvalue And bZ < contourvalue And cZ > contourvalue And dZ <
contourvalue Then   Average = (aZ + bZ + cZ + dZ) / 4
  If Average > contourvalue Then      Casevalue = 521
  Else      Casevalue = 522
  End If
End If
If aZ > contourvalue And bZ > contourvalue And cZ > contourvalue And dZ >
contourvalue Then   Casevalue = 6
End If

Select Case Casevalue
Case 1
  Area = 0
Case 21
  Area = GridspaceX * GridspaceY - Abs(Y4 - aY) * Abs(X1 - aX) / 2
Case 22
  Area = GridspaceX * GridspaceY - Abs(Y2 - bY) * Abs(X1 - bX) / 2
Case 23
  Area = GridspaceX * GridspaceY - Abs(Y2 - cY) * Abs(X3 - cX) / 2
```

```
Case 24
   Area = GridspaceX * GridspaceY - Abs(Y4 - dY) * Abs(X3 - dX) / 2
Case 31
   Area = (Abs(Y2 - cY) + Abs(Y4 - dY)) * GridspaceX / 2
Case 32
   Area = (Abs(X1 - aX) + Abs(X3 - dX)) * GridspaceY / 2
Case 33
   Area = (Abs(Y2 - bY) + Abs(Y4 - aY)) * GridspaceX / 2
Case 34
   Area = (Abs(X1 - bX) + Abs(X3 - cX)) * GridspaceY / 2
Case 41
   Area = Abs(Y4 - aY) * Abs(X1 - aX) / 2
Case 42
   Area = Abs(Y2 - bY) * Abs(X1 - bX) / 2
Case 43
   Area = Abs(Y2 - cY) * Abs(X3 - cX) / 2
Case 44
   Area = Abs(Y4 - dY) * Abs(X3 - dX) / 2
Case 511
   Area = GridspaceX * GridspaceY - Abs(Y4 - aY) * Abs(X1 - aX) / 2 - Abs(Y2 - cY) *
Abs(X3 - cX) / 2
Case 512
   Area = Abs(Y2 - bY) * Abs(X1 - bX) / 2 + Abs(Y4 - dY) * Abs(X3 - dX) / 2
Case 521
   Area = GridspaceX * GridspaceY - Abs(X1 - bX) * Abs(Y2 - bY) / 2 - Abs(Y4 - dY) *
Abs(X3 - dX) / 2
Case 522
   Area = Abs(X1 - aX) * Abs(Y4 - aY) / 2 + Abs(Y2 - cY) * Abs(X3 - cX) / 2
Case 6
   Area = GridspaceX * GridspaceY
End Select
FootprintArea = FootprintArea + Area
Next i
End If
End Function
```

2.5 Code for calculating population in an annoyace, concentration, or probability contours, *N(A)*, *N(C)*, or *N(P)* and annoyance-, concentration- and probability-weighted populations, $N_W(A)$, $N_W(C)$, $N_W(P)$

Grid()	A one-dimensional array containing the Z values of each grid
contourvalue	A variable containing the value of a contour
PIC	Set as "fa" to calculate N(A), N(C) or N(P)
	Set as "pwfa" to calculate $N_W(A)$, $N_W(C)$, $N_W(P)$

Public Function NumPeople(Grid, contourvalue, PIC)

```
Dim i, j, PDensity As Long
Dim v1, v2, v3, v4 As Long
Dim aX, aY, aZ, bX, bY, bZ, cX, cY, cZ, dX, dY, dZ As Double
Dim ElementArea, EVolume As Double

NumPeople = 0

If Model = "Beychok" Then

For i = 1 To (XnumberB - 1) * (YnumberB - 1) 'scan every element
   'calculate the vertix number of each element
   v1 = i + Fix((i - 1) / (XnumberB - 1))   v2 = v1 + 1
   v4 = v1 + XnumberB   v3 = v4 + 1

   aX = ((v1 - 1) Mod XnumberB) * GridspaceX
   aY = ((YnumberB - 1) / 2 - Fix(v1 / XnumberB)) * GridspaceY
   aZ = Grid(v1)

   bX = aX + GridspaceX   bY = aY   bZ = Grid(v2)
   cX = bX   cY = bY - GridspaceY   cZ = Grid(v3)
   dX = aX   dY = cY   dZ = Grid(v4)
   'Get the population density of the element
   For j = 1 To NumBlock
      If aX - XYP1(j, 1) >= 0 And aY - XYP1(j, 2) <= 0 And cX - XYP2(j, 1) <= 0 And
cY - XYP2(j, 2) >= 0 Then
         PDensity = Population(j)
      End If
   Next j

'Interpolation
If (aZ > contourvalue And bZ < contourvalue) Or (aZ < contourvalue And bZ >
contourvalue) Then   X1 = aX + (contourvalue - aZ) / (bZ - aZ) * GridspaceX
End If
If (bZ > contourvalue And cZ < contourvalue) Or (bZ < contourvalue And cZ >
contourvalue) Then   Y2 = bY - (contourvalue - bZ) / (cZ - bZ) * GridspaceY
```

```
End If
If (cZ > contourvalue And dZ < contourvalue) Or (cZ < contourvalue And dZ >
contourvalue) Then    X3 = dX + (contourvalue - dZ) / (cZ - dZ) * GridspaceX
End If
If (dZ > contourvalue And aZ < contourvalue) Or (dZ < contourvalue And aZ >
contourvalue) Then    Y4 = aY - (contourvalue - aZ) / (dZ - aZ) * GridspaceY
End If

'set casevalue
If aZ < contourvalue And bZ < contourvalue And cZ < contourvalue And dZ <
contourvalue Then    Casevalue = 1
End If
If aZ < contourvalue And bZ > contourvalue And cZ > contourvalue And dZ >
contourvalue Then    Casevalue = 21
End If
If aZ > contourvalue And bZ < contourvalue And cZ > contourvalue And dZ >
contourvalue Then    Casevalue = 22
End If
If aZ > contourvalue And bZ > contourvalue And cZ < contourvalue And dZ >
contourvalue Then    Casevalue = 23
End If
If aZ > contourvalue And bZ > contourvalue And cZ > contourvalue And dZ <
contourvalue Then    Casevalue = 24
End If
If aZ < contourvalue And bZ < contourvalue And cZ > contourvalue And dZ >
contourvalue Then    Casevalue = 31
End If
If aZ > contourvalue And bZ < contourvalue And cZ < contourvalue And dZ >
contourvalue Then    Casevalue = 32
End If
If aZ > contourvalue And bZ > contourvalue And cZ < contourvalue And dZ <
contourvalue Then    Casevalue = 33
End If
If aZ < contourvalue And bZ > contourvalue And cZ > contourvalue And dZ <
contourvalue Then    Casevalue = 34
End If
If aZ > contourvalue And bZ < contourvalue And cZ < contourvalue And dZ <
contourvalue Then    Casevalue = 41
End If
If aZ < contourvalue And bZ > contourvalue And cZ < contourvalue And dZ <
contourvalue Then    Casevalue = 42
End If
If aZ < contourvalue And bZ < contourvalue And cZ > contourvalue And dZ <
contourvalue Then    Casevalue = 43
End If
```

```
If aZ < contourvalue And bZ < contourvalue And cZ < contourvalue And dZ >
contourvalue Then   Casevalue = 44
End If
If aZ < contourvalue And bZ > contourvalue And cZ < contourvalue And dZ >
contourvalue Then   Average = (aZ + bZ + cZ + dZ) / 4
  If Average > contourvalue Then      Casevalue = 511
  Else      Casevalue = 512
  End If
End If
If aZ > contourvalue And bZ < contourvalue And cZ > contourvalue And dZ <
contourvalue Then   Average = (aZ + bZ + cZ + dZ) / 4
  If Average > contourvalue Then      Casevalue = 521
  Else      Casevalue = 522
  End If
End If
If aZ > contourvalue And bZ > contourvalue And cZ > contourvalue And dZ >
contourvalue Then   Casevalue = 6
End If

If PIC = "fa" Then
aZ = 1 bZ = 1 cZ = 1 dZ = 1 CV = 1
Else CV = contourvalue
End If

Select Case Casevalue
Case 1
  EVolume = 0
Case 21
  EVolume = Volume(aX, aY, aZ, bX, bY, bZ, cX, cY, cZ) + Volume(cX, cY, cZ, dX,
dY, dZ, aX, aY, aZ) - Volume(aX, Y4, CV, aX, aY, aZ, X1, aY, CV)
Case 22
  EVolume = Volume(aX, aY, aZ, bX, bY, bZ, cX, cY, cZ) + Volume(cX, cY, cZ, dX,
dY, dZ, aX, aY, aZ) - Volume(X1, bY, CV, bX, bY, bZ, bX, Y2, CV)
Case 23
  EVolume = Volume(aX, aY, aZ, bX, bY, bZ, cX, cY, cZ) + Volume(cX, cY, cZ, dX,
dY, dZ, aX, aY, aZ) - Volume(cX, Y2, CV, cX, cY, cZ, X3, cY, CV)
Case 24
  EVolume = Volume(aX, aY, aZ, bX, bY, bZ, cX, cY, cZ) + Volume(cX, cY, cZ, dX,
dY, dZ, aX, aY, aZ) - Volume(X3, dY, CV, dX, dY, dZ, dX, Y4, CV)
Case 31
  EVolume = Volume(cX, Y2, CV, cX, cY, cZ, dX, dY, dZ) + Volume(dX, dY, dZ, dX,
Y4, CV, cX, Y2, CV)
Case 32
  EVolume = Volume(aX, aY, aZ, X1, aY, CV, X3, dY, CV) + Volume(X3, dY, CV,
dX, dY, dZ, aX, aY, aZ)
Case 33
```

```
  EVolume = Volume(aX, aY, aZ, bX, bY, bZ, bX, Y2, CV) + Volume(bX, Y2, CV, aX,
Y4, CV, aX, aY, aZ)
Case 34
  EVolume = Volume(X1, bY, CV, bX, bY, bZ, cX, cY, cZ) + Volume(cX, cY, cZ, X3,
cY, CV, X1, bY, CV)
Case 41
  EVolume = Volume(aX, Y4, CV, aX, aY, aZ, X1, aY, CV)
Case 42
  EVolume = Volume(X1, bY, CV, bX, bY, bZ, bX, Y2, CV)
Case 43
   EVolume = Volume(cX, Y2, CV, cX, cY, cZ, X3, cY, CV)
Case 44
  EVolume = Volume(X3, dY, CV, dX, dY, dZ, dX, Y4, CV)
Case 511
  EVolume = Volume(aX, aY, aZ, bX, bY, bZ, cX, cY, cZ) + Volume(cX, cY, cZ, dX,
dY, dZ, aX, aY, aZ) - Volume(aX, Y4, CV, aX, aY, aZ, X1, aY, CV) - Volume(cX, Y2,
CV, cX, cY, cZ, X3, cY, CV)
Case 512
  EVolume = Volume(X1, bY, CV, bX, bY, bZ, bX, Y2, CV) + Volume(X3, dY, CV,
dX, dY, dZ, dX, Y4, CV)
Case 521
  EVolume = Volume(aX, aY, aZ, bX, bY, bZ, cX, cY, cZ) + Volume(cX, cY, cZ, dX,
dY, dZ, aX, aY, aZ) - Volume(X1, bY, CV, bX, bY, bZ, bX, Y2, CV) - Volume(X3, dY,
CV, dX, dY, dZ, dX, Y4, CV)
Case 522
  EVolume = Volume(aX, Y4, CV, aX, aY, aZ, X1, aY, CV) + Volume(cX, Y2, CV,
cX, cY, cZ, X3, cY, CV)
Case 6
  EVolume = Volume(aX, aY, aZ, bX, bY, bZ, cX, cY, cZ) + Volume(cX, cY, cZ, dX,
dY, dZ, aX, aY, aZ)
End Select
NumPeople = NumPeople + PDensity * EVolume
Next i
End If

If Model = "ReadFile" Then

For i = 1 To (Xnumber - 1) * (Ynumber - 1) 'scan every element
  'calculate the vertix number of each element
  v1 = i + Fix((i - 1) / (Xnumber - 1))
  v2 = v1 + 1    v4 = v1 + Xnumber    v3 = v4 + 1

  aX = XY(v1, 1)   aY = XY(v1, 2)   aZ = Grid(v1)
  bX = XY(v2, 1)   bY = XY(v2, 2)   bZ = Grid(v2)
  cX = XY(v3, 1)   cY = XY(v3, 2)   cZ = Grid(v3)
  dX = XY(v4, 1)   dY = XY(v4, 2)   dZ = Grid(v4)
```

```
  GridspaceX = Abs(aX - bX)   GridspaceY = Abs(bY - cY)

  For j = 1 To NumBlock
    If aX - XYP1(j, 1) >= 0 And aY - XYP1(j, 2) >= 0 And cX - XYP2(j, 1) <= 0 And
cY - XYP2(j, 2) <= 0 Then          PDensity = Population(j)
    End If
  Next j

'Interpolation
If (aZ > contourvalue And bZ < contourvalue) Or (aZ < contourvalue And bZ >
contourvalue) Then   X1 = aX + (contourvalue - aZ) / (bZ - aZ) * GridspaceX
End If
If (bZ > contourvalue And cZ < contourvalue) Or (bZ < contourvalue And cZ >
contourvalue) Then   Y2 = bY + (contourvalue - bZ) / (cZ - bZ) * GridspaceY
End If
If (cZ > contourvalue And dZ < contourvalue) Or (cZ < contourvalue And dZ >
contourvalue) Then   X3 = dX + (contourvalue - dZ) / (cZ - dZ) * GridspaceX
End If
If (dZ > contourvalue And aZ < contourvalue) Or (dZ < contourvalue And aZ >
contourvalue) Then   Y4 = aY + (contourvalue - aZ) / (dZ - aZ) * GridspaceY
End If
'set casevalue
If aZ < contourvalue And bZ < contourvalue And cZ < contourvalue And dZ <
contourvalue Then   Casevalue = 1
End If
If aZ < contourvalue And bZ > contourvalue And cZ > contourvalue And dZ >
contourvalue Then   Casevalue = 21
End If
If aZ > contourvalue And bZ < contourvalue And cZ > contourvalue And dZ >
contourvalue Then   Casevalue = 22
End If
If aZ > contourvalue And bZ > contourvalue And cZ < contourvalue And dZ >
contourvalue Then   Casevalue = 23
End If
If aZ > contourvalue And bZ > contourvalue And cZ > contourvalue And dZ <
contourvalue Then   Casevalue = 24
End If
If aZ < contourvalue And bZ < contourvalue And cZ > contourvalue And dZ >
contourvalue Then   Casevalue = 31
End If
If aZ > contourvalue And bZ < contourvalue And cZ < contourvalue And dZ >
contourvalue Then   Casevalue = 32
End If
If aZ > contourvalue And bZ > contourvalue And cZ < contourvalue And dZ <
contourvalue Then   Casevalue = 33
End If
```

```
If aZ < contourvalue And bZ > contourvalue And cZ > contourvalue And dZ <
contourvalue Then    Casevalue = 34
End If
If aZ > contourvalue And bZ < contourvalue And cZ < contourvalue And dZ <
contourvalue Then    Casevalue = 41
End If
If aZ < contourvalue And bZ > contourvalue And cZ < contourvalue And dZ <
contourvalue Then    Casevalue = 42
End If
If aZ < contourvalue And bZ < contourvalue And cZ > contourvalue And dZ <
contourvalue Then    Casevalue = 43
End If
If aZ < contourvalue And bZ < contourvalue And cZ < contourvalue And dZ >
contourvalue Then    Casevalue = 44
End If
If aZ < contourvalue And bZ > contourvalue And cZ < contourvalue And dZ >
contourvalue Then    Average = (aZ + bZ + cZ + dZ) / 4
   If Average > contourvalue Then        Casevalue = 511
   Else        Casevalue = 512
   End If
End If
If aZ > contourvalue And bZ < contourvalue And cZ > contourvalue And dZ <
contourvalue Then    Average = (aZ + bZ + cZ + dZ) / 4
   If Average > contourvalue Then        Casevalue = 521
   Else        Casevalue = 522
   End If
End If
If aZ > contourvalue And bZ > contourvalue And cZ > contourvalue And dZ >
contourvalue Then    Casevalue = 6
End If
If PIC = "fa" Then
aZ = 1 bZ = 1 cZ = 1 dZ = 1 CV = 1
Else CV = contourvalue
End If
Select Case Casevalue
Case 1
   EVolume = 0
Case 21
   EVolume = Volume(aX, aY, aZ, bX, bY, bZ, cX, cY, cZ) + Volume(cX, cY, cZ, dX,
dY, dZ, aX, aY, aZ) - Volume(aX, Y4, CV, aX, aY, aZ, X1, aY, CV)
Case 22
   EVolume = Volume(aX, aY, aZ, bX, bY, bZ, cX, cY, cZ) + Volume(cX, cY, cZ, dX,
dY, dZ, aX, aY, aZ) - Volume(X1, bY, CV, bX, bY, bZ, bX, Y2, CV)
Case 23
   EVolume = Volume(aX, aY, aZ, bX, bY, bZ, cX, cY, cZ) + Volume(cX, cY, cZ, dX,
dY, dZ, aX, aY, aZ) - Volume(cX, Y2, CV, cX, cY, cZ, X3, cY, CV)
```

```
Case 24
  EVolume = Volume(aX, aY, aZ, bX, bY, bZ, cX, cY, cZ) + Volume(cX, cY, cZ, dX,
dY, dZ, aX, aY, aZ) - Volume(X3, dY, CV, dX, dY, dZ, dX, Y4, CV)
Case 31
  EVolume = Volume(cX, Y2, CV, cX, cY, cZ, dX, dY, dZ) + Volume(dX, dY, dZ, dX,
Y4, CV, cX, Y2, CV)
Case 32
  EVolume = Volume(aX, aY, aZ, X1, aY, CV, X3, dY, CV) + Volume(X3, dY, CV,
dX, dY, dZ, aX, aY, aZ)
Case 33
  EVolume = Volume(aX, aY, aZ, bX, bY, bZ, bX, Y2, CV) + Volume(bX, Y2, CV, aX,
Y4, CV, aX, aY, aZ)
Case 34
  EVolume = Volume(X1, bY, CV, bX, bY, bZ, cX, cY, cZ) + Volume(cX, cY, cZ, X3,
cY, CV, X1, bY, CV)
Case 41
  EVolume = Volume(aX, Y4, CV, aX, aY, aZ, X1, aY, CV)
Case 42
  EVolume = Volume(X1, bY, CV, bX, bY, bZ, bX, Y2, CV)
Case 43
  EVolume = Volume(cX, Y2, CV, cX, cY, cZ, X3, cY, CV)
Case 44
  EVolume = Volume(X3, dY, CV, dX, dY, dZ, dX, Y4, CV)
Case 511
  EVolume = Volume(aX, aY, aZ, bX, bY, bZ, cX, cY, cZ) + Volume(cX, cY, cZ, dX,
dY, dZ, aX, aY, aZ) - Volume(aX, Y4, CV, aX, aY, aZ, X1, aY, CV) - Volume(cX, Y2,
CV, cX, cY, cZ, X3, cY, CV)
Case 512
  EVolume = Volume(X1, bY, CV, bX, bY, bZ, bX, Y2, CV) + Volume(X3, dY, CV,
dX, dY, dZ, dX, Y4, CV)
Case 521
  EVolume = Volume(aX, aY, aZ, bX, bY, bZ, cX, cY, cZ) + Volume(cX, cY, cZ, dX,
dY, dZ, aX, aY, aZ) - Volume(X1, bY, CV, bX, bY, bZ, bX, Y2, CV) - Volume(X3, dY,
CV, dX, dY, dZ, dX, Y4, CV)
Case 522
  EVolume = Volume(aX, Y4, CV, aX, aY, aZ, X1, aY, CV) + Volume(cX, Y2, CV,
cX, cY, cZ, X3, cY, CV)
Case 6
  EVolume = Volume(aX, aY, aZ, bX, bY, bZ, cX, cY, cZ) + Volume(cX, cY, cZ, dX,
dY, dZ, aX, aY, aZ)
End Select
NumPeople = NumPeople + PDensity * EVolume
Next i
End If
End Function
```

2.6 Code for calculating total annoyance-, concentration- and probability-weighted populations, $N_{WC}(T)$, $N_{WP}(T)$, $N_{WA}(T)$

Grid() A one-dimensional array containing the Z values of each grid

```
Dim i, j, PDensity As Long
Dim v1, v2, v3, v4 As Long
Dim aX, aY, aZ, bX, bY, bZ, cX, cY, cZ, dX, dY, dZ As Double

TotalPeople = 0 People_Max = 0
If Model = "Beychok" Then

For i = 1 To (XnumberB - 1) * (YnumberB - 1) 'scan every element

  'calculate the vertix number of each element
  v1 = i + Fix((i - 1) / (XnumberB - 1))
  v2 = v1 + 1   v4 = v1 + XnumberB   v3 = v4 + 1

  aX = ((v1 - 1) Mod XnumberB) * GridspaceX
  aY = ((YnumberB - 1) / 2 - Fix(v1 / XnumberB)) * GridspaceY
  aZ = Grid(v1)
  bX = aX + GridspaceX   bY = aY   bZ = Grid(v2)
  cX = bX   cY = bY - GridspaceY   cZ = Grid(v3)
  dX = aX   dY = cY   dZ = Grid(v4)
  'Get the population density of the element
  For j = 1 To NumBlock
    If aX - XYP1(j, 1) >= 0 And aY - XYP1(j, 2) <= 0 And cX - XYP2(j, 1) <= 0 And
cY - XYP2(j, 2) >= 0 Then
      PDensity = Population(j)
    End If
  Next j
If Parameter = "fa" Then
aZ = 1 bZ = 1 cZ = 1 dZ = 1
End If
    Temp = PDensity * (Volume(aX, aY, aZ, bX, bY, bZ, cX, cY, cZ) + Volume(cX,
cY, cZ, dX, dY, dZ, aX, aY, aZ))
  TotalPeople = TotalPeople + Temp
  If Temp > People_Max Then
    People_Max = Temp / 1000000
    People_Maxx = aX
    People_Maxy = aY
  End If
Next i
End If

If Model = "ReadFile" Then
```

```
For i = 1 To (Xnumber - 1) * (Ynumber - 1) 'scan every element
   'calculate the vertix number of each element
   v1 = i + Fix((i - 1) / (Xnumber - 1))   v2 = v1 + 1   v4 = v1 + Xnumber v3 = v4 + 1
   aX = XY(v1, 1)    aY = XY(v1, 2)   aZ = Grid(v1)
   bX = XY(v2, 1)   bY = XY(v2, 2)   bZ = Grid(v2)
   cX = XY(v3, 1)   cY = XY(v3, 2)   cZ = Grid(v3)
   dX = XY(v4, 1)   dY = XY(v4, 2)   dZ = Grid(v4)

   For j = 1 To NumBlock
      If aX - XYP1(j, 1) >= 0 And aY - XYP1(j, 2) >= 0 And cX - XYP2(j, 1) <= 0 And
cY - XYP2(j, 2) <= 0 Then
         PDensity = Population(j)
      End If
   Next j
   If Parameter = "fa" Then
   aZ = 1   bZ = 1   cZ = 1   dZ = 1
   End If
   Temp = PDensity * (Volume(aX, aY, aZ, bX, bY, bZ, cX, cY, cZ) + Volume(cX, cY,
cZ, dX, dY, dZ, aX, aY, aZ))
   TotalPeople = TotalPeople + Temp
   If Temp > People_Max Then
      People_Max = Temp
      People_Maxx = aX
      People_Maxy = aY
   End If
Next i
End If
End Function
```