Development of an air quality model for BCL Limited

by

Tiroyaone Tshukudu

Submitted in partial fulfilment of the requirements
for the degree of Master of Engineering (Environmental Engineering)
in the Faculty of Engineering, Built Environment and Information Technology

University of Pretoria

November 2003

Acknowledgements

My foremost acknowledgement is extended to Francois Friend for his advice on this research, inspiration and technical assistance throughout the course of this investigation.

Thank you to Mike Hagger and the personnel of the Environmental section at BCL Limited for all the time and data used in this research.

I would like to express my gratitude to the Air Pollution Control Division of the government of Botswana for making available meteorological and air quality data from Selebi-Phikwe.

Last I want to thank my wife for her encouragement, love and patience during my trips between Pretoria and Gaborone.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Development of an air quality model for BCL Limited

Author : T Tshukudu

Supervisor : JFC Friend

Department : Chemical Engineering (Environmental Engineering Group)

Degree : Master of Engineering (Environmental Engineering)

Synopsis

In order to understand the impact from a point source of pollution a comprehensive air quality monitoring programme must be in place. Measurement stations are at fixed locations and can only measure the relevant concentration if the wind is blowing in a particular direction. With changing wind direction, a measurement station needs to be coupled with a dispersion model to predict the impact from a point source, such as the BCL Limited smelter. The smelter produces 55 500 tonnes per annum of nickel/copper granules; in the process emitting waste gases to the atmosphere through a 153 m stack.

The EEGAIR-BCL1 dispersion model was developed for BCL Limited to complement the air quality monitoring programme in the Selebi-Phikwe area. The model was developed for the smelter specifically, using local meteorological data at Selebi-Phikwe and smelter specific emission data. Results from the model were tested against data obtained from existing measurements stations at Selebi-Phikwe for May 2002, using statistical analyses. The average index of agreement indicates that the model predictions were more accurate for Kopano (0,44) and SPSS (0,25) stations than for WUC (0,08) station. Yearly averaged emission data was used for model input and better correlation can be expected when using actual hourly and/or monthly emission data.

Based on the EEGAIR-BCL1 dispersion model results, it was found that the highest impact from the smelter stack was at a distance of 4 km to 7 km west from the stack. The model results indicate that, on average, the impact from the smelter stack in residential areas of Selebi-Phikwe was between 0 and 50 µg/m^3 for May 2002.

KEYWORDS: point source, air quality, dispersion model, smelter, emission.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

An integrated approach towards environmental management combines pre-active and preventative processes and procedures to maintain the environment in a good condition for short and long term sustainable use. A similar approach is followed in the field of air quality management, using suitable information systems to base educated decisions upon. At present, air quality information systems combine the latest sensors and monitor technologies with data transfer; database developments, statistical and numerical models, quality assurance and advanced computer platforms for processing, distributing and presenting air quality data and dispersion model results (Sivertsen, 2000).

Atmospheric dispersion modelling is an attempt to represent, by mathematical or statistical means, the observed phenomena of pollutants being diluted and dispersed in the open atmosphere (Cooper and Alley, 1994). These developed models are essential for any air quality management system and together with measured air quality data, these models become practically more useful than single point measurement data. Measurement data can be used to validate a model that can later be used to either predict the air quality in areas with no measurement stations or, predict the impact from a new source in a given area.

The use of these models is useful in estimating or determining the atmospheric impact from an industry. One such industry is that of nickel copper production, where substantial emissions of sulphur compounds, trace elements and particulates to the atmosphere take place. Control of such emissions has resulted in reduced production at some smelters due to the enforcement of new air quality regulations.

In the Selebi-Phikwe area, BCL Limited and the government of Botswana are operating three stations equipped with continuous gas analysers to monitor the sulphur dioxide emission from the smelter. Measurement stations are at fixed locations and can only measure the relevant concentration if the wind is blowing in a particular direction. With changing wind direction, a measurement station needs to be coupled with a dispersion model to predict the impact from a point source, such as the BCL Limited smelter.

A large number of dispersion models are available to assist employees from government and industries with air quality management. However, the majority of models available in the market are very expensive, often difficult to use and sometimes cannot be adjusted for specific areas. The objectives of this research are therefore to develop, validate and document an air dispersion model for emissions from the BCL Limited smelter. The model will be developed for the smelter specifically, using local meteorological data at Selebi-Phikwe. Results from the model will be tested against data obtained from existing measurements stations at Selebi-Phikwe.

In Chapter 2 the literature survey includes information and equations required in developing the proposed model; and a brief background of process operations at BCL Limited. The program code developed for the model is written in Delphi programming language and explained in Chapter 3. The results from testing the model using the emissions from BCL Limited, the local meteorological and measurement stations data are given in Chapter 4. Conclusions and recommendations as a result of this research are provided in Chapter 5.

CHAPTER 2

Literature Survey

2.1 INTRODUCTION

Government departments, agencies and local authorities increasingly rely on atmospheric dispersion models for making decisions related to air quality and traffic management, urban planning and public health. As a result, the community using dispersion models is becoming large and more diverse (Vardoulakis *et. al.*, 2002). According to Venkatram *et al.* (2001), the past ten years has seen the development of several dispersion models that attempt incorporating current understanding of micrometeorology and dispersion. Studies indicate that these models provide better estimates of observed concentrations than older models based on the empirical Pasquill-Gifford dispersion curves (Venkatram *et al.,* 2001).

Atmospheric dispersion models are widely used to study the relationship between air quality and emissions sources as a function of meteorology and source conditions. These simulation models are useful for the prediction of day-to-day variation in air quality, land use planning, selection of sites for industries and location of monitoring stations. Mathematical models used in air management range from simple empirical models to complex numerical models. In general, these models are based on the concentration equation governing the pollutant mass that is consistent with the physical principle of mass conservation. The governing equation can be solved with initial and boundary conditions for a given input of the source emission rates (Ku, 1984). Venkatram *et al.* (2001) stated that many different models have been tried with varying degree of success, with the Gaussian based models the most widely used today.

Gaussian plume or puff models are commonly used to determine ambient air quality. These models assume that the concentrations at a downward distance from the source are normally distributed about the plume centre line.

Gaussian plume models are steady state type models and assume a uniform horisontal wind and source strength for the period of simulation. Gaussian puff models have been developed to remedy some of the deficiencies inherent in the plume models. The puff models allow for the situation in which the release of a constituent occur for a short duration or in which meteorological conditions may change significantly during the average time. In particular, these models allow for variation of horisontal wind, source emission rates and dispersion parameters (Ku, 1984).

2.2 GOVERNING EQUATION

The Gaussian plume equation calculates steady state concentration at a point located downwind from the source. Equation 2.1 is based on Gaussian (normal) probability distribution of the concentration (particle density) in both the vertical and horisontal direction perpendicular to the plume centreline (Figure 2.1). It models the dispersion of a non-reactive gaseous pollutant from an elevated source (Cooper and Alley, 1994).

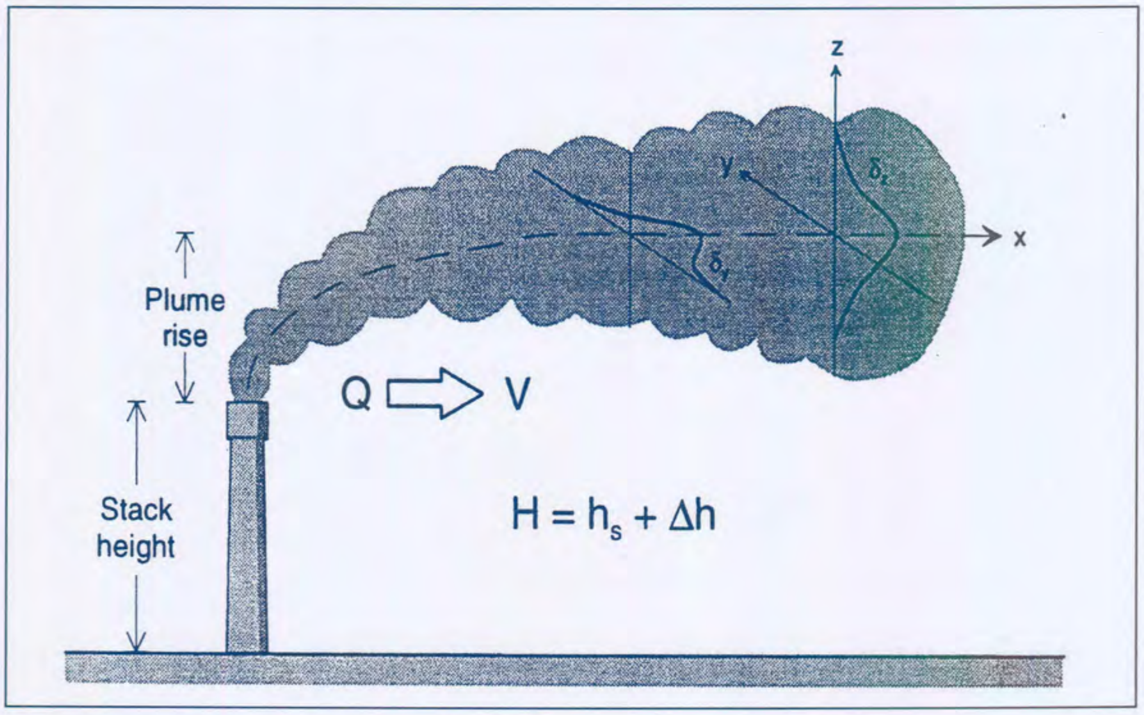


Figure 2.1 Coordinate system of the Gaussian plume model.

$$C_{x,y,z} = \frac{Q}{2\pi u \sigma_y \sigma_z} \exp\left[-\frac{y^2}{2{\sigma_y}^2}\right]\left\{\exp\left[-\frac{(z-H)^2}{2{\sigma_z}^2}\right] + \exp\left[-\frac{(z+H)}{2{\sigma_z}^2}\right]\right\} \qquad [2.1]$$

where $C_{x,y,z}$ = steady state concentration at a point (x, y, z) in g/m^3;

Q = pollutant emission rate in g/s;

π = constant pi = 3,14159;

u = average wind speed at stack height in m/s;

σ_y, σ_z = horisontal and vertical dispersion parameters in m (functions of downwind distance x and atmospheric stability);

y = horisontal distance from plume centreline in m;

z = vertical distance from ground level in m; and

H = effective stack height (H = h + Δh, where h = physical stack height and Δh = plume rise, all in m).

General relationships represented by Equation 2.1 (Cooper and Alley, 1994) are as follows:

- downwind concentration at any location is directly proportional to the source strength;
- downwind groundlevel (z=0) concentration is generally inversely proportional to wind speed;
- because σ_y and σ_z increases with downwind distance, the elevated plume centreline concentration continually declines with increasing x. However, groundlevel centreline concentration increases, go through a maximum, and then decreases with increasing distance away from the stack.
- the dispersion parameters, σ_y and σ_z increase with increasing atmospheric turbulence; and
- the maximum groundlevel concentration decreases as the effective stack height increases.

2.3 CONCENTRATION DEPENDENCE ON AVERAGING TIME

According to Cooper and Alley (1994) the concentrations predicted by Equation 2.1 using σ_y and σ_z values are 10 minutes average concentrations. For a longer averaging time, it is expected that the average concentration will be less compared to that predicted for a shorter time, owing to wind shifts and turbulence diffusion. The concentrations at two averaging times is approximated by the following power law (Cooper and Alley, 1994):

$$C_t = C_o \left[\frac{t_o}{t} \right]^p \qquad [2.2]$$

where C_t = concentration at averaging time t;

C_o = concentration at averaging time t_o; and

p = exponent and ranges between 0,2 and 0,5 depending on new averaging time.

2.4 ATMOSPHERIC STABILITY

Transport and diffusion of stack emissions occur within the atmospheric boundary layer where most of the local effects of the earth-air interface occur. The diffusion of air pollutants in the lower atmosphere is strongly influenced by the local atmospheric stability (Bøhler, 1987). Variations in the atmospheric stability serve to explain qualitatively the dilution of pollutants in the atmosphere. Stability is an excellent parameter for classifying a variety of meteorological and other statistics relating to air pollution (Ku, 1984). The stability classification proposed by Pasquill defined the following six turbulent classes (Cooper and Alley, 1994):

A = most unstable,

B = moderately unstable,

C = slightly unstable,

D = neutral,

E = slightly stable, and

F = most stable.

These stability parameters (A-F) are indicators of atmospheric turbulence. The stability parameters at any given time depend upon the following (USEPA, 1992):

- static stability – related to the changes in temperature with height,
- thermal turbulence - caused by heating of the ground at groundlevel, and
- mechanical turbulence – a function of wind speed and surface roughness.

The meteorological data used to estimate the stability class of the atmosphere are based on the angle of the sun, the extent of cloud cover, and surface wind speed (Cooper and Alley, 1994). However, this method is too subjective and empirical. The other method of estimating the Pasquill-Gifford (P-G) stability class is by using wind speed, solar radiation and vertical temperature gradient; see Table 2.1 (USEPA, 2000).

2.5 WIND SPEED

Air in motion near the earth's surface is retarded by friction that varies directly with surface roughness. The mean transport wind speed should be representative of the conditions through the vertical height interval in which the plume is dispersing. Wind speed affects dispersion in three distinct ways, namely (Colls, 1997):

- any emission is diluted by a factor proportional to the wind speed past the source,
- the wind creates mechanical turbulence that increases mixing and dilution, and
- a buoyancy source is kept closer to its release height at higher wind speeds.

Table 2.1 Determination of Pasquill-Gifford stability categories from solar radiation and vertical temperature gradient.

DAYTIME				
Wind speed (m/s)	Solar radiation (W/m^2)			
	> 925	925 – 675	675 - 175	< 175
< 2	A	A	B	D
2 – 3	A	B	C	D
3 – 5	B	B	C	D
5 – 6	C	C	D	D
> 6	C	D	D	D

NIGHTTIME		
Wind speed (m/s)	Vertical temperature gradient	
	< 0	> 0
< 2,0	E	F
2,0 – 2,5	D	E
> 2,5	D	D

The wind speed in the lower atmosphere varies with height above the ground and can be approximated by the following power law (Colls, 1997).

$$\frac{u}{u_o} = \left(\frac{z}{z_o}\right)^p \qquad [2.3]$$

where u = wind speed at height z in m/s,

u_o = wind speed at reference height z_o in m/s,

z = release height in m,

z_o = reference height in m, and

p = dimensionless wind speed exponent that varies with atmospheric stability class and surface roughness, see Table 2.2 (Colls, 997).

Table 2.2 Variation of wind speed exponent with atmospheric stability.

Stability class	p
A (most unstable)	0,07
B	0,07
C	0,10
D	0,15
E	0,35
F (most stable)	0,55

The power law equation may be used to adjust wind speed over a height range from about 10 m to 300 m. For release heights below 10 m, the reference wind speed should be used without adjustments (USEPA, 1992).

2.6 DISPERSION PARAMETERS

The rates of dispersion in the atmosphere are highly dependent on the atmospheric stability. The dispersion parameters, σ_y and σ_z are defined as the standard deviations of the concentration distribution in the y and z directions, respectively; and are estimated by (Cooper and Alley, 1994):

$$\sigma_y = ax^b \qquad [2.4]$$

$$\sigma_z = cx^d + f \qquad [2.5]$$

where a, b, c, d and f (see Table 2.3; Cooper and Alley, 1994) are constants that are dependent on the atmospheric stability and on distance x (with x in km), and are used for calculating dispersion coefficients as a function of downwind distance and atmospheric stability.

Table 2.3 Constants for calculating dispersion coefficients.

Stability	Constants							
	a	b	x < 1 km			x > 1km		
			c	d	f	c	d	f
A	213,0	0,894	440,80	1,941	9,27	459,7	2,094	-9,6
B	156,0	0,894	106,60	1,149	3,30	108,2	1,098	2,0
C	104,0	0,894	61,00	0,911	0,00	61,0	0,911	0,0
D	68,0	0,894	33,20	0,725	-1,70	44,5	0,516	-13,0
E	50,5	0,894	22,80	0,678	-1,30	55,4	0,305	-34,0
F	34,0	0,894	14,35	0,740	-0,35	62,6	0,180	-48,6

2.7 DOWNWIND AND CROSSWIND DISTANCE IN A WIND FIELD

In wind field modelling, interpolation techniques to evaluate velocity vectors in specific locations are often employed. According to Douglas (1996), one of the most commonly used techniques for interpolation between points (scatter or receptor points) with known values for a certain variable is the inverse distance weighted (IDW) interpolation. Inverse distance weighted methods are based on the assumption that a required variable should be influenced mostly by the nearby points and less by the most distant points (Douglas, 1996). Therefore, the weight assigned to each point diminishes as the distance from the interpolation point (of the required variable) to a relevant scatter or receptor point increases (Douglas, 1996).

In order to take advantage of the dispersion model to be developed, accurate information regarding wind speed and direction (wind vector) at a receptor point is required. The inverse distance weighted interpolation will be used to interpolate between measuring points (meteorological stations) to determine the wind vector at a receptor point. The IDW interpolation scheme is as follows (Douglas, 1996):

$$v_{(x,y)} = \sum_{n=1}^{N} \frac{u_{(x,y)}}{r_n^{\,2}} \Bigg/ \sum_{n=1}^{N} \frac{1}{r_n^{\,2}} \qquad [2.6]$$

where $v_{(x,y)}$ = wind vector at a receptor point,
$u_{(x,y)}$ = wind vector at a measuring point,
N = number of stations considered in the interpolation, and
r_n = horisontal distance from the receptor point (x_r ,y_r) to the nth measuring point and is given by:

$$r_n = \sqrt{(x_r - x_n)^2 + (y_r - y_n)^2} \qquad [2.7]$$

where x_r = x coordinates of a receptor in km,
y_r = y coordinates of a receptor in km,
x_n = x coordinates of a measuring point in km, and
y_n = y coordinates of a measuring point in km.

The x and y components of the wind vector at a receptor point (v_x and v_y) are calculated using Equation 2.6, which are then used in calculating the wind speed and direction at a receptor point. Wind speed at a receptor point is given by:

$$r = \sqrt{v_x^{\,2} + v_y^{\,2}} \qquad [2.8]$$

where r = wind speed at a receptor point in m/s,
v_x = x component of the wind vector at a receptor point, and
v_y = y component of the wind vector at a receptor point.

The wind angle α used in calculating the wind direction at a receptor point is given by:

$$\alpha = \arctan\left[\frac{v_y}{v_x}\right] \qquad [2.9]$$

where α = wind angle in degrees,

v_x = x component of the wind vector at a receptor point, and

v_y = y component of the wind vector at a receptor point.

The equation used in calculating the wind direction (θ) from the wind angle (α) depends on the quadrant from which the wind is blowing. Figure 2.2 shows the relevant quadrants and the corresponding equation. The interpolated wind direction is used in calculating the downwind and crosswind distances.

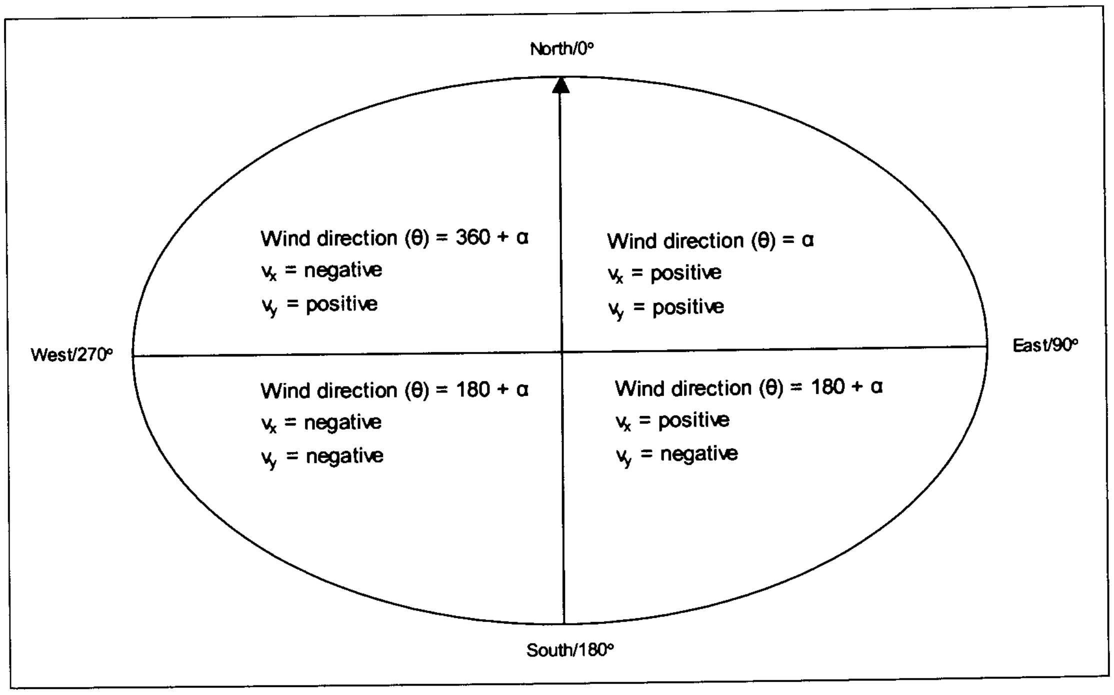


Figure 2.2 Wind direction calculation for each quadrant.

The downwind distance (x) is positive to the east of the user defined origin and the crosswind distance (y) is positive to the north. If x and y coordinates of the source are x(s) and y(s), the downwind distance (x) to the receptor, along the direction of the plume is given by (USEPA, 1995):

$$x = -[x(r) - x(s)]\sin\theta - [y(r) - y(s)]\cos\theta \qquad [2.10]$$

where θ = direction from which the wind is blowing in degrees,
x(s) = x coordinates of the source in km,
y(s) = y coordinates of the source in km,
x(r) = x coordinates of the receptor in km, and
y(r) = y coordinates of the receptor in km.

The downwind distance (x) is used in calculating the distance dependent plume rise and the dispersion parameters. The crosswind distance (y) or horisontal distance to a receptor point from the plume centreline is given by (USEPA, 1995):

$$y = [x(r) - x(s)]\cos\theta - [y(r) - y(s)]\sin\theta \qquad [2.11]$$

2.8 PLUME RISE

The calculation of plume rise is an important element in dispersion modeling. Except in strong winds, plume rise will increase the effective stack height (H) between 2 to 10 times the physical stack height (h), for a typical elevated buoyant source (Ku, 1984). The characteristics of the stack effluent at stack outlet and those of the ambient atmosphere determine the formation and rise of a plume in the atmosphere. A plume of hot gases emitted vertically has both momentum and buoyancy contributing to its rise. As the plume moves away from the stack, it looses its vertical momentum and as the vertical momentum declines, the plume bends over in the direction of the mean wind speed (Cooper and Alley, 1994). Plume rise is given by the elevation of the plume centreline above the stack outlet, as a function of distance downwind of the stack. The plume rise for the different stability classes is calculated as described in Sections 2.8.1 and 2.8.2.

2.8.1 Neutral or Unstable Buoyancy Rise

The plume rise, as a function of downwind distance, for neutral-unstable atmospheres (Classes A, B, C, and D) is calculated by (Cooper and Alley, 1994):

$$\Delta h = \frac{1{,}6F_b^{1/3}}{u} x^{2/3} \quad \text{for } x < x_f \qquad [2.12]$$

$$\Delta h = \frac{1{,}6F_b^{1/3}}{u} x_f^{2/3} \quad \text{for } x >= x_f \qquad [2.13]$$

where Δh = plume rise,
F_b = initial buoyancy flux in m^4/s^3,
u = wind speed at stack height in m/s, and
x_f = distance to final rise in m and is given by:

$$x_f = 3{,}5x^* \qquad [2.14]$$

where x^* = distance beyond which entrainment of ambient air by the plume is dominated by ambient turbulence instead of plume turbulence and is given by (Cooper and Alley, 1994):

$$x^* = 34F_b^{2/5} \quad \text{for } F_b >= 55 \qquad [2.15]$$

$$x^* = 14F_b^{5/8} \quad \text{for } F_b < 55 \qquad [2.16]$$

The buoyancy flux term F_b is given by (Ku, 1984):

$$F_b = w_o R_o^2 \frac{g}{T_{po}}(T_{po} - T_a) \qquad [2.17]$$

where F_b = initial buoyancy flux in m^4/s^3,
w_o = initial plume speed in m/s,

R_o = inside stack radius in m,

T_{po} = initial plume temperature in degrees K,

T_a = ambient temperature at stack height in degrees K, and

g = gravitational constant, 9,8 m/s^2.

2.8.2 Stable atmospheres (Class E and F)

The plume rise for stable atmospheres is calculated by (Bøhler, 1987):

$$\Delta h = 2{,}6\left[\frac{F_b}{u\ S}\right]^{1/3} \qquad [2.18]$$

where S = the stability parameter and is given by (Cooper and Alley, 1994):

$$S = \frac{g}{T_a}\left(\frac{\Delta T}{\Delta z}\right) \qquad [2.19]$$

where g = gravitational constant, 9,8 m/s^2,

T_a = ambient temperature at stack height in degrees K, and

$\Delta T / \Delta z$ = potential temperature gradient.

If data for the potential temperature gradient is not available; 0,02 K/m is used for class E and 0,035 K/m for class F (Cooper and Alley, 1994). The distance to final rise (x_f) for stable conditions is given by (USEPA, 1995):

$$x_f = 2{,}1715\frac{u}{\sqrt{S}} \qquad [2.20]$$

If the distance upwind from the receptor to the source is less than the distance to final rise ($x < x_f$), then Equation 2.12 is used to determine the plume rise for stable conditions. This plume rise is used only for buoyancy dominated conditions; should it exceed the plume rise at x_f, then the plume rise at x_f is used instead (USEPA, 1995).

2.9 PLUME PENETRATION INTO ELEVATED STABLE LAYER

Quite frequently the lower atmosphere consists of a well-mixed, nearly adiabatic layer adjacent to the ground, capped by a thick stable layer (inversion). A buoyant plume rising into such a layer may partially or completely penetrate the elevated stable layer. This will happen if the effective height is greater than the height of the inversion layer. The fraction (p) of the plume that penetrates the elevated stable layer is computed from Briggs criteria for bent over plumes (Ku, 1984);

$$p = \frac{1{,}5\Delta h - z_i}{1{,}5\Delta h - 0{,}5\Delta h} \qquad [2.21]$$

where p = penetration fraction,
Δh = plume rise in m, and
z_i = z_m (mixing height) - h (stack height) in m.

The mixing height is the distance above the ground to which relatively unrestricted vertical mixing occurs in the atmosphere. The mixing height is calculated as (USEPA, 1992):

$$z_m = 320u$$

and effective stack height under these conditions is given by (Ku, 1984):

$$H = h + (2/3 + 1/3\,p)z_i \qquad [2.22]$$

where H = effective stack height in m.

If p = 1 then the entire plume penetrated the stable layer and no groundlevel concentrations occurs. It can then be assumed that Q (for Equation 2.1) is equal to Q_s, which equals zero; where Q_s is the emission rate remaining in the mixed layer.

If $p = 0$ the plume remains within the mixed layer and $Q_s = Q$. Where $0 < p < 1$, the plume partially penetrated the stable layer and the effective stack height is calculated by Equation 2.22 and $Q_s = Q\ (1 - p)$ (Ku, 1984).

2.10 BCL LIMITED BACKGROUND

BCL Limited is situated in Selebi-Phikwe, Botswana. The company extracts nickel and copper ore from three underground mines in the area, producing a nickel/copper concentrate. The concentrate is dried and then smelted, resulting in different mattes (granulated mixtures of nickel, copper, iron and some heavy metals). These mattes are exported for separation and refining of the relevant metals prior to final sale (BCL Limited, 1982).

The overall material flow for the BCL Limited plant is shown in Figure 2.3. Ore from the three shafts of Selebi, Selebi-North and Phikwe mine is transported by rail up to the rail ore bin where the processing plant begins. Ore from the bin is sequentially crushed in the primary, secondary and tertiary crushers. Oversized ore is conveyed to the next crusher and all undersized ore from the crushers to the mills. Oversized ore from the tertiary crusher is returned to the ore stockpile to undergo tertiary crushing again.

At the mills the ore is crushed into fine material and conveyed to the flotation circuit. The mineral is concentrated during the froth flotation process when collector and inter-froth reagents are added. The froth from this process will contain the concentrate and the underflow is tailings (mining waste). The tailings is further concentrated or thickened and part of it is pumped directly to the top of the tailings dam, while the remainder is pumped underground for use as backfill. The concentrate is fed to the smelter (BCL Limited, 1982).

At the smelter the concentrate is first dried using pulverised fuel (PF) coal from the coal plant. Additional concentrate is added from Tati mine in the northeastern part of Botswana. The dry concentrate is conveyed to the flash furnace where it is smelted in an oxygen enriched atmosphere with a suitable flux to separate the matte from impurities. Silica used as flux in the flash furnace is provided from sand harvested at the nearby Motloutse river. The sand is screened to remove coarse material and dried before being conveyed to the flash furnace. The liquid matte collects in the lower part of the furnace and is removed and transferred to the converters (BCL Limited, 1982).

At the converters, the matte is further upgraded with oxygen enriched air to remove the iron as slag and part of the sulphur as sulphur dioxide (SO_2), producing a low sulphur nickel copper matte. The slag from the converters and flash furnace is further processed at the electric furnace to recover more matte. Some reverts (scrap metals) are also smelted in the electric furnace. Finally, the slag is granulated and then transported by trucks to the slag dump (BCL Limited, 1982).

Waste gases from the flash furnace go through the waste heat boiler and the precipitator and are then emitted though the main stack (BCL Limited, 1982). All gases from the converters are ducted, using converter hoods, straight to the main stack. The small (73 m) stack is used for the drying plant but for this research only the main stack is considered. In total there are four main contributors to the gases emitted from BCL Limited, namely the flash furnace, converters, electric furnace and the converter slag cleaning vessel. Material balances, based on 2002 production figures (Hagger, 2003), across these processes are presented in Sections 2.10.1 to 2.10.4.

2.10.1 Flash Furnace

The overall material balance for the flash furnace is shown in Figure 2.4. The flash furnace processes an average of 850 000 tonnes of concentrate per annum from the drying plant to produce 870 000 tonnes of slag and 55 000 tonnes of matte. The other inflows to the flash furnace are 66 000 tonnes of PF coal, 12 600 tonnes of lump coal, 230 000 tonnes of silica flux and 2 681 400 tonnes of reaction air with oxygen. The flow of the waste gases from the flash furnace is 2 915 000 tonnes per annum with a sulphur content of 5,9%.

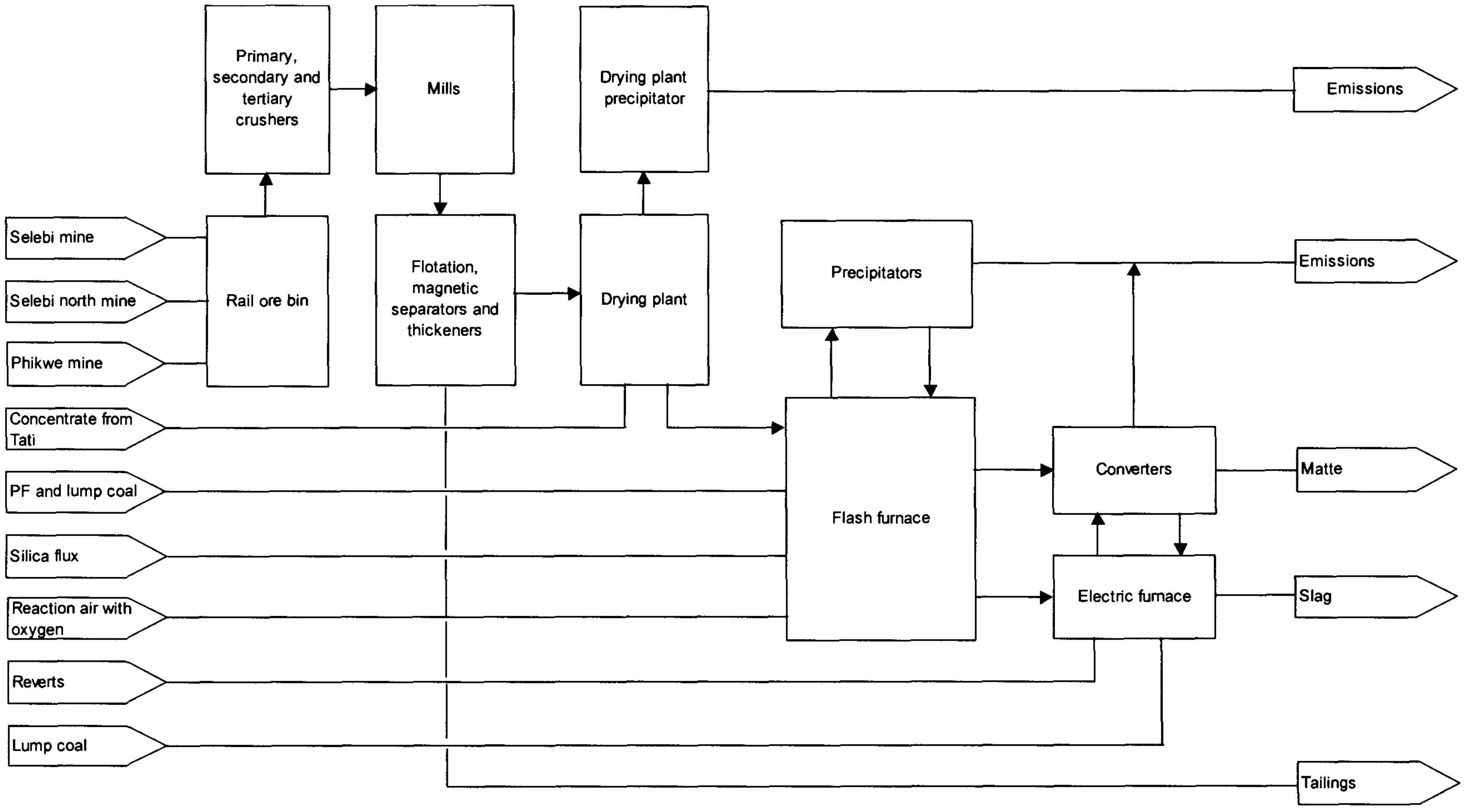
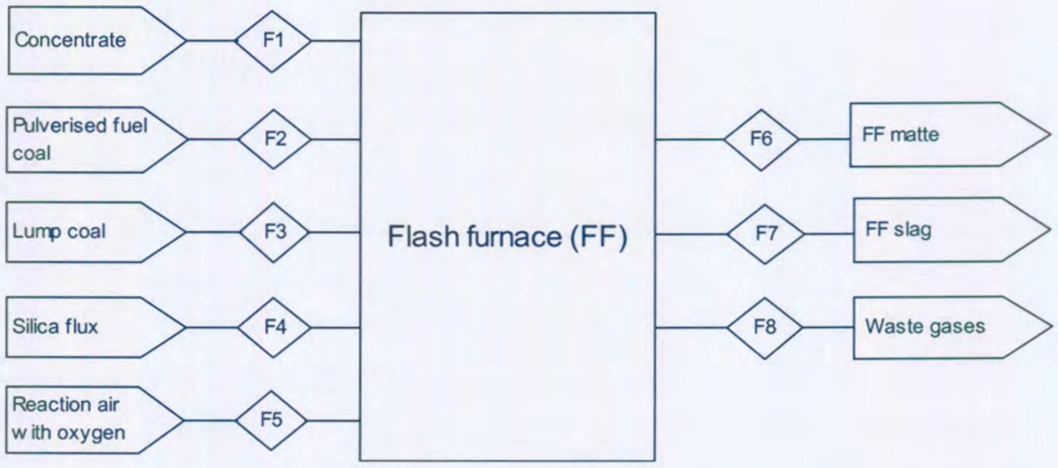


Figure 2.3 Overall material balance for BCL Limited plant.



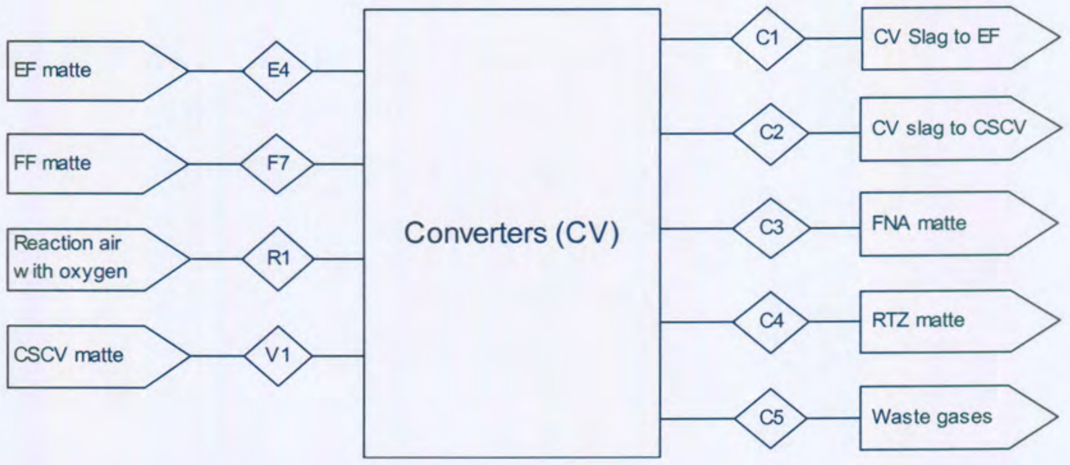
Stream	Description	Tonnes per annum*	Composition*
F1	Concentrate	850 000	3,5% Ni, 3,2% Cu
F2	Pulverised fuel coal	66 000	
F3	Lump coal	12 600	
F4	Silica flux	230 000	78% SiO_2
F5	Reaction air with oxygen	2 681 400	
F6	Matte	55 000	
F7	Slag	870 000	
F8	Waste gases to stack	2 915 000	5,9% S

* data from BCL Limited production, 2002.

Figure 2.4 Overall balance for BCL Limited flash furnace.

2.10.2 Converters

In Figure 2.5 the overall material balance for the converters is shown. The matte from the flash furnace (F7), electric furnace (E4) and the converter slag cleaning vessel (V1) are transferred to the converters to reduce its sulphur contents with oxygen enriched air (R1). The converters produce 33 000 tonnes per annum of FNA matte (matte produced for the Norwegian market with 22% sulphur content) and 22 500 tonnes per annum of RTZ matte (matte produced for the Zimbabwean market with 6% sulphur content).



Stream	Description	Tonnes per annum*	Composition*
E4	Matte from electric furnace	64 000	
F7	Matte from flash furnace	55 000	
R1	Reaction air with oxygen	3 662 500	
V1	CSCV matte	16 000	
C1	Converter slag to electric furnace	28 000	
C2	Converter slag to CSCV	100 000	
C3	FNA matte	33 000	22% S
C4	RTZ matte	22 500	6% S
C5	Waste gases	3 614 000	1,5% S

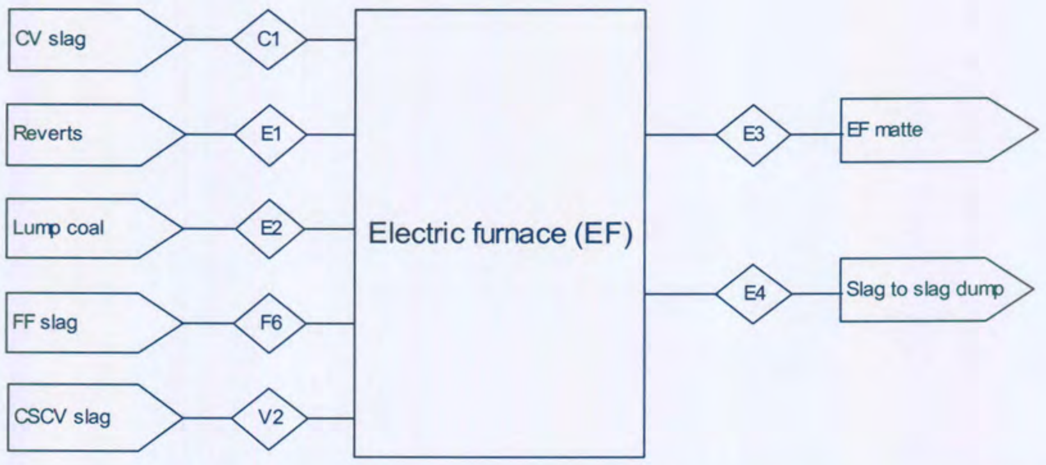
* data from BCL Limited production, 2002.

Figure 2.5 Overall balance for BCL Limited Converter.

Slag from the converters is split between the electric furnace (C1) and the converter slag cleaning vessel (C2), to recover more matte from the slag. The flow of waste gases from the converters is 3 614 000 tonnes per annum with a sulphur content of 1,5%.

2.10.3 Electric Furnace

The overall material balance for the electric furnace is shown in Figure 2.6. The electric furnace recovers matte in the slag from the flash furnace (F6), converters (C1) and the converter slag cleaning vessel (V2) using lump coal (E2) and reverts (E1). The outputs from the electric furnace are 64 000 tonnes per annum of matte that is transferred to the converters, and 954 600 tonnes per annum of granulated slag that is transported by truck to the slag dump.



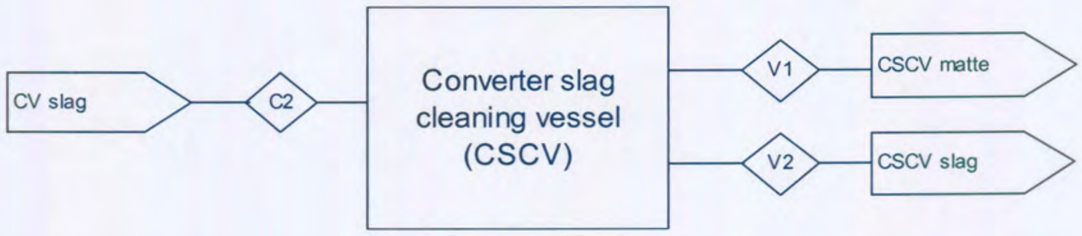
Stream	Description	Tonnes per annum*
C1	Converter slag	28 000
E1	Reverts	25 000
E2	Lump coal	11 600
F6	Slag from flash furnace	870 000
V2	Slag from CSCV	84 000
E3	Electric furnace matte	64 000
E4	Slag to slag dump	954 600

* data from BCL Limited production, 2002.

Figure 2.6 Overall balance for BCL Limited electric furnace.

2.10.4 Converter Slag Cleaning Vessel

In Figure 2.7 the overall material balance for the converter slag cleaning vessel is shown. 100 000 tonnes per annum of slag is transferred from the converter to the converter slag cleaning vessel for further processing. The CSCV produce 16 000 tonnes per annum of matte that is returned to the converters. The slag from the cleaning vessel is transferred to the electric furnace for further processing.



Stream	Description	Tonnes per annum*
C2	Converter slag to CSCV	100 000
V1	CSCV matte	16 000
V2	CSCV slag	84 000

* data from BCL Limited production, 2002.

Figure 2.7 Overall balance for BCL Limited converter slag cleaning vessel.

2.10.5 Sulphur Balance

A total sulphur balance for the four processes illustrated in Figures 2.4 – 2.7 is given in Table 2.4 and 2.5. The smelter processes about 850 000 tonnes per annum of concentrate, using 66 000 tonnes of pulverised coal and 24 200 tonnes of lump coal. Total sulphur input to the four smelter processes is 247 673 tonnes. Total production from the smelter is 22 500 tonnes per annum of RTZ matte and 33 000 tonnes per annum of FNA matte.

Table 2.4 Sulphur in the inputs.

Process stream	Tonnes per annum	% Sulphur	Tonnes per annum S
Concentrate	850 000	29,0	246 500
Pulverised fuel coal	66 000	1,3	858
Lump coal	24 200	1,3	315
Total			247 673

Table 2.5 Sulphur in the outputs.

Process stream	Tonnes per annum	% S	Tonnes per annum S	Tonnes per annum SO_2
Flash furnace off-gas	2 915 000	5,90	171 985	343 970
Converter off-gas	3 614 000	1,50	54 210	108 420
RTZ matte	22 500	6,00	1 350	
FNA matte	33 000	22,00	7 260	
Slag	954 600	±1,35	12 868	
Total			247 673	452 390

CHAPTER 3

Model Description

3.1 INTRODUCTION

Physical scientist and engineers have developed plume dispersion models in an attempt to predict the concentration of air pollutants in the vicinity of point sources of pollution. The most common, widely studied and validated are the Gaussian plume dispersion models (Scott *et. al.*, 2003). Varying in sophistication, these models predict the concentration at downwind distances from industrial sources of pollution by relying on the following:

- information about the source itself (for example, stack height, exit velocity and exit temperature) to estimate the height of the plume; and

- atmospheric conditions (including mixing height, wind speed and direction and atmospheric stability) and terrain to estimate dispersion.

Selebi-Phikwe is located in the vicinity of one such point source of pollution. The major pollutants emitted during the production of nickel/copper at BCL Limited are sulphur compounds, trace elements and particulates. Hot gases leave the smelter stack at high velocity providing both buoyancy and lift, thereby allowing for extended period of dispersion within the atmospheric mixing layer before pollutants reach groundlevel (Scott *et. al.*, 2003). Measurement stations are in place to determine the impact of the plume at fixed locations. However, with changing wind direction measurement stations needs to be coupled with a model to determine the impact at locations in Selebi-Phikwe with no measurement stations.

The algorithms used in developing the BCL Limited model (EEGAIR-BCL1) is described in Section 3.2, using Equations 2.1 to 2.22 developed in Chapter 2.

Requirements for the various inputs to the program are stipulated in Section 3.3 and the resultant outputs given in Section 3.4.

3.2 MODEL ALGORITHMS

The model is developed to calculate or predict groundlevel sulphur dioxide concentrations from the BCL Limited smelter stack at specified receptor points. It accepts hourly meteorological data records to define the conditions for plume rise, dispersion and transport. The model uses the steady state Gaussian plume equation that models the dispersion of a non-reactive gaseous pollutant from an elevated source. Computations can be made for up to 500 receptor points. In practical applications, the number of receptor points should be kept low to avoid excessive run time.

A flowdiagram of the BCL Limited dispersion model is depicted in Figure 3.1. The model is written in the Delphi programming language and is divided into several algorithms that are combined into the main program. The eleven algorithms used in the program are as follows:

- Algorithm 1 - initialising model constants,
- Algorithm 2 - determining the receptor points,
- Algorithm 3 - determining the atmospheric stability,
- Algorithm 4 - determining the smelter stack characteristics,
- Algorithm 5 - determining the wind speed at stack height,
- Algorithm 6 - determining the wind speed and direction at each receptor points,
- Algorithm 7 - determining the crosswind and downwind distances,
- Algorithm 8 - determining the dispersion parameters,
- Algorithm 9 - determining the plume rise,
- Algorithm 10 - determining the plume penetration, and
- Algorithm 11 - determining the concentration at each receptor point.

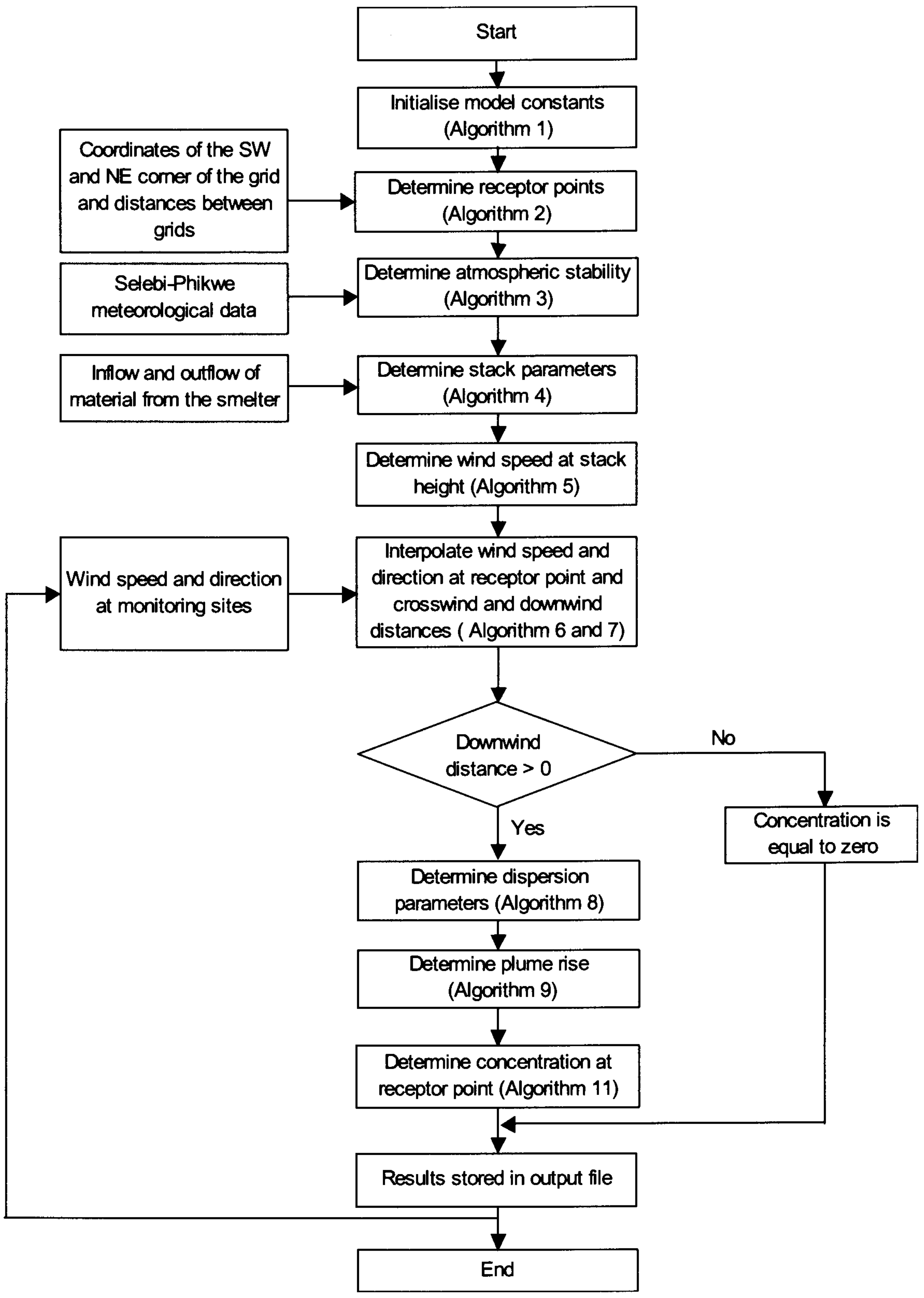

Start
Initialise model constants (Algorithm 1)
Coordinates of the SW and NE corner of the grid and distances between grids
Determine receptor points (Algorithm 2)
Selebi-Phikwe meteorological data
Determine atmospheric stability (Algorithm 3)
Inflow and outflow of material from the smelter
Determine stack parameters (Algorithm 4)
Determine wind speed at stack height (Algorithm 5)
Wind speed and direction at monitoring sites
Interpolate wind speed and direction at receptor point and crosswind and downwind distances (Algorithm 6 and 7)
Downwind distance > 0
No
Concentration is equal to zero
Yes
Determine dispersion parameters (Algorithm 8)
Determine plume rise (Algorithm 9)
Determine concentration at receptor point (Algorithm 11)
Results stored in output file
End

Figure 3.1 BCL Limited model.

Algorithm 1: Initialising model constants

At the start of the program, all constant parameters used in the model are initialised. The initialised constants are for stability related power law exponents (Table 2.2) and dispersion coefficients (Table 2.3). This algorithm is only executed at the beginning of the program.

Algorithm 2: Determining the receptor points

The EEGAIR-BCL1 program can determine concentrations at receptors (coordinate points in a defined grid) for a distance of 10 km from the stack in the east, south, west and north directions respectively. In defining the grid, the algorithm has two options for determining the coordinates of each receptor point, which are as follows:

- using receptor points generated by the model, and
- using receptor points defined by the user.

Using the receptor points generated by the model option, the solution to this procedure is as follows:

(1) The algorithm defines predetermined x, y and z coordinates of the southwest and northeast corners of the modelling grid (based on the Selebi-Phikwe area).

(2) It then defines the distance between grid points as 1 km intervals.

(3) The area between the southwest and the northeast corner is divided into grid squares of the same length as the distance between grid points (1 km). Receptor points are placed at each corner of the grid squares and given a reference number; and the smelter stack is located at the origin (Figure 3.2).

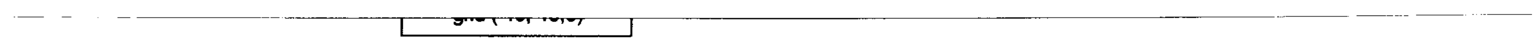


Figure 3.2 Illustration of grid points specifications.

When the user defined receptor points option is used, the solution procedure is as follows:

(1) The number of receptor points to be used by the model is entered.

(2) The x, y and z coordinates of each receptor point is manually entered.

(3) The receptor point coordinates are stored in a three-dimensional array and are generated for each meteorological period calculation.

At the beginning of the program, the user chooses which option to use. The model can handle up to 500 receptor points but should be kept to a minimum to avoid excessive run time.

Algorithm 3: Determining the atmospheric stability

The diffusion of air pollutants in the lower atmosphere is strongly influenced by the local atmospheric stability. The meteorological data from Selebi-Phikwe is required to determine the stability class for each meteorological period. The solution procedure is as follows:

(1) For daytime, determine the stability class from known values of wind speed and solar radiation using Table 2.1.

(2) For nighttime, determine the stability class from known values of wind speed and vertical temperature gradient using Table 2.1.

The class determined now constitutes the Pasquill-Gifford stability class for the meteorological period. It depends on the meteorology prevailing for each hour and will change with changing meteorological periods.

Algorithm 4: Determining the smelter stack characteristics

In order to estimate the impact of a stationary source like BCL Limited's smelter stack on air quality, certain characteristics of the source must be known. The following information must be available:

- pollutant emission rate,
- stack height,

- stack gas temperature, stack inside diameter and stack gas exit velocity, and
- location of the point of emission.

(A) Pollutant emission rate

The material balance for the smelter is required to determine emission rate of sulphur dioxide from the smelter stack. The solution procedure is as follows:

(1) Determine the amount of sulphur in the smelter inflow using Table 2.4.

(2) Determine the amount of sulphur in the smelter outflow using Table 2.5.

(3) Determine a calculated value of sulphur dioxide emitted from known values of sulphur in the inflow and outflow.

The value determined in step A now constitutes the emission rate of sulphur dioxide. It depends on the inflow of the concentrate to the smelter and the coal used and is calculated at the beginning of the program.

(B) Stack gas exit velocity

The mass flow of flue gas or waste gases from the smelter is required to determine the gas exit velocity in the stack. The solution procedure is as follows:

(1) Determine the mass flow of the flue gas using Table 2.5.

(2) Determine the flue gas density from known values of the stack gas temperature. This assumes that the density of stack gas is close to that of air (SO_2 represents approximately 3,46% (m/m) of the waste gases).

(3) Determine the gas volumetric flow rate in the stack as the ratio of mass flow rate and gas density.

(4) Determine the area of the stack from known value of the stack diameter.

(5) Determine the stack gas velocity as the ratio of stack gas volumetric flow rate and stack area.

The values determined in step B now constitute the exit gas velocity from the main stack. Exit gas velocity depends on the flow of the flue gas from the furnace and the converters and is calculated at the beginning of the program.

Other stack parameters mentioned above are entered as constant values at the beginning of the program.

Algorithm 5: Determining the wind speed at stack height or release height

The mean transport wind speed is an important variable within the Gaussian plume equation as it is inversely proportional to the modelled concentration. Wind speed measured at Selebi-Phikwe monitoring stations is required to determine the wind speed at stack height. The solution procedure is as follows:

(1) Obtain the stability related power law exponent value, based on the stability class determined in Algorithm 3, using Table 2.2.

(2) Determine wind speed at release height from known values of anemometer height, wind speed at anemometer height, stability related power law exponent and stack height using Equation 2.3. For the case where the calculated value of wind speed at stack height is less than 1 m/s, wind speed is set equal to 1 m/s (USEPA, 1995).

The value determined now constitutes the wind speed at stack height. Wind speed at stack height is dependent on the stability and wind speed at anemometer height. Therefore, it is determined for every change in meteorological period.

Algorithm 6: Determining the wind speed and direction at each receptor point

One of the most commonly used techniques for interpolation of scatter points is the inverse distance weighted interpolation. The wind speed and direction recorded at the three meteorological stations in Selebi-Phikwe is required to interpolate wind speed and direction at each receptor point. The solution procedure is as follows:

(1) Determine the x and y components of the velocity vector (wind speed and direction) at each meteorological station.

(2) Determine the distance between the receptor point and each station from known station and receptor coordinates using Equation 2.7.

(3) Determine x and y components of the velocity vector at the receptor point from known values of x and y component of the velocity vector at each station using Equation 2.6.

(4) Determine the wind speed at the receptor point from known values of x and y component of the velocity vector at the receptor point using Equation 2.8.

(5) Determine the wind direction at each receptor point from known values of x and y components of the velocity vector at the receptor point using Equation 2.9.

The values determined now constitute the wind speed and direction at a receptor point. Wind speed and direction at each receptor point depends on the wind speed and direction at the monitoring station and the distance between measuring stations and the relevant receptor point. Therefore, they will change with changes in receptor points and meteorology.

The interpolation algorithm can only be used for interpolating meteorological data from the three stations in Selebi-Phikwe.

Algorithm 7: Determining the crosswind and downwind distances

The source coordinates, receptor point coordinates and wind direction at receptor points are required to determine the downwind and crosswind distances. The solution procedure is as follows:

(1) Determine the downwind distance (x) to a receptor point along the direction of the plume from known values of wind direction, source coordinates and receptor point coordinates, using Equation 2.10.

(2) Determine the crosswind distance (y) to a receptor point from the plume centreline from known values of wind direction, source coordinates and receptor point coordinates, using Equation 2.11.

The x and y values determined now constitutes the downwind and crosswind distances respectively. The downwind and crosswind distances depend on the wind direction and receptor point coordinates. Therefore, they should be determined for each change in receptor point and meteorological period.

Algorithm 8: Determining the dispersion parameters

The rate of dispersion in the atmosphere is dependent on the atmospheric stability. The dispersion parameters (σ_y and σ_z) are the standard deviations of the concentration distribution in the y and z directions, respectively.

(A) The solution procedure for σ_y is as follows:

(1) Determine the constants a and b from known stability using Table 2.3.

(2) Determine σ_y from the constants a, b and downwind distance (x) using Equation 2.4.

The value determined in step A now constitutes the dispersion parameter (σ_y) of the plume concentration. It depends on the stability class and downwind distance (x). The dispersion parameter is calculated for every change in the meteorological period and downwind distance.

(B) The solution procedure for σ_z is as follows:

(1) Determine the constants c, d and f from known stability and downwind distance (x) using Table 2.3.

(2) Determine σ_z from the constants c, d and f and downwind distance (x) using Equation 2.5.

(3) The calculated value of σ_z is set equal to 5 000 m if its greater than 5 000 m (USEPA, 1995).

The value determined in step B now constitutes the dispersion parameter (σ_z) of the plume concentration in the z direction. The dispersion parameters depend on the stability class and downwind distance (x) along the direction of the plume. Therefore, it is calculated for every change in meteorological period and downwind distance.

Algorithm 9: Determining the plume rise

In dispersion modeling, plume rise is used to determine the effective stack height. The characteristics of the flue gas and ambient conditions are required to determine plume rise. The required stack characteristics are exit gas velocity, stack diameter and exit gas temperature while the required ambient conditions are wind speed and ambient temperature at stack height.

In calculating plume rise, the initial stage is to determine the buoyancy flux from known values for the exit gas velocity, stack diameter, exit gas temperature and ambient temperature using Equation 2.17. The plume rise is then determined for either neutral/unstable conditions or stable conditions.

The solution procedure for unstable and neutral conditions is as follows:

(1) For buoyancy flux greater than 55, determine the distance (x^*) beyond which entrainment of ambient air by the plume is dominated by ambient turbulence instead of plume turbulence, from known value of the buoyancy flux using Equation 2.15.

(2) For buoyancy flux less than 55, determine x^* from known values of the buoyancy flux term using Equation 2.16.

(3) Determine the distance to final rise (x_f) from known value of x^* using Equation 2.14.

(4) For crosswind distances less than the distance to final rise, determine plume rise (Δh) from known values of buoyancy flux, wind speed at stack height and downwind distances using Equation 2.12.

(5) For crosswind distance greater than the distance to final rise, determine plume rise (Δh) from known values of buoyancy flux, wind speed at stack height and distance to final rise using Equation 2.13.

(6) The calculated value of plume rise in step (5) is compared with the value from step (4). If the value in step (5) is less than that in step (4), the plume rise for x less than x_f is set equal to the value in step (5).

The solution procedure for stable conditions is as follows:

(1) Determine the stability parameter (S) from known values of potential temperature gradient and air temperature using Equation 2.19.

(2) Determine the distance to final rise (x_f) from known values of wind speed at stack height and stability parameter using Equation 2.20.

(3) For crosswind distance less than the distance to final rise, determine plume rise (Δh) from known values of buoyancy flux, wind speed at stack height and downwind distance using Equation 2.12.

(4) For crosswind distance greater than the distance to final rise, determine plume rise (Δh) from know values of buoyancy flux, wind speed at stack height and stability parameter using Equation 2.18.

(5) The calculated value of plume rise in step (4) is compared with the value from step (3). If the value in step (4) is less than that in step (3), the plume rise for x less than x_f is set equal to the value in step (4).

The value determined now constitutes plume rise during the meteorological period. It depends on the characteristics of the stack and the meteorology of the hour. This implied that plume rise changes with every change in meteorology.

Algorithm 10: Determining the plume penetration

Plumes from tall stacks may frequently interact with the capping inversion at the top of the mixed layer. A fraction of the plume mass may penetrate the inversion, and therefore be unavailable for immediate mixing to the ground. The inversion height, plume rise and stack height are required to determine the fraction of the plume that penetrates the elevated stable layer and the plume material remaining in the mixing layer. The solution procedure is as follows:

(1) Determine the height of the inversion from know values of the wind speed at stack height. The mixing height is set as 320 x u_{10} for unstable and neutral conditions.

(2) Determine the fraction of the plume that penetrates the elevated stable layer from known values of the plume rise, inversion height and the stack height using Equation 2.21.

(3) Determine the effective source height from known values of stack height and fraction of the plume that penetrated using Equation 2.22.

(4) For total penetration, set the source strength remaining in the mixed layer to zero.

(5) For no penetration, the source strength remaining in the mixed layer is the same as the pollutant emission rate.

(6) For partial penetration, determine the source strength from known values of the pollutant emission rate and the fraction of the plume that penetrated the elevated stable layer.

The values determined now constitute the source strength remaining in the mixed layer and the effective stack height. In cases where the effective stack height exceeds the mixing height, the plume is assumed to fully penetrate the elevated stable layer and the groundlevel concentration is set equal to zero.

Algorithm 11: Determining the concentration at each receptor point

The hourly groundlevel concentration at downwind distance (x) and crosswind distance (y) is calculated using the Gaussian equation. This is the main algorithm that combines other algorithms to determine the concentration at each receptor point. The parameters required are:

- source emission rate determined by Algorithm 4,
- wind speed at stack height determined by Algorithm 5,
- crosswind distance determined by Algorithm 7,
- dispersion parameters determined by Algorithm 8, and
- effective stack height determined by Algorithm 9.

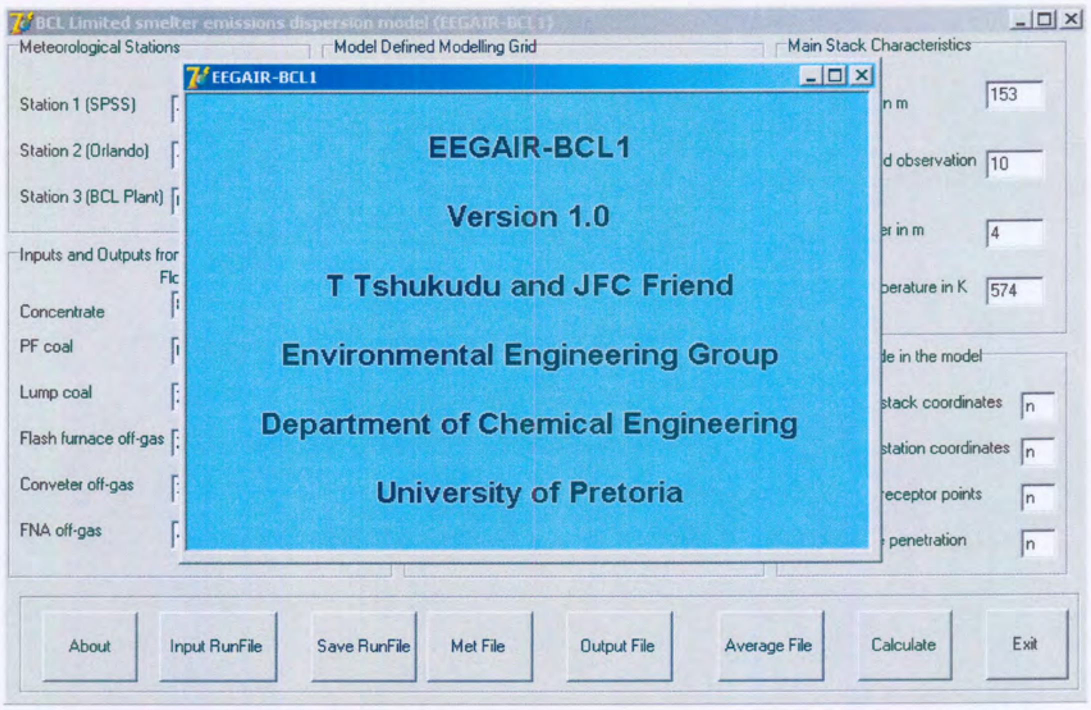
Concentrations at each receptor point are determined from known values of the above using Equation 2.1.

The above algorithms are executed for each receptor point and at the beginning of each meteorological period (except Algorithm 1). The concentration at each receptor for the previous meteorological period are stored in a two-dimensional array of the same size as the number of receptor points and used as the background concentration for the next period. This does not apply for the start period where background concentrations are assumed to be zero.

The source code of the above algorithms, together with the menu system of EEGAIR-BCL1, is attached in Appendix A.

3.3 PROGRAM STARTUP AND INPUT

The EEGAIR-BCL1 model is used to calculate the sulphur dioxide concentration from the smelter stack at specified receptor points. At startup, the program automatically displays Screen 1 for 5 seconds, which is followed by Screen 2.



BCL Limited smelter emissions dispersion model (EEGAIR-BCL1)

Meteorological Stations — Station 1 (SPSS), Station 2 (Orlando), Station 3 (BCL Plant)

Model Defined Modelling Grid

Main Stack Characteristics: 153, 10, 4, 574

Inputs and Outputs fror — Concentrate, PF coal, Lump coal, Flash furnace off-gas, Conveter off-gas, FNA off-gas

EEGAIR-BCL1

EEGAIR-BCL1

Version 1.0

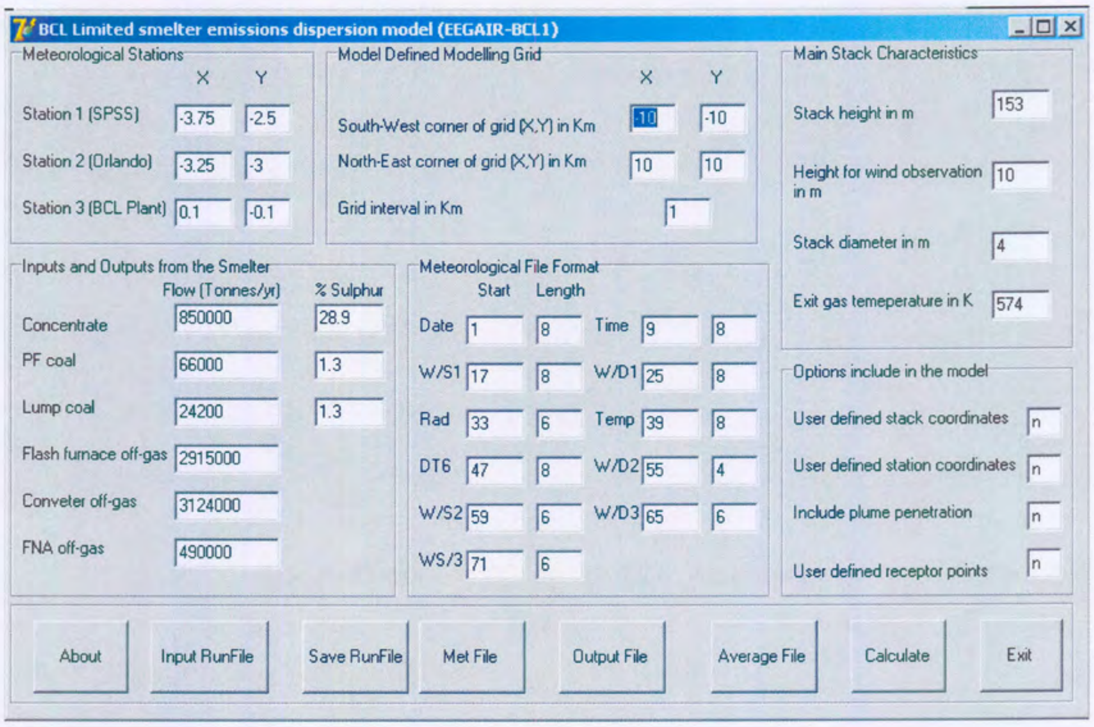
T Tshukudu and JFC Friend

Environmental Engineering Group

Department of Chemical Engineering

University of Pretoria

Screen 1



BCL Limited smelter emissions dispersion model (EEGAIR-BCL1)

Meteorological Stations

	X	Y
Station 1 (SPSS)	-3.75	-2.5
Station 2 (Orlando)	-3.25	-3
Station 3 (BCL Plant)	0.1	-0.1

Model Defined Modelling Grid

	X	Y
South-West corner of grid (X,Y) in Km	-10	-10
North-East corner of grid (X,Y) in Km	10	10
Grid interval in Km	1	

Main Stack Characteristics

Stack height in m	153
Height for wind observation in m	10
Stack diameter in m	4
Exit gas temeperature in K	574

Inputs and Outputs from the Smelter

	Flow (Tonnes/yr)	% Sulphur
Concentrate	850000	28.9
PF coal	66000	1.3
Lump coal	24200	1.3
Flash furnace off-gas	2915000	
Conveter off-gas	3124000	
FNA off-gas	490000	

Meteorological File Format

	Start	Length		Start	Length
Date	1	8	Time	9	8
W/S1	17	8	W/D1	25	8
Rad	33	6	Temp	39	8
DT6	47	8	W/D2	55	4
W/S2	59	6	W/D3	65	6
WS/3	71	6			

Options include in the model

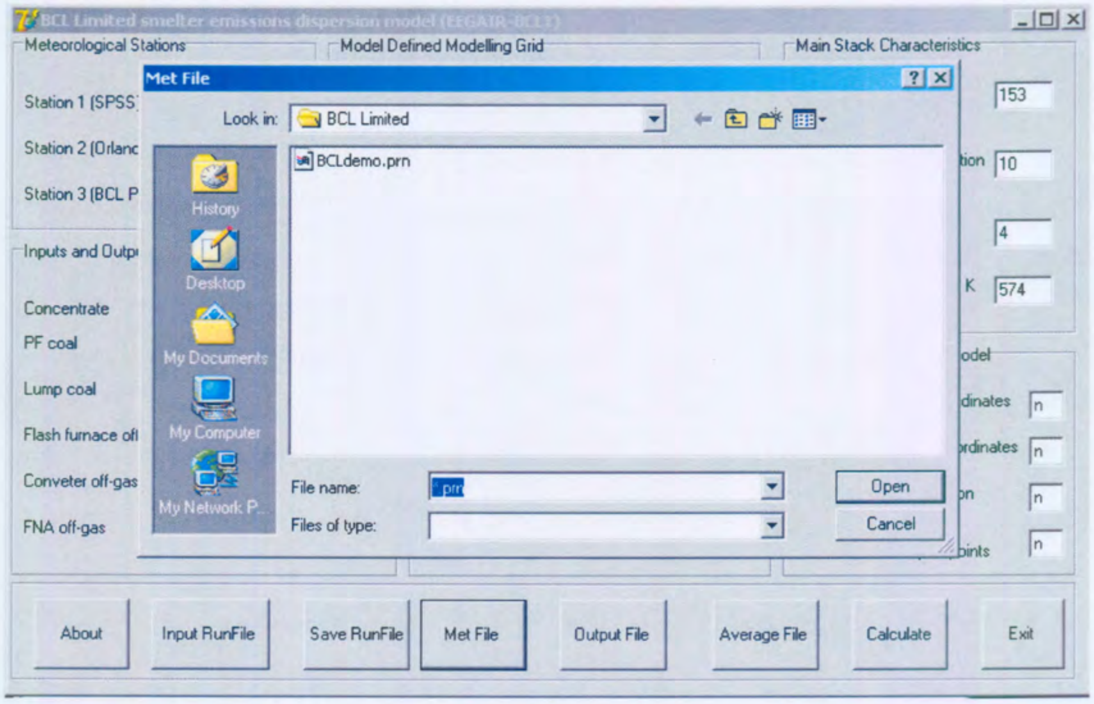
User defined stack coordinates	n
User defined station coordinates	n
Include plume penetration	n
User defined receptor points	n

Screen 2

All data input and calculations are performed from Screen 2. Data input to the program consists of 9 records. Record 1 is used to define both the directory and the filename for the meteorological input file and the files for storing the model output (Screens 3 to 5). Clicking on the **Met File**, **Output File** and **Average File** buttons of Screen 2 respectively, activates Screens 3, 4 and 5. Screen 3 is for defining the meteorological file while Screen 4 is for the output file and Screen 5 for the average file.

Record 1

Variable	Description	Type
Infile	Meteorological input file.	String
OutFile	Results output file.	String
avgFile	Average of results file.	String



Look in: BCL Limited

BCLdemo.prn

File name: *.prn

Files of type:

153

tion 10

4

K 574

odel

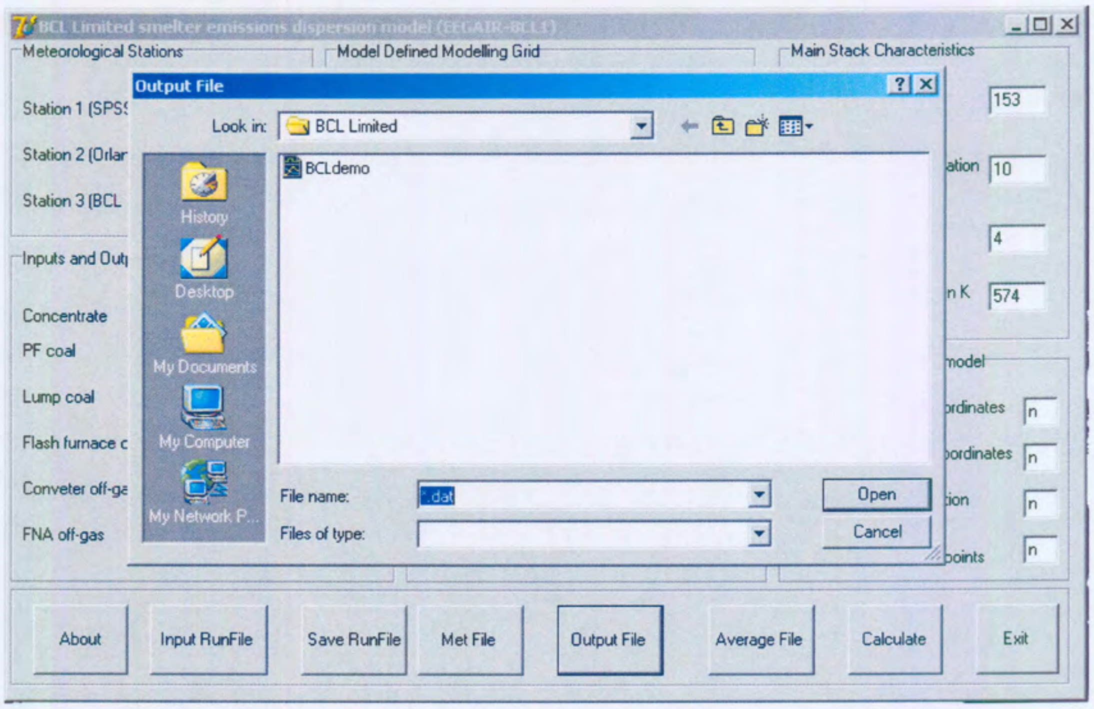
dinates n

rdinates n

on n

oints n

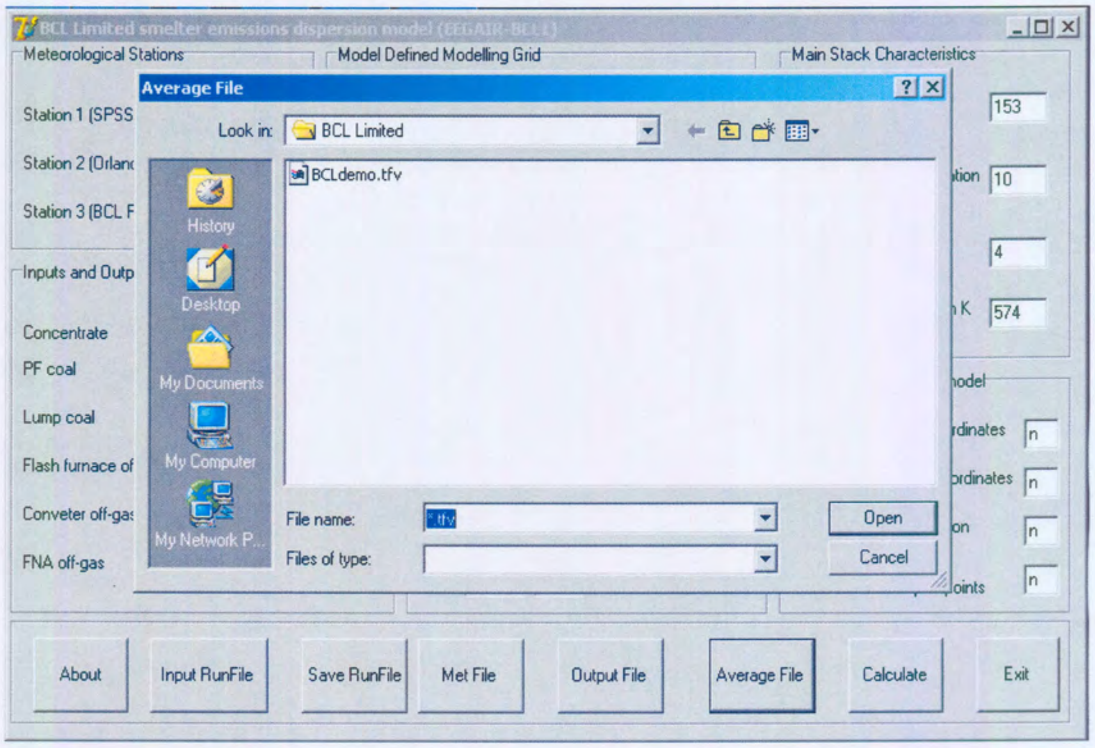
Screen 3



Look in: BCL Limited
BCLdemo
File name: *.dat
Files of type:

153
ation 10
4
n K 574
rdinates n
ordinates n
tion n
points n

Screen 4



Look in: BCL Limited
BCLdemo.tfv
File name: *.tfv
Files of type:

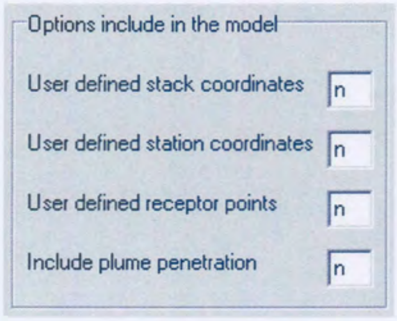
153
tion 10
4
K 574
rdinates n
ordinates n
on n
oints n

Screen 5

Record 2 is used to determine if the options for user defined stack coordinates, meteorological station coordinates and receptor points coordinates are included (Screen 6). It is also used to define if the plume penetration option is included in the calculations.

Record 2

Variable	Description	Type
StkStr	Option for user defined stack coordinates.	Logical
Astn	Option for user defined station coordinates.	Logical
MetString	Option for user defined receptor points.	Logical
PenStr	Option for including plume penetration into an elevated stable layer.	Logical


Options include in the model

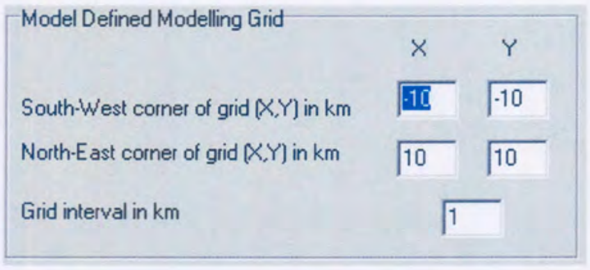
User defined stack coordinates	n
User defined station coordinates	n
User defined receptor points	n
Include plume penetration	n

Screen 6

Record 3 is only used if the option for user defined receptor points is false (MetString = 'n'). It defines the modelling grid and places the receptor points at the corners of each grid square (Screen 7).

Record 3

Variable	Description	Units
Xsw	x coordinates of the southwest corner of the model grid.	km
Ysw	y coordinates of the southwest corner of the model grid.	km
Xne	x coordinates of the northeast corner of the model grid.	km
Yne	y coordinates of the northeast corner of the model grid.	km
GridInt	Distance between each grid point.	km


Model Defined Modelling Grid

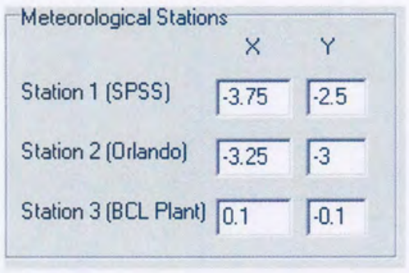
	X	Y
South-West corner of grid (X,Y) in km	-10	-10
North-East corner of grid (X,Y) in km	10	10
Grid interval in km	1	

Screen 7

Record 4 defines the coordinates of the stations where meteorological data is collected (Screen 8). This record is used if the option for user defined station coordinates is false (Astn = 'n').

Record 4

Variable	Description	Units
Xs1	x coordinates of station at SPSS.	km
Ys1	y coordinates of station at SPSS.	km
Xs2	x coordinates of station at Orlando.	km
Ys2	y coordinates of station at Orlando.	km
Xs3	x coordinates of station at BCL plant.	km
Ys3	y coordinates of station at BCL plant.	km
Zo	height for wind observation.	m



Meteorological Stations

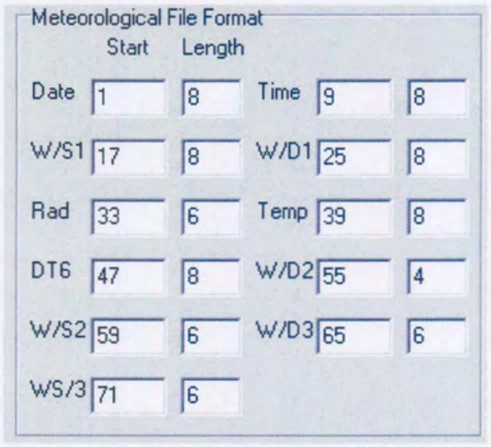
	X	Y
Station 1 (SPSS)	-3.75	-2.5
Station 2 (Orlando)	-3.25	-3
Station 3 (BCL Plant)	0.1	-0.1

Screen 8

Record 5 is the meteorological data read from a file defined in record 1. Screen 9 defines the format of the file using the start position and the length of each parameter. This record is read at the beginning of each hour.

Record 5

Variable	Description	Units
Date		yy/mm/dd
Time		hh:mm
OrlSpeed	Wind speed from the station at Orlando.	m/s
OrlWdir	Wind direction from the station at Orlando.	degrees
Nrad	Solar radiation from the station at Selebi-Phikwe Senior Secondary School (SPSS).	w/m^2
T25	Air temperature from the station at Orlando.	^{o}C
DT6	Vertical temperature gradient, between 6 m and 25 m above the station, at Orlando.	
SpWdir	Wind direction from the station at SPSS.	degrees
SpSpeed	Wind speed from the station at SPSS.	m/s
PltWdir	Wind direction from the station within the BCL plant.	degrees
PltSpeed	Wind speed from the station within BCL plant.	m/s



Meteorological File Format

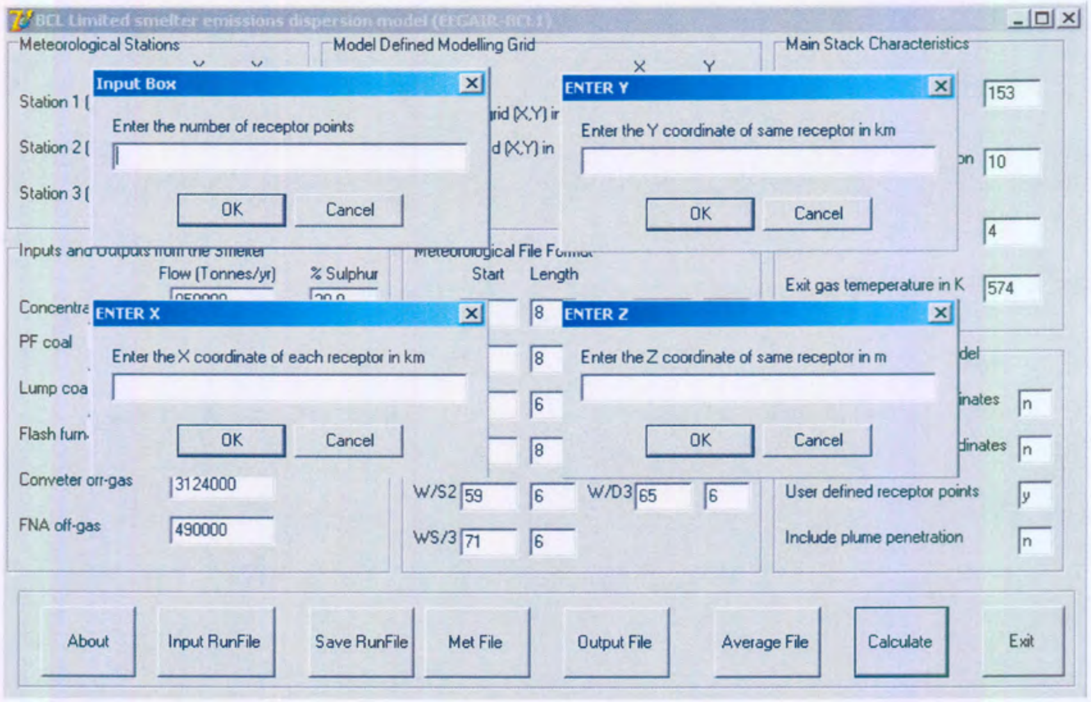
	Start	Length		Start	Length
Date	1	8	Time	9	8
W/S1	17	8	W/D1	25	8
Rad	33	6	Temp	39	8
DT6	47	8	W/D2	55	4
W/S2	59	6	W/D3	65	6
WS/3	71	6			

Screen 9

Record 6 is used if the option for user defined receptor points is true (MetString = 'y'). It defines the x, y and z coordinates of the receptor points and is executed "Nrec" times (Screen 10). The z coordinates of receptor points are set as 3 m (this is based on the height of the sample inlet manifold at the monitoring stations from groundlevel).

Record 6

Variable	Description	Units
Nrec	Number of receptor points.	
Xrec	x coordinates of the receptor point.	km
Yrec	y coordinates of the receptor point.	km
Zrec	z coordinates of the receptor point.	km



BCL Limited smelter emissions dispersion model (EEGAIR-BCL1)

Meteorological Stations — Station 1 (— Station 2 (— Station 3 (

Model Defined Modelling Grid

Main Stack Characteristics: 153; 10; 4; Exit gas temeperature in K 574

Input Box — Enter the number of receptor points — OK — Cancel

ENTER Y — Enter the Y coordinate of same receptor in km — OK — Cancel

ENTER X — Enter the X coordinate of each receptor in km — OK — Cancel

ENTER Z — Enter the Z coordinate of same receptor in m — OK — Cancel

Inputs and Outputs from the Smelter — Flow (Tonnes/yr) — % Sulphur

Concentra — PF coal — Lump coa — Flash furn. — Conveter orr-gas 3124000 — FNA off-gas 490000

Meteorological File Format — Start — Length — 8 — 8 — 6 — 8 — W/S2 59 6 — W/D3 65 6 — WS/3 71 6

inates n — dinates n — User defined receptor points y — Include plume penetration n

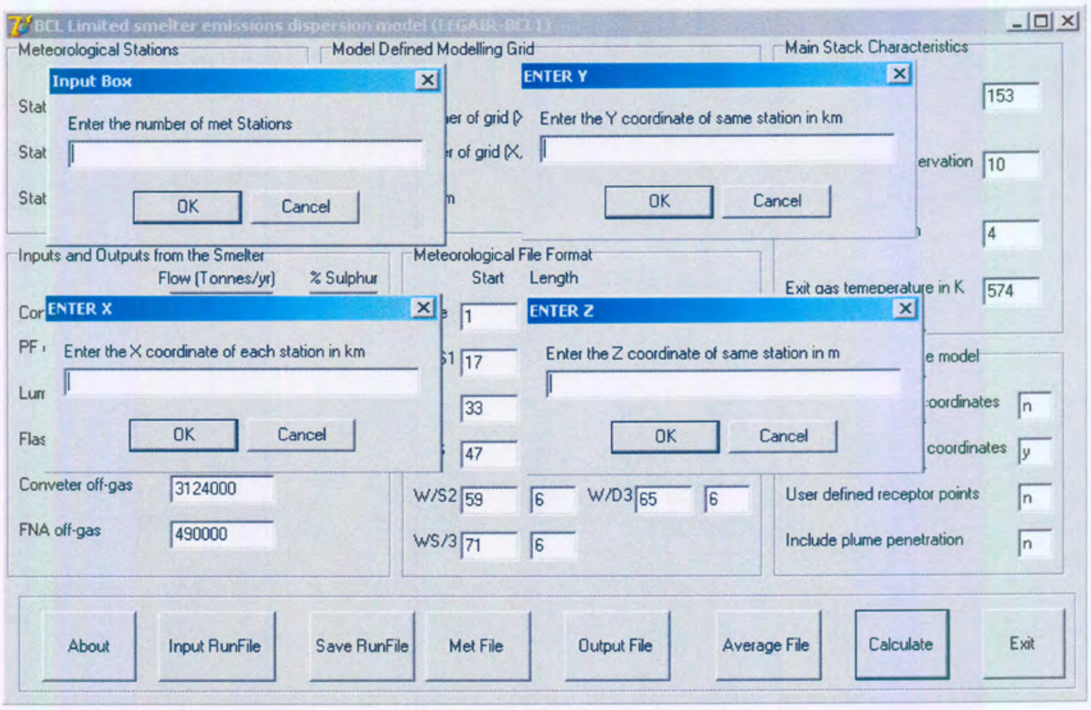
About — Input RunFile — Save RunFile — Met File — Output File — Average File — Calculate — Exit

Screen 10

Record 7 is used if the option for user defined station coordinates is true (Astn = ‘y’). It defines the x, y and z coordinates of the station coordinates and is executed “Nstn” times.

Record 7

Variable	Description	Units
Nstn	number of stations.	
Xrec	x coordinates of the station.	km
Yrec	y coordinates of the station.	km
Zrec	z coordinates of the station.	km


BCL Limited smelter emissions dispersion model (EEGAIR-BCL1)

Meteorological Stations | Model Defined Modelling Grid | Main Stack Characteristics

Input Box
Enter the number of met Stations

ENTER Y
Enter the Y coordinate of same station in km

ENTER X
Enter the X coordinate of each station in km

ENTER Z
Enter the Z coordinate of same station in m

Main Stack Characteristics: 153; ervation 10; 4; Exit gas temeperature in K 574

Inputs and Outputs from the Smelter — Flow (Tonnes/yr), % Sulphur
Conveter off-gas 3124000
FNA off-gas 490000

Meteorological File Format — Start, Length
1
17
33
47
W/S2 59 6
W/D3 65 6
WS/3 71 6

coordinates n
coordinates y
User defined receptor points n
Include plume penetration n

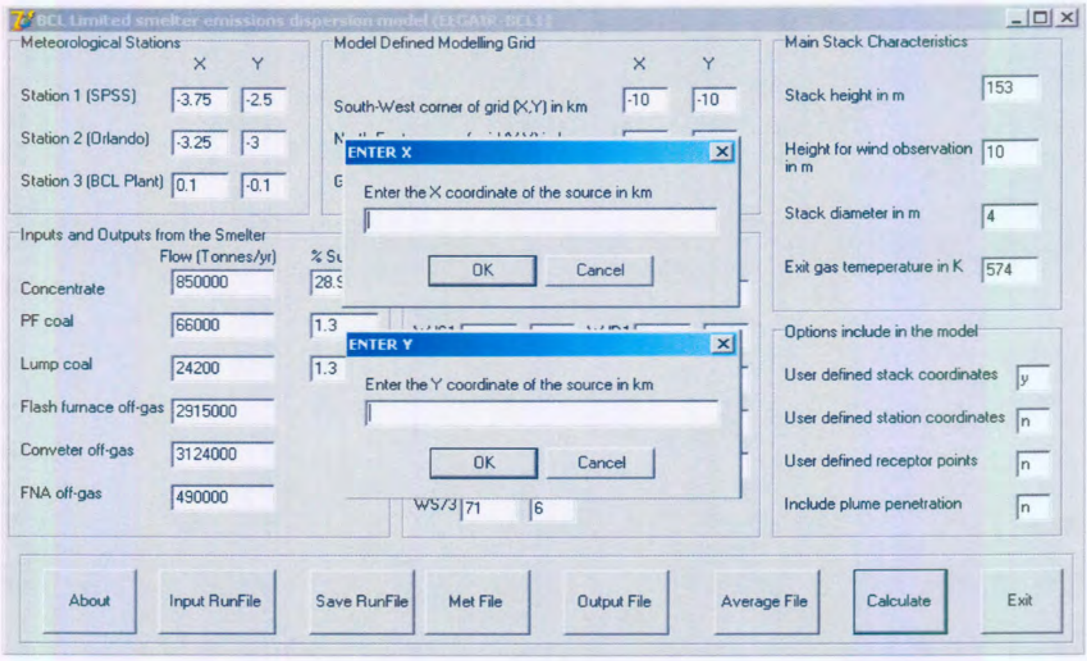
Screen 11

Record 8 is used if user defined stack coordinates is true (StkStr = 'y'). It defines the x and y coordinates of the stack (Screen 12). If not specified, the stack is placed at the origin of the modelling grid.

Record 8

Variable	Description	Units
Xs	x coordinates of the stack.	km
Ys	y coordinates of the stack.	km

Record 9 is used to describe the pollutant emission rate, stack gas temperature, stack inside diameter, stack height and stack gas exit velocity (Screens 14 and 15).


BCL Limited smelter emissions dispersion model (EEGAIR-BCL1)

Meteorological Stations

	X	Y
Station 1 (SPSS)	-3.75	-2.5
Station 2 (Orlando)	-3.25	-3
Station 3 (BCL Plant)	0.1	-0.1

Model Defined Modelling Grid

	X	Y
South-West corner of grid (X,Y) in km	-10	-10

ENTER X

Enter the X coordinate of the source in km

ENTER Y

Enter the Y coordinate of the source in km

Inputs and Outputs from the Smelter

	Flow (Tonnes/yr)	% Su
Concentrate	850000	28.9
PF coal	66000	1.3
Lump coal	24200	1.3
Flash furnace off-gas	2915000	
Conveter off-gas	3124000	
FNA off-gas	490000	

WS73 71 6

Main Stack Characteristics

Stack height in m	153
Height for wind observation in m	10
Stack diameter in m	4
Exit gas temeperature in K	574

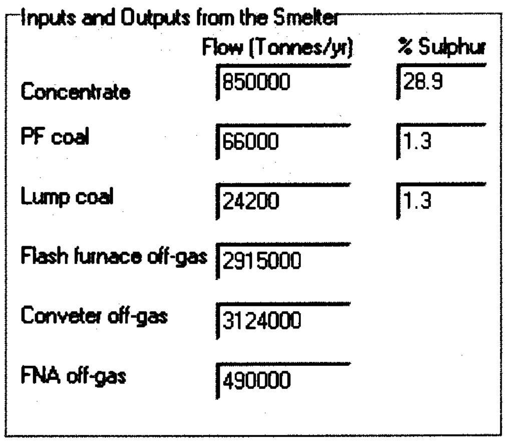
Options include in the model

User defined stack coordinates	y
User defined station coordinates	n
User defined receptor points	n
Include plume penetration	n

Screen 12

Record 9

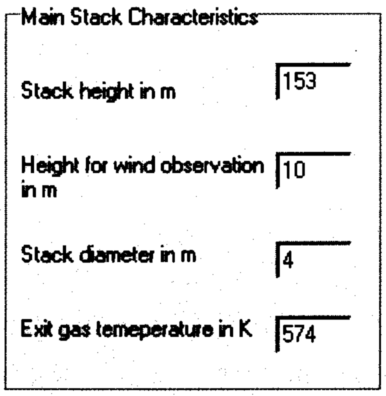
Variable	Description	Units
ConcTrate	Flow of concentrate into the smelter area.	tonnes/yr
Sconc	Sulphur content of the concentrate.	%
PFCoal	Flow of pulverised fuel (PF) coal used in the smelter.	tonnes/yr
SPFC	Sulphur content in the PF coal.	%
Lcoal	Flow of lump coal used in the smelter.	tonnes/yr
SLC	Sulphur content of the lump coal.	%
D	Stack diameter.	m
Hs	Stack height.	m
Ts	Stack gas temperature.	K
FFG	Flow of off-gas from the furnace.	tonnes/yr
CVG	Flow of off-gas from the converters.	tonnes/yr
FNG	Flow of off-gas from FNA matte.	tonnes/yr



Inputs and Outputs from the Smelter

	Flow (Tonnes/yr)	% Sulphur
Concentrate	850000	28.9
PF coal	66000	1.3
Lump coal	24200	1.3
Flash furnace off-gas	2915000	
Conveter off-gas	3124000	
FNA off-gas	490000	

Screen 14



Main Stack Characteristics

Stack height in m	153
Height for wind observation in m	10
Stack diameter in m	4
Exit gas temeperature in K	574

Screen 15

3.4 PROGRAM OUTPUT

The purpose of EEGAIR BCL1 model is to predict hourly mean air quality at monitoring sites or at specified receptor points. Defining all the records listed under Section 3.3, the model generates an output file as shown in Figure 3.3, after clicking on the **Calculate** button of Screen 2.

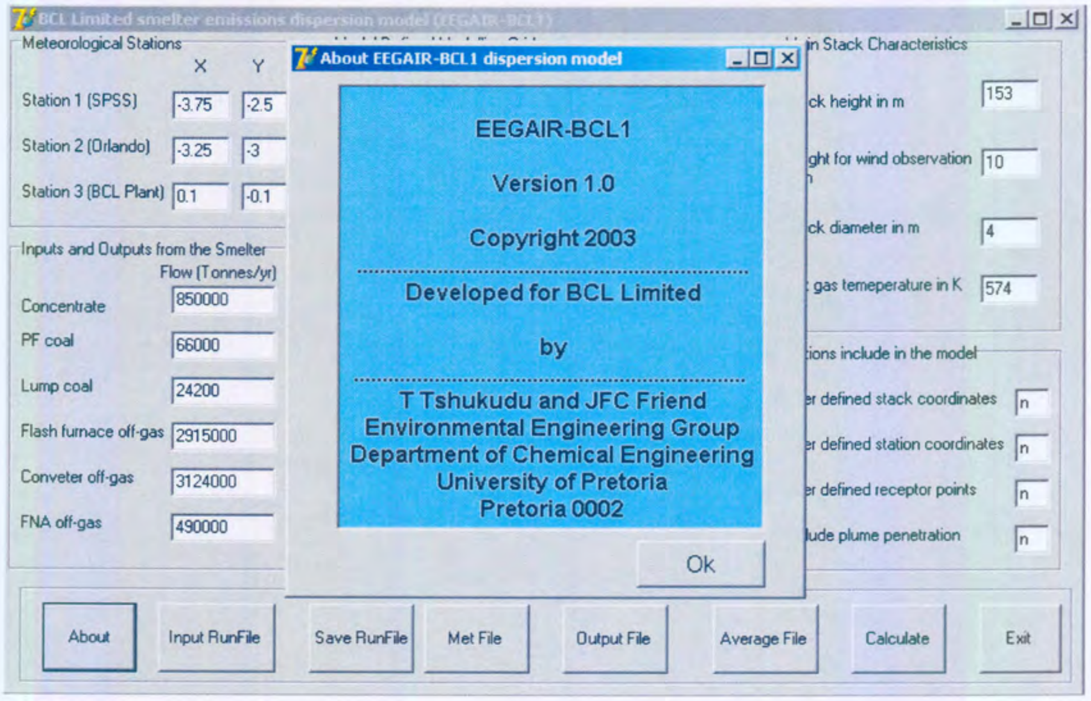
The output begins with a definition statement of the title of the model run and the governing equation. This is build within the model and can only be changed in the model source code. The next output indicates the model options of Record 2 from Section 3.3 that were executed in the program.

```
Air Quality Dispersion Model For Emissions From BCL Limited Smelter
**** The model uses the Gaussian Plume Equation *************

** MODEL OPTIONS
            A T INDICATES THAT THE OPTION HAS BEEN EXERCISED

********* USER DEFINED STACK COORDINATES        = F
********* USER DEFINED STATION COORDINATES      = F
********* USER DEFINED RECEPTOR POINTS          = T
********* PLUME PENETRATION                     = F

********* ANALYSIS OF THE MAIN STACK *************************

*********** Main Stack Input Information *****************
*** Source Strength (g/s)           14,244.58
*** Stack Height (m)                153.00
*** Stack Gas Temperature (k)       574.00
*** Stack Gas Velocity (m/s)        26.85
*** Stack Diameter (m)              4.00

************ MODEL OUTPUT ******************************
Average Concentration at Receptors for Simulation Period: 02/05/01   09:00
******** The Stability for this hour is Class:  d  **********

X(km)   Y(km)   Z(m)   Conc. (µg/m3)
-10.00  -4.00   3.00    8.70
-10.00  -3.00   3.00    1.56
-10.00  -2.00   3.00   11.12
-10.00  -1.00   3.00    3.16

************ MODEL OUTPUT ******************************
Average Concentration at Receptors for Simulation Period: 02/05/01   10:00
******** The Stability for this hour is Class:  d  **********

X(km)   Y(km)   Z(m)   Conc. (µg/m3)
-10.00  -4.00   3.00   11.46
-10.00  -3.00   3.00   19.71
-10.00  -2.00   3.00    6.01
-10.00  -1.00   3.00    0.05
```

Figure 3.3 Sample of model output.

The next output information is on the calculated value of the source strength and the stack gas exit velocity as well as the value of stack height, diameter and exit gas temperature used in the model. The output also indicates the simulation period and the stability class during the particular meteorological period. The x, y and z co-ordinates of each receptor point and the calculated value of the concentration at these receptor points for each meteorological period are also presented.

The other screens (Screen 16 to 18) generated by the model are shown below. Clicking on the **About**, **Input RunFile** and **Save RunFile** buttons of Screen 2 in Section 3.3 respectively, activates Screens 16, 17 and 18. Screen 16 displays general information about the model while Screen 17 is for defining the run file and Screen 18 for saving the run file. Clicking on the **New tfa file** button of Screen 17, Screen 2 is displayed with all the records blank while **Load tfa file** button will load the records of Screen 2 from a run file. Screen 18 is used to save all the records defined in Screen 2 to a run file.


BCL Limited smelter emissions dispersion model (EEGAIR-BCL1)
About EEGAIR-BCL1 dispersion model
Ok
About
Input RunFile
Save RunFile
Met File
Output File
Average File
Calculate
Exit

Meteorological Stations

	X	Y
Station 1 (SPSS)	-3.75	-2.5
Station 2 (Orlando)	-3.25	-3
Station 3 (BCL Plant)	0.1	-0.1

Inputs and Outputs from the Smelter

	Flow (Tonnes/yr)
Concentrate	850000
PF coal	66000
Lump coal	24200
Flash furnace off-gas	2915000
Conveter off-gas	3124000
FNA off-gas	490000

EEGAIR-BCL1

Version 1.0

Copyright 2003

Developed for BCL Limited

by

T Tshukudu and JFC Friend
Environmental Engineering Group
Department of Chemical Engineering
University of Pretoria
Pretoria 0002

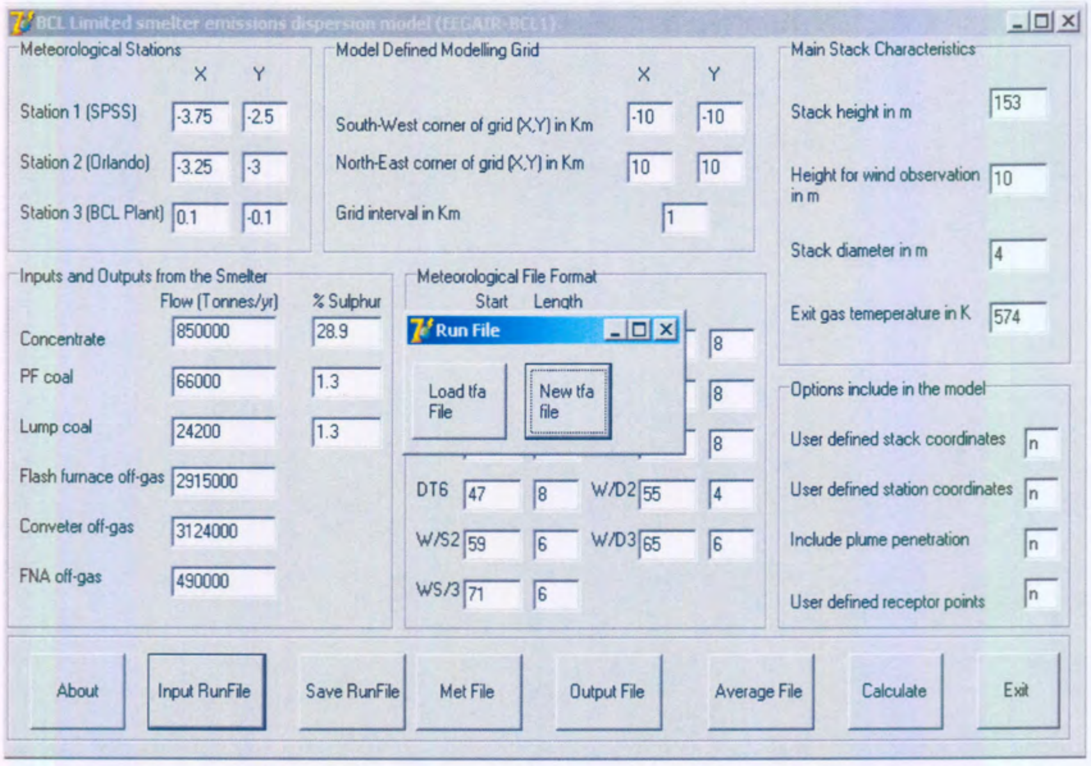
in Stack Characteristics

ck height in m	153
ght for wind observation	10
ck diameter in m	4
gas temeperature in K	574

ions include in the model

er defined stack coordinates	n
er defined station coordinates	n
er defined receptor points	n
lude plume penetration	n

Screen 16



Meteorological Stations

	X	Y
Station 1 (SPSS)	-3.75	-2.5
Station 2 (Orlando)	-3.25	-3
Station 3 (BCL Plant)	0.1	-0.1

Model Defined Modelling Grid

	X	Y
South-West corner of grid (X,Y) in Km	-10	-10
North-East corner of grid (X,Y) in Km	10	10
Grid interval in Km	1	

Main Stack Characteristics

Stack height in m	153
Height for wind observation in m	10
Stack diameter in m	4
Exit gas temeperature in K	574

Inputs and Outputs from the Smelter

	Flow (Tonnes/yr)	% Sulphur
Concentrate	850000	28.9
PF coal	66000	1.3
Lump coal	24200	1.3
Flash furnace off-gas	2915000	
Conveter off-gas	3124000	
FNA off-gas	490000	

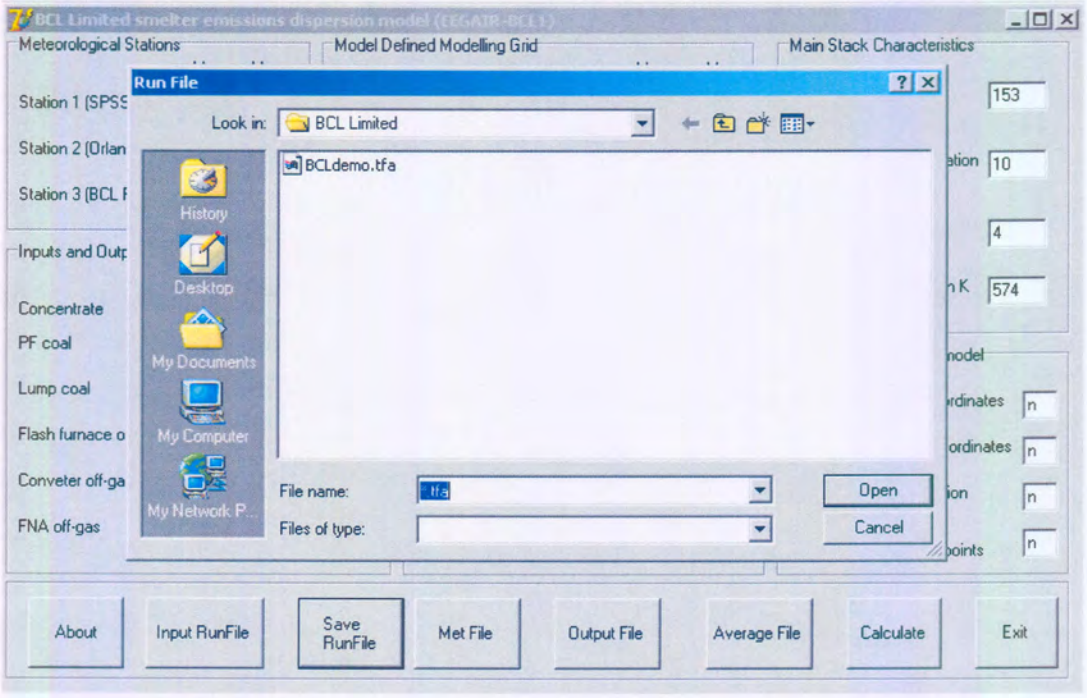
Meteorological File Format

	Start	Length		Start	Length
					8
					8
					8
DT6	47	8	W/D2	55	4
W/S2	59	6	W/D3	65	6
WS/3	71	6			

Options include in the model

User defined stack coordinates	n
User defined station coordinates	n
Include plume penetration	n
User defined receptor points	n

Screen 17



Look in: BCL Limited

BCLdemo.tfa

File name: *.tfa

Files of type:

Screen 18

CHAPTER 4

Model Validation

4.1 INTRODUCTION

Dispersion models allow the comparison of various options to improve air quality. However, these models need to be validated using monitoring data. Their accuracy depend on many factors; including the accuracy of the source emission data, the quality of meteorological data in the area, and the assumption about the physical and chemical processes in the atmosphere, involving the transport and transformation of pollutants (Elbir, 2003).

Technically speaking, air dispersion models accomplish two principal objectives, namely:

- simulation of the downwind dispersion process, and
- simulation of the movement and behaviour of an emission plume.

Although different types of models exist to accomplish these objectives, Gaussian models are widely used for regulatory purposes. In order to validate these models, statistical analyses can be used in comparing predicted values from a model with those observed from actual monitoring stations.

MONITORING PROGRAMME

In the Selebi-Phikwe area, BCL Limited and the government of Botswana are operating six monitoring stations; three to monitor sulphur dioxide (SO_2) emissions from the smelter; and three stations supplying meteorological data (the Selebi-Phikwe Secondary School – SPSS, monitors SO_2 emissions and supplies meteorological data). Table 4.1 shows the parameters monitored at each station. The meteorological data will be used as input to the model while the air quality data will be used to test the model results.

Table 4.1 Monitoring stations in Selebi-Phikwe.

Stations	Measured parameters					
	Wind speed	Wind direction	Temp	Solar radiation	Temp gradient	SO_2
BCL Limited plant	x	x				
Kopano						x
Orlando	x	x	x		x	
SPSS	x	x	x	x		x
Water Utilities (WUC)						x

4.2.1 Meteorology

The meteorological data used as the model input was made available by both BCL Limited and the government of Botswana. Figure 4.1 shows the locations of each monitoring station with respect to the BCL Limited stack. The meteorological data for May 2002 was considered appropriate for the purpose of testing the EEGAIR-BCL1 dispersion model (this was the month with the highest number of peak concentrations during 2002).

The wind direction data (Table 4.2) shows that the direction of prevailing winds in Selebi-Phikwe is from the sector between east and south (90^o – 180^o) with percentage occurrence of 44,57%, 42,44% and 51,94% for BCL Limited plant, Orlando and Selebi-Phikwe Secondary School (SPSS) stations respectively.

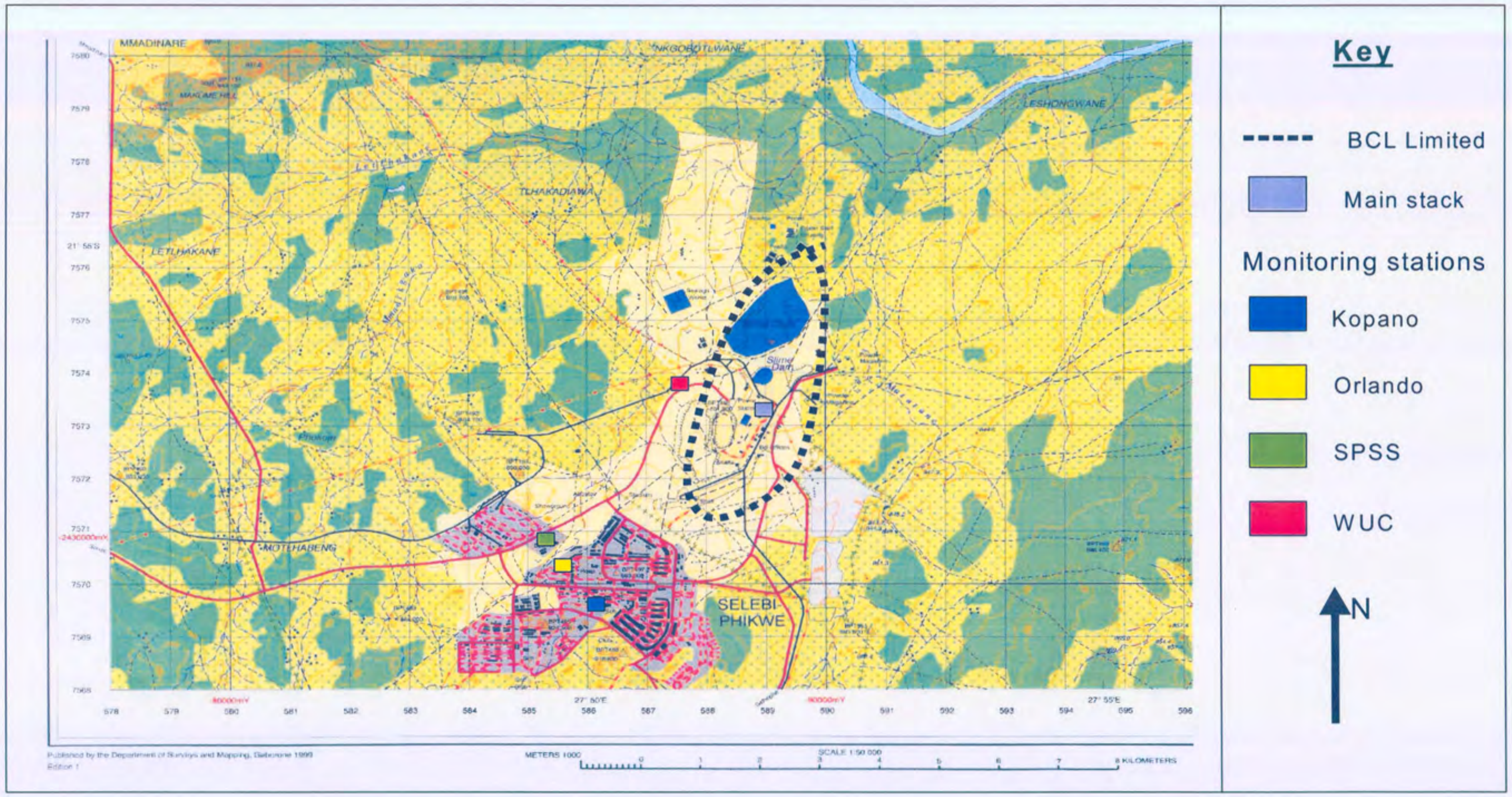


Figure 4.1 Map of Selebi-Phikwe showing monitoring stations and BCL Limited stack.

Table 4.2 Percentage of occurrence for wind direction sectors.

Degrees	BCL Limited plant	Orlando	SPSS
0 – 30	0,00	5,23	0,19
30 – 60	0,00	3,29	1,74
60 – 90	0,78	21,12	13,76
90 – 120	0,58	20,93	29,07
120 – 150	15,31	12,40	13,95
150 – 180	26,68	9,11	8,91
180 – 210	14,15	10,08	9,50
210 – 240	16,09	6,59	6,59
240 – 270	8,14	3,68	7,75
270 – 300	7,56	2,13	6,40
300 – 330	8,14	3,49	1,94
330 – 360	0,58	1,94	0,19

Table 4.3 shows that the BCL Limited plant station recorded the highest percentage of calm winds (wind speed less than 0,6 m/s) with 54,84% compared to 24,22% at Orlando and 0,00 % at SPSS. This station is located within the BCL Limited premises, which is characterised by high buildings that will influence both the wind direction and speed measured at this station. The SPSS station did not record any calm winds, as it is located in an open area compared to Orlando station that is located within a residential area.

Table 4.3 Percentage of occurrence for wind speed classes.

Wind speed	BCL Plant	Orlando	SPSS
Calm	54,84	24,22	0,00
0,6 – 1	11,82	19,77	0,19
1 – 2	11,05	27,71	6,78
2 – 3	7,95	15,31	37,02
3 – 4	6,59	7,56	28,88
4 – 5	4,84	5,04	11,24
5 – 6	2,91	0,39	9,50
> 6	0,00	0,00	6,40

4.2.2 Measured Concentrations

Air pollution is one of the most prevalent environmental problems in Selebi-Phikwe, Botswana. An ambient air quality monitoring network has been implemented with the purpose of monitoring the impact of the BCL Limited smelter stack in the residential areas of Selebi-Phikwe. The monitoring network consists of three permanent stations located at Kopano, SPSS and WUC. Figure 4.1 shows the location of the three stations in the Selebi-Phikwe area. The stations are located at a distance of approximately 4,45 km, 4,5 km and 1,3 km from the stack for Kopano, SPSS and WUC respectively.

Both BCL Limited and the government of Botswana made the air quality data from the above stations available for this research. Data for the month of May 2002 will be compared with results obtained from the EEGAIR-BCL1 dispersion model.

4.3 MODEL PERFORMANCE EVALUATION

Model evaluation will be done through the variation of the measured and predicted hourly average SO_2 concentrations and, evaluating the agreement of the model predictions with observations using statistical analyses of the hourly average concentrations. The statistical analyses included the calculation of:

- index of agreement,
- correlation coefficient, and
- the root mean square error (RMSE).

The index of agreement (d) determines the degree to which magnitudes and signs of a measured value about a mean concentration are related to the model results deviation about the model average, and allows for sensitivity towards difference in measured and calculated values, as well as proportionality changes (Elbir, 2003). It is a measure of the degree to which model results are error free.

The index of agreement (d) is calculated as follows (Elbir, 2003):

$$d = 1 - \frac{\sum_{i=1}^{N} (P_i - O_i)^2}{\sum_{i=1}^{N} \left(|P_i - O| + |O_i - O|\right)^2} \qquad [4.1]$$

where N = number of data points,
P_i = model prediction,
O_i = measured concentration, and
O = mean of the measured data.

The index of agreement (d) varies from 0,0 (theoretical minimum) to 1,0; which indicates perfect agreement between the measured concentration and model results (Elbir, 2003).

The correlation coefficient is a generalised measure of the relationship between pairs of variable from two samples. It is a number between –1 and 1 that measures the degree to which two variables are linearly related. The correlation coefficient (r) is given by (Elbir, 2003):

$$r = \left[\frac{\sum_{i=1}^{N} (O_i - O)(P_i - P)}{\sigma_o \sigma_p} \right] \qquad [4.2]$$

where P_i = model prediction,

P = mean of model predictions,

O_i = measured concentration,

O = mean of the measured data,

σ_o = standard deviations of the measured concentrations, and

σ_p = standard deviations of the predicted concentrations.

Correlation coefficients of 1 indicate perfect positive linearity between two variables (as one variable increases, so does the other variable), and coefficients of -1 indicate perfect negative linearity between two variables (as one variable increases, the other decreases). A correlation coefficient of zero means there is no linear relationship between the variables (Paul, 1989).

The RMSE is a measure of the model's error and consists of the systematic RMSE and the unsystematic RMSE components. It indicates the sources or types of errors, which can be of considerable help in refining a model. When RMSE is entirely systematic, further refinement in the model is required for the model to predict at its maximum possible accuracy. On the other hand, if RMSE is entirely, or largely composed of unsystematic RMSE, the model may not require any major modification (Ku, 1984).

The systematic component of the RMSE is given by (Elbir, 2003):

$$RMSE_s = \left(\frac{1}{N} \sum_{i=1}^{N} \left(\bar{O}_i - O_i \right)^2 \right)^{1/2} \qquad [4.3]$$

where $\bar{O}_i$ = least square linear regression of P_i and O_i and is given by:

$$\bar{O}_i = a + bO_i$$

where a = intercept of the regression line, and
b = slope of the regression line.

The unsystematic component of RMSE is given by (Elbir, 2003):

$$RMSE_u = \left(\frac{1}{N} \sum_{i=1}^{N} \left(\bar{O}_i - P_i \right)^2 \right)^{1/2} \text{ and} \qquad [4.4]$$

$$Total\ RMSE^2 = RMSE_s{}^2 + RMSE_u{}^2$$

For a good model, the total RMSE value should be close to 0,0 for a good prediction (Elbir, 2003); and the systematic RMSE should approach zero, while the unsystematic RMSE should approach total RMSE (Paul, 1989).

4.3.1 Time Average Concentrations

The model was used to predict the SO_2 concentration at receptor points placed at the location of the three SO_2 monitoring stations as a function of time.

Figures 4.2 to 4.4 shows the hourly average predicted and measured concentration as a function of time at Kopano, SPSS and WUC stations respectively (for the relevant dates these monitoring stations were in operation). Some correlation between the model predicted concentrations and those observed at Kopano and SPSS do exist. However, there is no trend between the predicted and observed concentrations at WUC station. This station is located approximately 1,3 km from the 153 m stack and is too close to be impacted by the stack unless under the most unstable of conditions. In the case of BCL Limited, this station is mostly impacted by low level sources like the diffused emissions from the converters during matte blowing. However, the impact from diffuse emissions was not included in the EEGAIR-BCL1 dispersion model predictions.

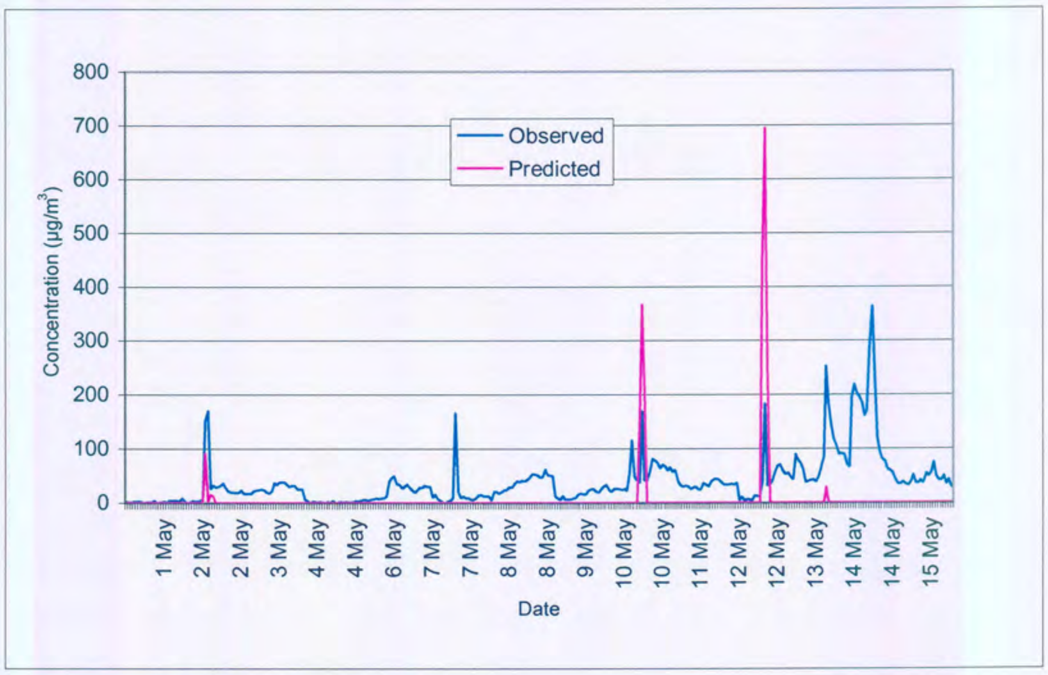


Figure 4.2 Graphical representation of predicted and observed hourly SO_2 concentration at Kopano for the period 1 to 15 May 2002.

The predicted and measured SO_2 concentrations were the highest at the SPSS station. This can be ascribed to the location of the SPSS station, close to the sector of prevailing winds in the area. Model results also showed that the model did not predict any concentrations of note during nighttime, within the modelling grid; whereas measurements were made during the same period. This is due to nighttime being characterised by neutral to stable conditions, thereby resulting in the model predicting the plume to travel outside the modelling grid before reaching groundlevel. The time plots for

Kopano and SPSS shows that the air quality at these stations is characterised by occasional short term peak concentrations and long periods of low concentrations.

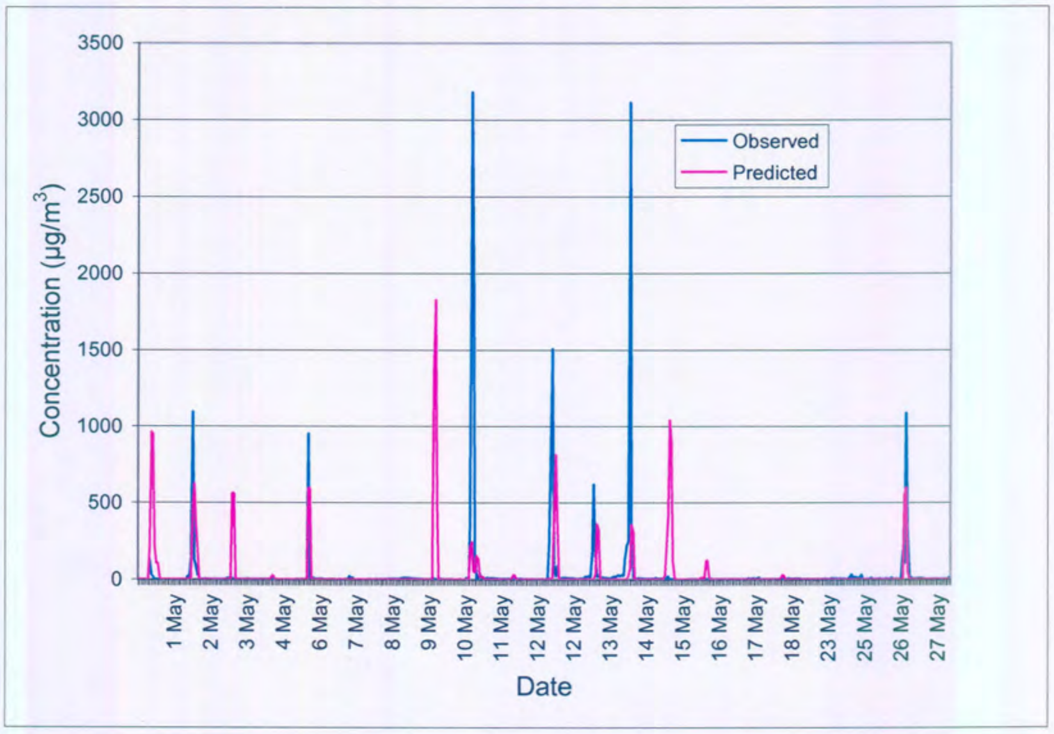


Figure 4.3 Graphical representation of predicted and observed hourly SO_2 concentration at SPSS for the period 1 to 27 May 2002.

Figure 4.5 shows the average predicted and measured SO_2 concentrations for May 2002 at the three stations. The measured average concentrations were higher than the predicted average concentrations at all the stations.

The uncertainty of under prediction by the model generally arises from three different sources, namely:

- quality of the observed concentration measurements,
- quality of the emission data, and
- dispersion modelling.

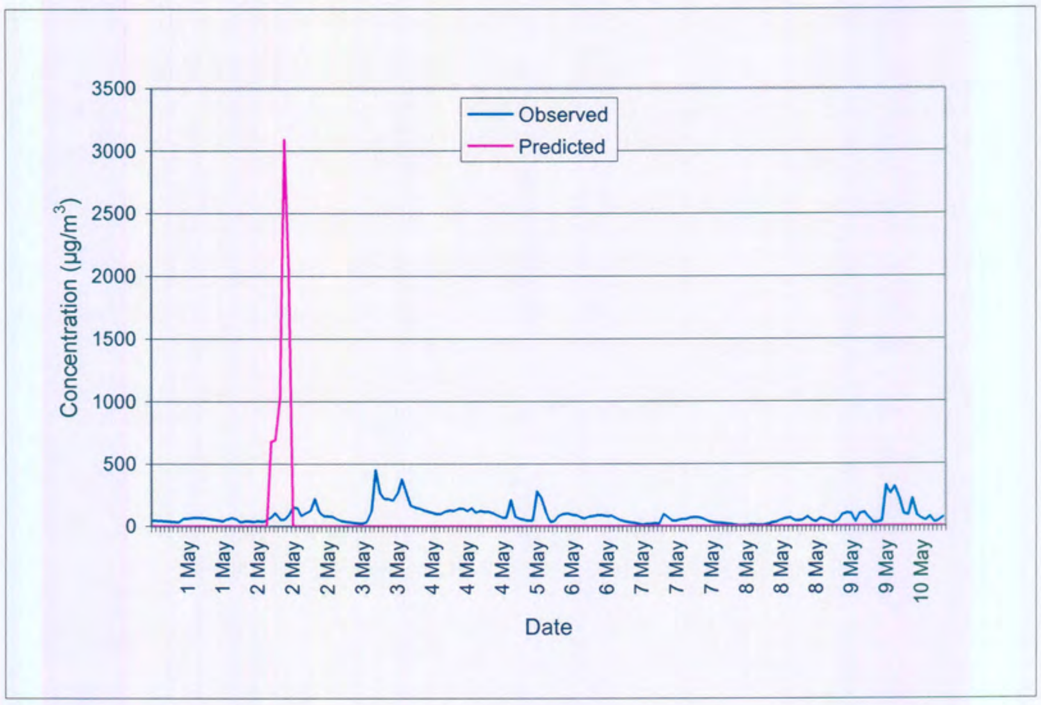


Figure 4.4 Graphical representation of predicted and observed hourly SO_2 concentration at WUC for the period 1 to 10 May 2002.

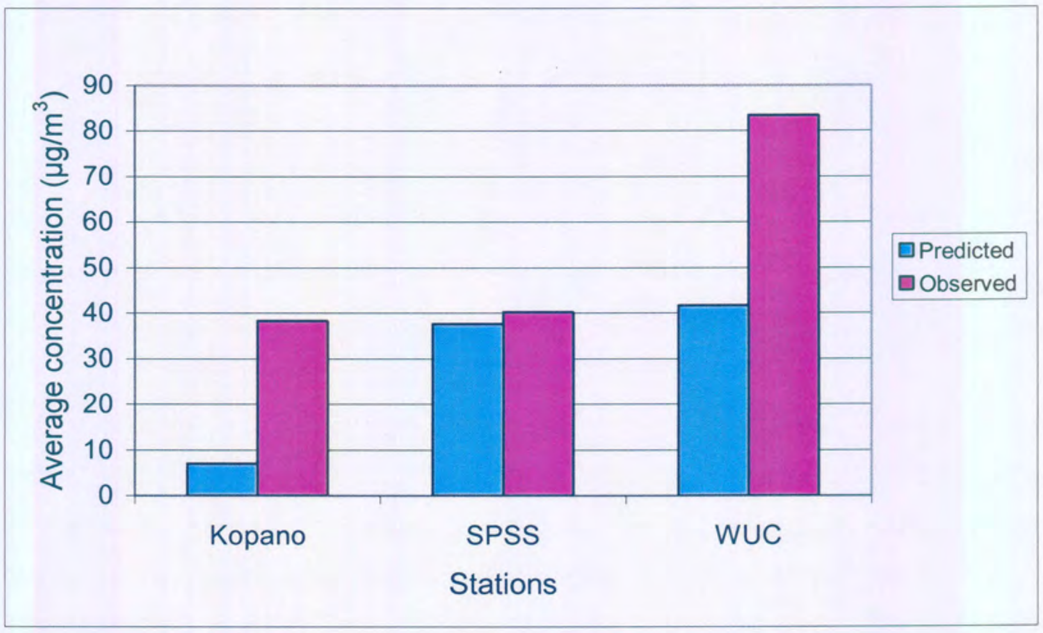


Figure 4.5 The predicted and observed average concentrations for May 2002.

The quality of the observed concentration measurements is contributing to the model's uncertainty. The measurement stations sometimes do not measure high concentrations in situations where the prevailing winds are in the direction of the monitoring site. This can be due to the wind direction fluctuating during the averaging period whereas the model is not taking this into account. The other factor contributing to the quality of observed concentrations is the quality of the standard gas used in the calibration of the monitors.

The contribution of the emissions data to the model's uncertainty is based on the use of the annual production data to estimate the hourly emission rate for the simulation period. This was assuming that the hourly production for May 2002 approximates the hourly average of the annual production rate. The emission rate through the stack varies as a function of time; therefore the best result with the model will be to use the daily or hourly production rates to estimate the hourly emission rate.

The option of the plume penetration into the elevated stable layer was not included in the model calculations. This was due to lack of data describing the atmospheric boundary layer to estimate the height of the inversion layer in Selebi-Phikwe.

4.3.2 Statistical Analysis of the SO_2 results

Various statistical parameters were utilised in the comparison of predicted and measured data sets. Table 4.4 shows the mean, maximum and the standard deviation, together with other statistical parameters for the measured and predicted hourly SO_2 concentration for May 2002.

The mean difference was calculated as 31,3; 2,66 and 41,8 for Kopano, SPSS and WUC stations respectively; indicating a trend towards under prediction by the model. The average index of agreement (d) for the month of May 2002 varied from 0,08 to 0,44 at the three stations. This indicates that the model predictions were more accurate for Kopano (0,44) and SPSS (0,25) stations than for WUC (0,08) station. The index of agreement should be close to 1,0 for a good prediction (Elbir, 2003). In order to further assess the relationship, the index of agreement as a function of time was calculated. Figures 4.6 to 4.8 shows the daily variation of the index of agreement at Kopano, SPSS and WUC stations respectively. The daily variations indicated that the model predictions were more accurate on certain days compared to the monthly variation. This is possibly as a result of a particular day's actual emission being closely representative to the yearly averaged emission data used in the model.

Table 4.4 Statistical analyses of the predicted and measured or observed hourly SO_2 concentration in May 2002.

Statistical parameters	Kopano		SPSS		WUC	
	P*	O*	P*	O*	P*	O*
Mean ($\mu g/m^3$)	7,00	38,30	37,60	40,26	41,60	83,40
Max ($\mu g/m^3$)	694,00	363,00	1829,00	3179,00	3092,00	541,00
Stand deviation	54,10	47,44	164,20	248,60	293,00	71,54
No of Obs	324		492		182	
d	0,44		0,25		0,083	
R	-2,8E-12		1,11E-11		3,19E-12	
$RMSE_s$	48,2		226,1		90,7	
$RMSE_u$	52,9		162,5		292,8	
Total RMSE	71,6		278,5		306,5	

* P = predicted, O = observed.

The correlation coefficient (Table 4.4) for the three stations is approximately zero. This suggests a considerable error in the individual case-by-case comparison between measured concentrations and predicted concentrations. The possible source of the error in the individual case-by-case comparison is the quality of both the meteorological data and concentration measurements, and the quality of the emission data.

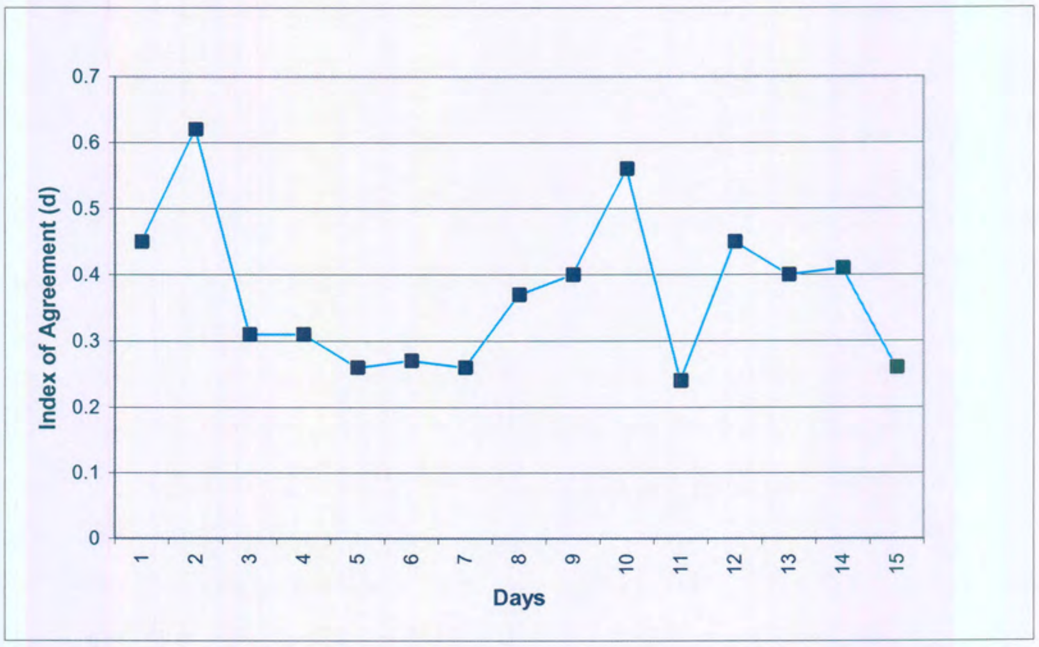


Figure 4.6 Daily variation of index of agreement at Kopano.

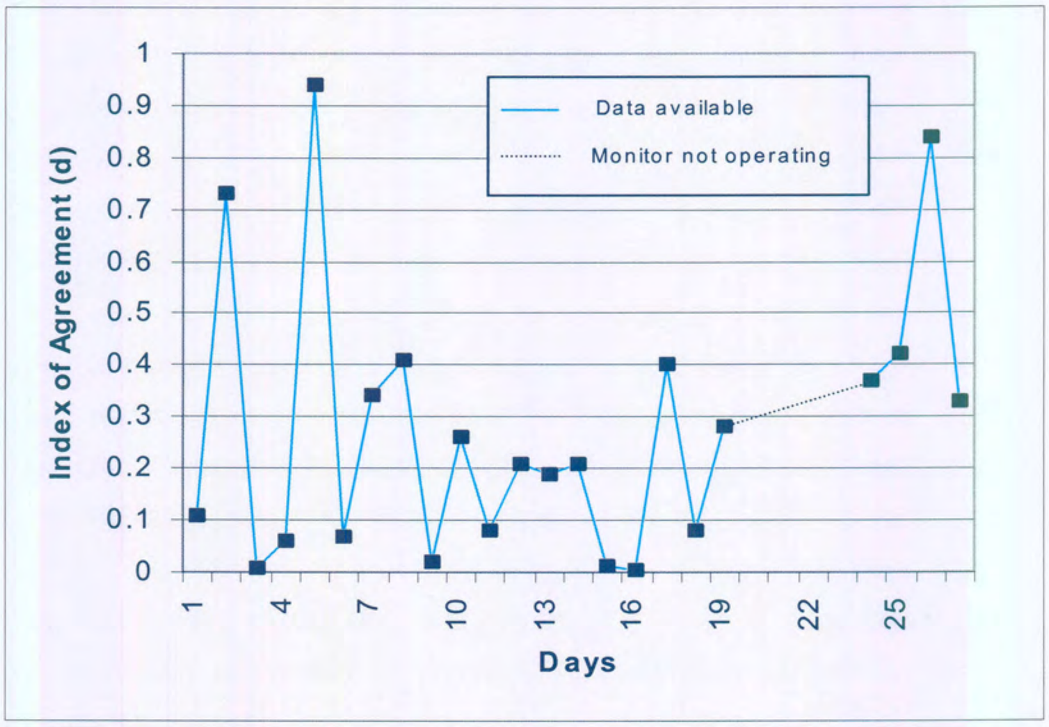


Figure 4.7 Daily variation of index of agreement at SPSS.

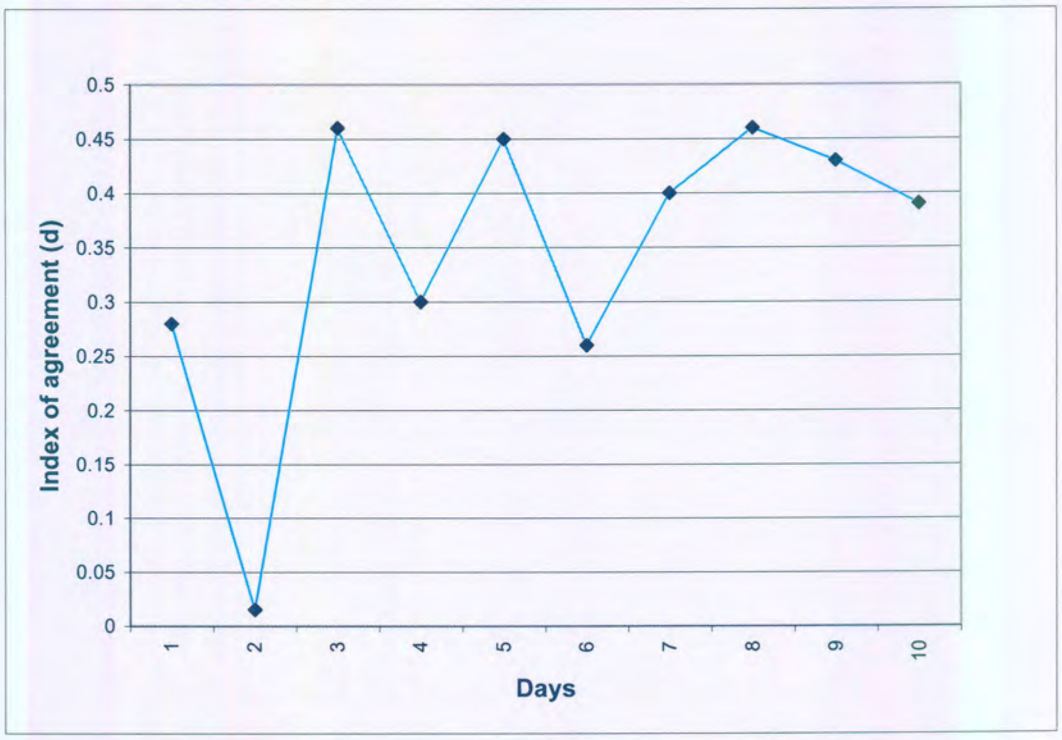


Figure 4.8 Daily variation of index of agreement at WUC.

The analysis of the RMSE from Table 4.4 indicates that the model's total RMSE varied from 71,6 to 306,5. The contribution to the total RMSE was from both systematic and unsystematic RMSE at all stations. This suggests that the model can approximate concentrations at these stations but still needs to be refined to minimise errors. Except for emission data, the source of the systematic error may come from meteorological data and model formulation (Ku, 1984). The error on the meteorological data from the Selebi-Phikwe area is unknown. The model formulation on various atmospheric conditions may cause difference performance during daytime than nighttime. The atmospheric conditions during nighttime are mostly stable and neutral while daytime is mostly unstable and neutral. Time plots showed that a large deviation occurred during nighttime periods compared to daytime.

The systematic error in a model can be minimised by calibrating the model. However, EEGAIR-BCL1 is a short term model and can not be calibrated. Calibration of short term models is not common practice and is subject to much greater error and misunderstanding (USEPA, 2003). There have been some attempts by some to compare short term estimates and concentration measurements on an event by event basis, and then to calibrate a model with results of that comparison. This approach is severely limited by uncertainty in both source and meteorological data and therefore difficult to precisely estimate concentrations at an exact location for a specific increment in time. Therefore short term model calibration is not recommended (USEPA, 2003).

4.3.3 Model Results

The meteorological data for May 2002 was used to predict the concentration in a 21 km by 21 km grid around the Selebi-Phikwe area. Figure 4.9 shows the monthly average predicted concentrations at receptor points in the modelling grid with the BCL Limited smelter placed at the origin (0,0,0). The figure shows that the highest impact of emissions from BCL Limited smelter is not in the residential area of Selebi-Phikwe. For the month of May 2002, the monthly average at the three monitoring stations in residential areas was predicted to be between 0 and 50 $\mu g/m^3$. The figure shows that the maximum predicted average was between 200 and 250 $\mu g/m^3$ at a distance of 4 to 7 km west from the stack.

The meteorological data for 16 May 2002 was also used to predict the daily average for all receptors in a 21 km by 21 km modelling grid. Figure 4.10 shows the predicted daily average concentration at specified receptor points in the modelling grid. The maximum daily average was predicted between 500 and 550 $\mu g/m^3$ at a distance of 5 to 6 km west from the stack. The daily average concentration in residential areas of Selebi-Phikwe was predicted to be between 0 and 50 $\mu g/m^3$.

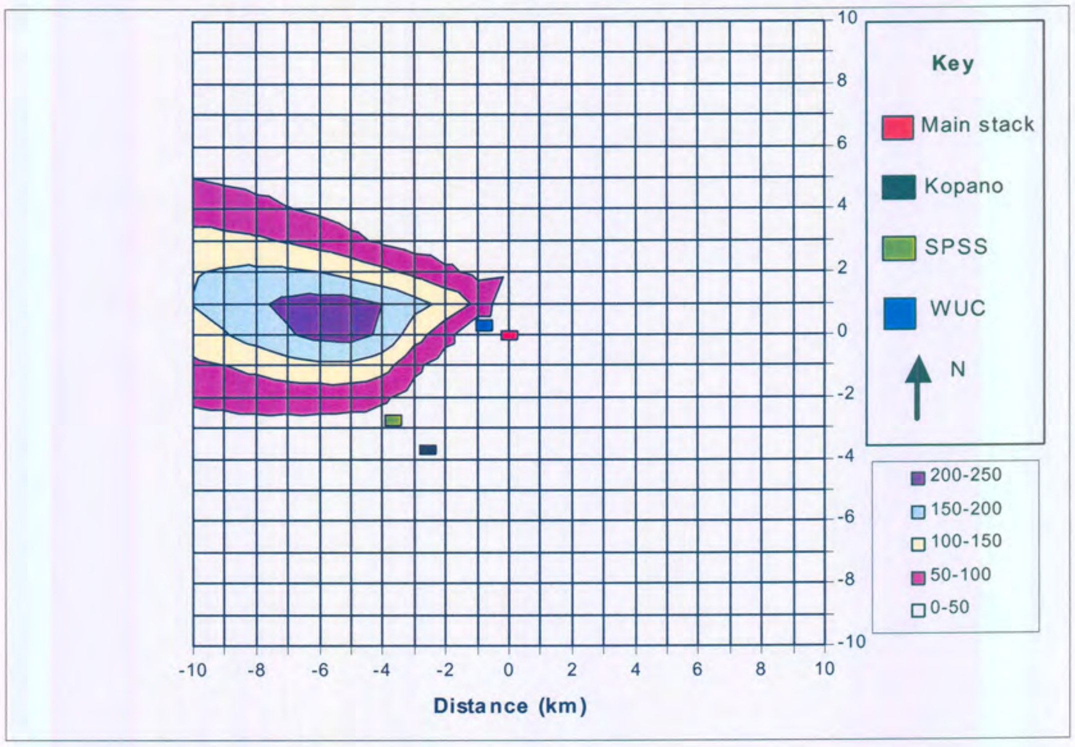


Figure 4.9 Monthly average concentrations for May 2002 at receptor points.

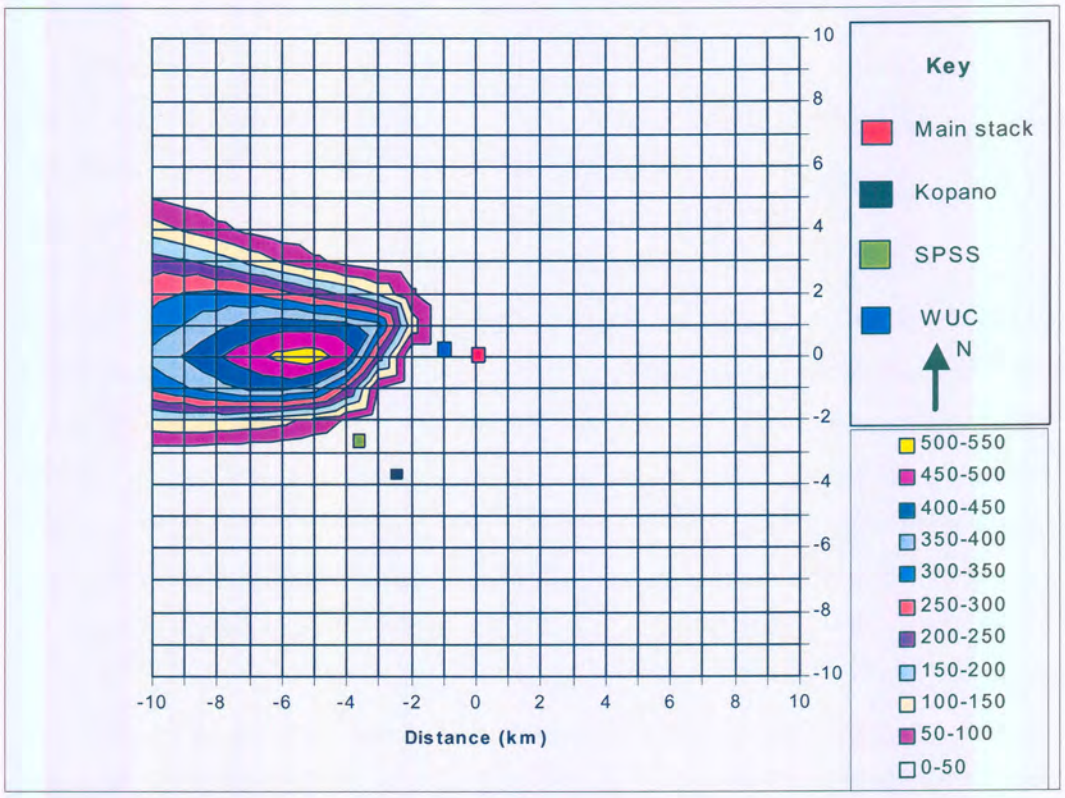


Figure 4.10 Daily average concentrations for 16 May 2002 at receptor points.

The meteorological data for 11:00 on 16 May 2002 was used to predict the hourly concentration for all receptors in a 21 km by 21 km modelling grid. Figure 4.11 shows the predicted hourly concentration at specified receptor points in the modelling grid. The maximum hourly concentration was predicted between 1 800 and 2 000 µg/m^3 at a distance of 4 to 6 km west from the stack. The hourly concentration in residential areas of Selebi-Phikwe was predicted to be between 0 and 200 µg/m^3 at this hour.

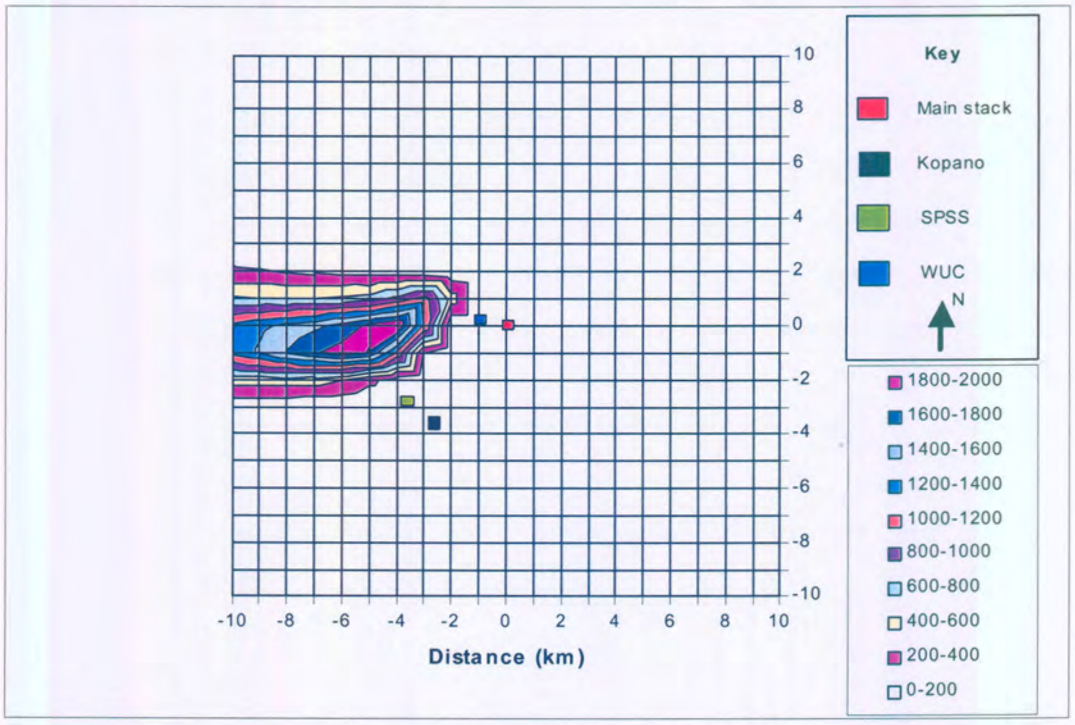


Figure 4.11 Hourly concentrations for 11:00 on 16 May 2002 at receptor points.

The model results showed that the impact from BCL Limited emissions was in a westerly direction from the stack for the month of May. This coincides or is in agreement with the prevailing wind data given in Table 4.2, which indicates prevailing winds from the southeast. This is an area with no measurement stations at present, but is also away from the residential area of Selebi-Phikwe.

CHAPTER 5

Conclusions and recommendations

The EEGAIR-BCL1 dispersion model was developed for BCL Limited to complement the air quality monitoring programme in the Selebi-Phikwe area. The model was developed for the smelter specifically, using local meteorological data at Selebi-Phikwe and smelter specific emission data. Results from the model were tested against data obtained from existing measurement stations at Selebi-Phikwe using various statistical methods.

The comparison between model predicted and measured concentrations and concentration difference (trend) indicates that some correlation exist. The average index of agreement for May 2002 indicates that the model predictions were more accurate for Kopano (0,44) and SPSS (0,25) stations than for WUC (0,08) station. The impact at WUC station is mostly from low level sources like the diffused emissions from the converters during matte blowing. The main possible cause for lack of better correlation between predicted and measured values could be due to the emission data input. Yearly averaged values are used in the model that minimise variation in groundlevel concentrations, as would be observed by the measurement stations.

The model predicted and observed concentrations indicates that the air quality in residential areas of Selebi-Phikwe is characterised by occasional short term peak concentations from the smelter stack and by long periods of low concentrations.

The model results indicates that the highest impact from the smelter stack during May 2002 was at a distance of approximately 4 km to 7 km west from the stack. This coincides or is in agreement with the prevailing wind data from the Selebi-Phikwe area, which indicates prevailing winds from the southeast. This is an area with no measurement stations at present, but is also away from the residential area of Selebi-Phikwe. The model predicted concentrations indicate th ct from the smelter stack in residential areas of Selebi-Phikwe was between 0 and 50 μg m^3 for May 2002.

In Selebi-Phikwe, further investigation should include the monitoring of parameters describing the atmospheric boundary layer to estimate the height of the inversion layer in Selebi-Phikwe. This will require a further modification of the model to incorporate the plume penetration into an elevated stable layer.

The siting or locating of additional monitoring stations in the area between west and north (270^o – 360^o) of the smelter stack at a distance greater than 3 km from the stack is another area to be considered. This is the area of highest impact from the smelter stack emissions. The monitoring programme should also be expanded to include particulates and selected metals. Monitoring of the stack emission rate on an hourly or daily basis is also required for accurate simulation within the dispersion model program.

Finally, it is recommended that future work include a study that will identify any vulnerable human population, flora and fauna and also provide an estimate of the health and environmental risks that the operations pose to these target areas.

References

BCL LIMITED (1982) Botswana's copper nickel mining complex. *BCL Limited publication*, Selebi-Phikwe, Botswana.

BØHLER T (1987) Users guide for the Gaussian type dispersion models CONCX and CONDEP. *Norwegian Institute for Air Research publication*, Lillestrom, Norway.

CALVERT S and ENGLUND HM (1984) *Handbook of Air Pollution Technology*. John and Sons Inc., USA.

COLLS J (1997) *Air Pollution: An Introduction.* E & FN Spon, London.

COOPER CD and ALLEY FC (1994) *Air Pollution Control, A Design Approach*. Second Edition, Waveland Press, Illinois.

DOUGLAS D (1996) *Assessment of two interpolation methods, inverse distance weighting and geostatistical kriging*, http://www.carleton.ca/~cwilson/interpolation/INTERPOL.htm [13 May 2003].

ELBIR T (2003) Comparison of model predictions with the data of an urban air quality monitoring network in Izmir, Turkey. *Atmospheric Environment*, **37**, pages 2149 – 2157.

HAGGER M (2003) BCL Limited smelter flow, *personal communication*, Manager: Environmental Department. May 2003, Selebi-Phikwe, Botswana.

KU JY (1984) *Numerical Simulation of Air Pollution in Urban Areas.* Dissertation report, Department of Science and Mathematics, State University of New York at Albany, USA.

PAUL NC (1989) *Encyclopaedia of Environmental Control Technology*. Gulf Publishing Company, Houston, Texas.

SCOTT HM, SOSKOLNE CL, MARTIN SW, ELLEHOJ EA, COPPOCK RW, GUIDOTTI TL and LISSEMORE KD. (2003) Comparison of two atmospheric dispersion models to assess farm site exposure to sour gas processing plant emissions. *Preventive Veterinary Medicine*, **57**, pages 15-34.

SIVERTSEN B (2000) Understanding air quality measurements. *Norwegian Institute for Air Research publication*, Kjeller, Norway.

STERN AC (1976) *Air Pollution: Air Pollutants, their transformation and transport*. Third Edition, Volume 1, Academic Press, New York, USA.

USEPA (1992) Screening procedures for estimating the air quality impact of stationary sources. *United States Environmental Protection Agency publication,* October 1992, North Carolina.

USEPA (1995) User's guide for the industrial source complex (ISC3) dispersion models. *United States Environmental Protection Agency publication*, September 1995, North Carolina.

USEPA (2000) Meteorological monitoring guidance for regulatory modeling application. *United States Environmental Protection Agency publication*, February 2000, North Carolina.

USEPA (2003) Guidelines on air quality models. *United States Environmental Protection Agency publication*, April 2003, North Carolina.

VARDOULAKIS S, FISHER BEA, GONZALEZ-FLESCA N and PERICLEOUS K (2002) Model sensitivity and uncertainty analysis using roadside air quality measurements. *Atmospheric Environment*, **36**, pages 2121-2134.

VENKATRAM A, BRODE R, CIMORELLI A, LEE R, PAINE R, PERRY S, PETERS W, WEIL J and WILSON R (2001) A complex terrain dispersion model for regular applications. *Atmospheric Environment*, **35**, pages 4211-4221.

APPENDIX A
Model source code

```
**************************************************************************************************************
        Written by:     T Tshukudu and JFC Friend
        Date:           May 2003
        Filename:       BCLVersion1.dpr
        Description:    Calculates concentrations at receptor points in Selebi-Phikwe
                        from BCL Limited smelter emissions.
**************************************************************************************************************
Unit BCLVersion1;
Interface
Uses
 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
 Dialogs, StdCtrls, DB, DBTables, Grids, DBGrids, ExtCtrls, Menus;
Type
 TfrmBCL = class(TForm)
  grpGrid: TGroupBox;
  edtSWX: TEdit;
  edtSWY: TEdit;
  edtNEX: TEdit;
  edtNEY: TEdit;
  edtInterval: TEdit;
  lblSW: TLabel;
  lblNE: TLabel;
  lblInterval: TLabel;
  grpStack: TGroupBox;
  edtDiameter: TEdit;
  edtHeight: TEdit;
  edtTemp: TEdit;
  edtWH: TEdit;
  lblDiameter: TLabel;
  lblHeight: TLabel;
  lblExitGas: TLabel;
  lblHw: TLabel;
  grpMetStations: TGroupBox;
  lblX: TLabel;
  lblY: TLabel;
  lblStation1: TLabel;
  lblStation2: TLabel;
  lblStation3: TLabel;
  edtX1: TEdit;
  edtY1: TEdit;
  edtX2: TEdit;
```

```
edtY2: TEdit;
edtX3: TEdit;
edtY3: TEdit;
grpSmelter: TGroupBox;
edtConc: TEdit;
edtPFCoal: TEdit;
edtLCoal: TEdit;
lblConc: TLabel;
lblPFC: TLabel;
lblLC: TLabel;
edtConcS: TEdit;
edtPFCoalS: TEdit;
edtLCoalS: TEdit;
lblFlow: TLabel;
lblSulphur: TLabel;
lblFurnace: TLabel;
lblConverter: TLabel;
lblFNA: TLabel;
edtFurnace: TEdit;
edtConverter: TEdit;
edtFNA: TEdit;
lblMX: TLabel;
lblMY: TLabel;
dlgInput: TOpenDialog;
dlgOutput: TOpenDialog;
grpDialog: TGroupBox;
lblstkstr: TLabel;
edtStkStr: TEdit;
lblAstn: TLabel;
lblMetString: TLabel;
lblPenStr: TLabel;
edtAstn: TEdit;
edtMetString: TEdit;
edtPenStr: TEdit;
dlgRun: TOpenDialog;
dlgAverage: TOpenDialog;
dlgTable: TOpenDialog;
grpMetFile: TGroupBox;
lblDate: TLabel;
lblTime: TLabel;
lblWS1: TLabel;
lblWD1: TLabel;
lblRad: TLabel;
lblT25: TLabel;
lblDT6: TLabel;
```

```
lblWD2: TLabel;
lblWS2: TLabel;
lblWD3: TLabel;
lblWS3: TLabel;
lblStart: TLabel;
lblLen: TLabel;
edtds: TEdit;
edtdl: TEdit;
edtts: TEdit;
edttl: TEdit;
edtws1: TEdit;
edtwl1: TEdit;
edtwd1: TEdit;
edtwdl1: TEdit;
edtrs: TEdit;
edtrl: TEdit;
edtTes: TEdit;
edttel: TEdit;
edtdts: TEdit;
edtdtl: TEdit;
edtwds2: TEdit;
edtwdl2: TEdit;
edtwss2: TEdit;
edtwsl2: TEdit;
edtwds3: TEdit;
edtwdl3: TEdit;
edtwss3: TEdit;
edtwsl3: TEdit;
grpControls: TGroupBox;
btnModel: TButton;
btnRunFile: TButton;
btnSave: TButton;
btnInput: TButton;
btnOutput: TButton;
btnAverage: TButton;
btnExit: TButton;
cmdRun: TButton;
Procedure Start(Sender: TObject);
Procedure Execute(Sender: TObject);
Procedure Load(Sender: TObject);
Procedure btnExitClick(Sender: TObject);
Procedure btnSaveClick(Sender: TObject);
Procedure OpenDatasubform(Sender: TObject);
Procedure btnRunClick(Sender: TObject);
Procedure btnAverageClick(Sender: TObject);
```

```
 Procedure FormActivate(Sender: TObject);
 Procedure btnModelClick(Sender: TObject);
 Procedure popAbout(Sender: TObject);
Private
 { Private declarations }
Public
 Q, D, Hs, Ts, Vi, Zo, He, Hmix, DH: Real;
 Stab: String;
 OrlSpeed, OrlWdir, T25, DT12, NRad, DT6: Real;
 SpSpeed, SpWdir, PltSpeed, PltWdir: Real;
 Ta, Xsourc, Ysourc, Xrec, Yrec, Zrec, X, Y: Real;
 Thedate: TDateTime;
 TheTime: TDateTime;
 RecDir, NWspeed, Nspeed: Real;
 Year, Month, Day, Hour, Min, Sec, MSec: Word;
 Xsw, Ysw, Xne, Yne, GridInt, Ycnt: Real;
 Nrec, I, RN: Integer;
 Cint, Qnew: Real;
 Nstn: Integer;
 Ka, Kb, Kc, Kd, Ke, Kf: Real;
 Aa, Ab, Ac, Ad, Ae, Af, Bsy: Real;
 Ca1, Da1, Fa1, Cb1, Db1, Fb1, Cc1, Dc1, Fc1, Cd1, Dd1, Fd1, Ce1, De1, Fe1, Cf1, Df1,
 Ff1: Real;
 Ca2, Da2, Fa2, Cb2, Db2, Fb2, Cc2, Dc2, Fc2, Cd2, Dd2, Fd2, Ce2, De2, Fe2, Cf2, Df2,
 Ff2: Real;
 StkStr: String;
 Tstart : Integer;
 end;

 Receptor = Array of Array of Real;
 BackConc = Array of Array of Real;
 CStation = Array of Array of Real;
 CAverage = Array of Array of Real;

Var

frmBCL: TfrmBCL;

 Function Wspeed(NWspeed, Hs,Zo,Ka,Kb,Kc,Kd,Ke,Kf: Real; Stab: String): Real;
 Function sigmay(X, Aa, ab, Ac, Ad, Ae, Af, Bsy: Real; Stab: String): Real;
 Function sigmaz(X,Ca1,Da1,Fa1,Cb1,Db1,Fb1,Cc1,Dc1,Fc1,Cd1,Dd1,Fd1,Ce1,De1,
                 Fe1, Cf1, Df1, Ff1, Ca2, Da2, Fa2, Cb2, Db2, Fb2, Cc2, Dc2, Fc2, Cd2,
                 Dd2, Fd2, Ce2, De2, Fe2, Cf2, Df2, Ff2: Real; Stab: String): Real;
 Function DStab(TowerSpeed, Nrad, DT6: Real; Hour: Word): String;
 Procedure DeltaH(Vi, D, Ts, Ta, xd, u:Real; Stab: String; var DH: Real);
 Procedure Distance(Xr, Yr, Xs, Ys, Wdir: Real; var X, Y: Real);
 Procedure Interpol(scr: CStation; Xrec, Yrec, Wdir1, Ws1, Wdir2, Ws2, Wdir3, Ws3: Real;
                    var RecDir, NWspeed: Real);
 Procedure XYGrid(var Xsw, Ysw, Xne, Yne, GridInt, Xrec, Yrec, Ycnt: Real);
 Procedure UserGrid(scr: Receptor; Nrec:Real; var I:Integer; var Xrec ,Yrec,
 Zrec: Real);
 Procedure BackGrd(scr: BackConc; var RN: Integer; Cint: Real);
 Procedure Penet(DH, Hs, NWspeed, Q: Real; var Qnew, He: Real);
 Procedure Rsourc(var Q, Hs, Ts, Vi, D, Xsourc, Ysourc: Real; var StkStr: String);
 Procedure Concen(Q, Y, Nspeed, sigmay1, sigmaz1, Zrec, He: Real; var C6: Real);
 Procedure Paramet(var Ka, Kb, Kc, Kd, Ke, Kf, Aa, Ab, Ac, Ad, Ae, Af, Bsy, Ca1, Da1, Fa1,
                   Cb1, Db1, Fb1, Cc1, Dc1, Fc1, Cd1, Dd1, Fd1, Ce1, De1, Fe1, Cf1,
                   Df1, Ff1,Ca2, Da2, Fa2, Cb2, Db2, Fb2, Cc2, Dc2, Fc2, Cd2, Dd2, Fd2,
                   Ce2, De2, Fe2, Cf2, Df2, Ff2: Real);

 Implementation

 Uses BCLsub1, BCLsub2, BCLscreen, BCLmodel;

 {$R *.dfm}

 Procedure TfrmBCL.Start(Sender: TObject);
 {*********************************************************************************************************
         THIS IS THE MAIN PROGRAM
         CALLING ALL THE SUBROUTINE
 *********************************************************************************************************}

 Var
 inFile: TextFile;

 outFile, avgFile: TextFile;
Year, Month, Day, Hour, Min, Sec, MSec: Word;
trial: string[255];
sigmaz1, sigmay1: Real;
C6, Ct, Cav: Real;
Co: BackConc; //Array for storing background concentrations
CB: Real;
tDate, tTime, MetString, PenStr: String;
Nstn, Nt, N, NewN, StartH, RS: Integer;
Nstation, Ic: Integer;
code: Integer;
Rec: Receptor; //Array for user defined receptors
Stn: CStation;
Avg: CAverage;
Astn: String;
label Exit;
label Proceed;
Label LineCont;
Label Nocalc;

 Begin

 //Opening the input and output files
 Application.MessageBox('CALCULATIONS STARTING', 'ABOUT...', MB_OK);

 if dlgOutput.FileName = '*.dat' then begin
 Application.MessageBox('Output File not selected','ABOUT...', MB_OK);
 goto Nocalc;
 end
 else
 AssignFile(outFile, dlgOutput.FileName);

 if dlgInput.Filename = '*.pm' then begin
 Application.MessageBox('Meteorological File not selected','ABOUT...', MB_OK);
 goto Nocalc;
 end
 else
 AssignFile(inFile, dlgInput.FileName);

 if dlgAverage.FileName = '*.tfv' then begin
 Application.MessageBox('Average File not selected','ABOUT...', MB_OK);
 goto Nocalc;
 end
 else
 AssignFile(avgFile, dlgAverage.FileName);

 reset(inFile);
 Rewrite(outFile);
 Rewrite(avgFile);
 //Source Information
 Rsourc(Q, Hs, Ts, Vi, D, Xsourc, Ysourc, StkStr);

 Zo := StrToFloat(edtWH.Text);      //Height for wind observation m
 Nstation := 0;
 Astn := edtAstn.Text;

 if (Astn = 'y') then begin
 Nstn:= StrToInt(InputBox('Input Box','Enter the number of met Stations', ''));
 Nstation := Nstn;
```

```
SetLength(Stn, Nstn, 3);

  for Nt := 0 to (Nstn - 1) do begin
  Val(InputBox('ENTER X', 'Enter the X coordinate of each station in Km',")
                    , Stn[Nt, 0], code);
  Val(InputBox('ENTER Y', 'Enter the Y coordinate of same station in Km',")
                    , Stn[Nt, 1], code);
  Val(InputBox('ENTER Z', 'Enter the Z coordinate of same station in m',")
                    , Stn[Nt, 2], code);
  end;
end
else if (Astn = 'n') then begin
Nstation := 3;
SetLength(Stn,3,2);

Stn[0,0] := StrToFloat(edtX1.Text);  //X coordinates for station 1
Stn[0,1] := StrToFloat(edtY1.Text);  //Y coordinates for station 1
Stn[1,0] := StrToFloat(edtX2.Text);  //X coordinates for station 2
Stn[1,1] := StrToFloat(edtY2.Text);  //Y coordinates for station 2
Stn[2,0] := StrToFloat(edtX3.Text);  //X coordinates for station 3
Stn[2,1] := StrToFloat(edtY3.Text);  //Y coordinates for station 3
end;
//Choosing between using the model defined grid and user defined grid
MetString := edtMetString.Text;

if (MetString = 'n') then begin
//Getting model defined corners of the grid from the main form
Xsw := StrToFloat(edtSWX.Text); //X coordinates of sw corner
Ysw := StrToFloat(edtSWY.Text); //y coordinates of sw corner
Xne := StrToFloat(edtNEX.Text); //x coordinates of ne corner
Yne := StrToFloat(edtNEY.Text); //y coordinates of ne corner
GridInt := StrToFloat(edtInterval.Text); //length of grid square
Zrec := 3; //z
Ycnt := Ysw;
end
else if (MetString = 'y') then begin
//Storing user defined receptors in a multi dimensional dynamic array
N:= StrToInt(InputBox('Input Box','Enter the number of receptor points', "));
SetLength(Rec, N, 3);
Nrec := N;

  for NewN := 0 to (N-1) do begin
  Val(InputBox('ENTER X', 'Enter the X coordinate of each receptor in Km',")
                    , Rec[NewN, 0], code);
  Val(InputBox('ENTER Y', 'Enter the Y coordinate of same receptor in Km',")
                  , Rec[NewN, 1], code);
   Val(InputBox('ENTER Z', 'Enter the Z coordinate of same receptor in m','')
                  , Rec[NewN, 2], code);
   end;
end;
StartH := 0;
SetLength(Co, 500, 2);  //Max number of receptor points is set at 500
SetLength(Avg, 500, 3); //MAx number of receptor points is set at 500
PenStr := edtPenStr.Text;

Writeln(outFile, 'Air Quality Dispersion Model For Emissions From BCL Limited Smelter');
Writeln(outFile, '**** The model uses the Gaussian  Plume Equation **************');
Writeln(outFile, '');
Writeln(outFile, '** MODEL OPTIONS  ');
Writeln(outFile, '                  A T INDICATES THAT THE OPTION HAS BEEN EXERCISED');
Writeln(outFile, '');

if (StkStr = 'n') then begin
Writeln(outFile, '********* USER DEFINED STACK COORDINATES   = F ');
end
else if (StkStr = 'y') then begin
Writeln(outFile, '********* USER DEFINED STACK COORDINATES   = T ');
end;

if (Astn = 'y') then begin
Writeln(outFile, '********* USER DEFINED STATION COORDINATES = T ');
end
else if (Astn = 'n') then begin
Writeln(outFile, '********* USER DEFINED STATION COORDINATES = F ');
end;

if (MetString = 'y') then begin
Writeln(outFile, '********* USER DEFINED RECEPTOR POINTS     = T ');
end
else if (MetString = 'n') then begin
Writeln(outFile, '********* USER DEFINED RECEPTOR POINTS     = F ');
end;

if (PenStr = 'y') then begin
Writeln(outFile, '********* PLUME PENETRATION               = T ');
end
else if (PenStr = 'n') then begin
Writeln(outFile, '********* PLUME PENETRATION               = F ');
end;
Writeln(outFile, '');
Writeln(outFile, '********* ANALYSIS OF THE MAIN STACK **************************');
Writeln(outFile, '');
Writeln(outFile, '*********** Main Stack Input Information *****************');
Writeln(outFile, '*** Source Strength (g/s)    ', ' ', Format('%10.2n', [Q]), '  ');
Writeln(outFile, '*** Stack Height (m)        ', ' ', Format('%10.2n', [Hs]), '  ');
Writeln(outFile, '*** Stack Gas Temperature (k)', ' ', Format('%10.2n', [Ts]),  '  ');
Writeln(outFile, '*** Stack Gas Velocity (m/s) ', ' ', Format('%10.2n', [Vi]),  '  ');
Writeln(outFile, '*** Stack Diameter (m)       ', ' ', Format('%10.2n', [D]),  '  ');
RS := 0;
while not(eof(inFile)) do begin

 //Reading parameters from the met input file
if (Nstation = 3) then begin
Readln(inFile, trial);
TheDate := StrToDate(Copy(trial, StrToint(edtds.Text), StrToint(edtdl.Text)));
tDate := Copy(trial, StrToint(edtds.Text), StrToint(edtdl.Text));
DecodeDate(TheDate, Year, Month, Day);
TheTime := StrToTime(Copy(trial, StrToint(edtts.Text), StrToint(edttl.Text)));
tTime := Copy(trial, StrToint(edtts.Text), StrToint(edttl.Text));
DecodeTime(TheTime, Hour, Min, Sec, MSec);
OrlSpeed := StrToFloat(Copy(trial, StrToint(edtws1.Text), StrToint(edtwl1.Text))); //Orlando
OrlWdir := StrToFloat(Copy(trial, StrToint(edtwd1.Text), StrToint(edtwdl1.Text))); //Orlando
Nrad :=StrToFloat(Copy(trial, StrToint(edtrs.Text), StrToint(edtrl.Text)));
T25 := StrToFloat(Copy(trial, StrToint(edttes.Text), StrToint(edttel.Text)));
DT6 := StrToFloat(Copy(trial, StrToint(edtdts.Text), StrToint(edtdtl.Text)));
SpWdir := StrToFloat(Copy(trial, StrToint(edtwds2.Text), StrToint(edtwdl2.Text)));  //SPSS
SpSpeed := StrToFloat(Copy(trial, StrToint(edtwss2.Text), StrToint(edtwsl2.Text))); //SPSS
PltWdir := StrToFloat(Copy(trial, StrToint(edtwds3.Text), StrToint(edtwdl3.Text)));
PltSpeed := StrToFloat(Copy(trial, StrToint(edtwss3.Text), StrToint(edtwsl3.Text)));
end

else if (Nstation < 3) then begin
Readln(inFile, trial);
TheDate := StrToDate(Copy(trial, StrToint(edtds.Text), StrToint(edtdl.Text)));
tDate := Copy(trial, StrToint(edtds.Text), StrToint(edtdl.Text));
DecodeDate(TheDate, Year, Month, Day);
TheTime := StrToTime(Copy(trial, StrToint(edtts.Text), StrToint(edttl.Text)));
tTime := Copy(trial, StrToint(edtts.Text), StrToint(edttl.Text));
DecodeTime(TheTime, Hour, Min, Sec, MSec);
OrlSpeed := StrToFloat(Copy(trial, StrToint(edtws1.Text), StrToint(edtwl1.Text))); //Orlando
OrlWdir := StrToFloat(Copy(trial, StrToint(edtwd1.Text), StrToint(edtwdl1.Text))); //Orlando
Nrad :=StrToFloat(Copy(trial, StrToint(edtrs.Text), StrToint(edtrl.Text)));
T25 := StrToFloat(Copy(trial, StrToint(edttes.Text), StrToint(edttel.Text)));
DT6 := StrToFloat(Copy(trial, StrToint(edtdts.Text), StrToint(edtdtl.Text)));
end;
```

```
RS := RS + 1;
//Modelling Grid Receptors i.e. model defined user grid
Xsw := StrToFloat(edtSWX.Text);
Ysw := StrToFloat(edtSWY.Text);
Xne := StrToFloat(edtNEX.Text);
Yne := StrToFloat(edtNEY.Text);
GridInt := StrToFloat(edtInterval.Text);
Ycnt := Ysw;
 //
Writeln(outFile, '');
Writeln(outFile, '************ MODEL OUTPUT ********************************');
Writeln(outFile, 'Average Concentration at Receptors for Simulation Period:',
 ' ', tDate, ' ', tTime);

//Call procedure Paramet to initialise all constants
 Paramet(Ka, Kb, Kc, Kd, Ke, Kf, Aa, Ab, Ac, Ad, Ae, Af, Bsy,Ca1,Da1,Fa1, Cb1,Db1, Fb1,
        Cc1, Dc1, Fc1, Cd1, Dd1, Fd1, Ce1, De1, Fe1, Cf1, Df1, Ff1, Ca2, Da2, Fa2,
        Cb2, Db2, Fb2, Cc2, Dc2, Fc2, Cd2, Dd2, Fd2, Ce2, De2, Fe2, Cf2, Df2, Ff2);

//Call function DStab to determine stability class
Stab := DStab(OrlSpeed, Nrad, DT6, Hour);  //Use OrlSpeed instead of SPSS

Writeln(outFile, '******** The Stability for this hour is Class:', ' ', Stab, ' ', '***********');
Writeln(outfile, '');
Writeln(outFile, 'X(km)', '    ', 'Y(km)', '    ', 'Z(m)','    ','Conc(µg/m3)');

//Parameter initialising
RN := 0;
I := 0;

repeat // looping for all receptors for each meteorological period
if (MetString = 'y') then begin
//Call procedure UserGrid to generate user defined receptors
UserGrid(Rec, Nrec, I , Xrec, Yrec, Zrec);
end

else if (MetString = 'n') then begin
//Call procedure XYGrid to generate X,Y coordinates of the receptors
XYGrid(Xsw, Ysw, Xne, Yne, GridInt, Xrec, Yrec, Ycnt);
Zrec := 3;
end;

if (Nstation < 3) then begin
RecDir := OrlWdir;
NWspeed := OrlSpeed;
```

```
 if NWspeed < 0.6 then begin
 NWspeed := 0.6;
 end;
end
else if (Nstation = 3) then begin
//Call procedure Interpolation to calculate wind speed and direction at receptor
Interpol(Stn, Xrec, Yrec, OrlWdir, OrlSpeed, SpWdir, SpSpeed, PltWdir, PltSpeed, RecDir,
        NWspeed);
end;

//Call procedure Distances to determine crosswind and downwind distances
Distance(Xrec,Yrec, Xsourc, Ysourc, RecDir, X, Y);

//Determing if the downwind distance is positive to proceed with calculations
//If its negative goto to the end of the loop to get the next metereological period
if X <= 0  then begin
C6 := 0;
goto Exit;
end;
 // Call function Wspeeed to deterine wind speed at stack height
   Nspeed := Wspeed(NWspeed, Hs, Zo, Ka, Kb, Kc, Kd, Ke, Kf, Stab);

//Call procedure DeltaH to calculate plume rise
DeltaH(Vi, D, Ts, T25, X, Nspeed, Stab, DH);
 He := Hs + DH; //effective stack height
//Call procedure penetration to calculate source strength and effecive plume height

 if (PenStr = 'y') then begin
 Penet(DH, Hs, NWspeed, Q, Qnew, He)
 end
 else if (PenStr = 'n')then begin
 goto LineCont;
 end;
 LineCont:

 sigmay1 :=sigmay(X, Aa, Ab, Ac, Ad, Ae, Af, Bsy, Stab);//call function sigmay

//call function sigmaz
Sigmaz1 := sigmaz(X,Ca1,Da1,Fa1,Cb1,Db1,Fb1,Cc1,Dc1,Fc1,Cd1,Dd1,Fd1,Ce1, De1,
                  Fe1, Cf1, f1, Ff1, Ca2, Da2, Fa2, Cb2, Db2, Fb2, Cc2, Dc2, Fc2, Cd2,
                  Dd2, Fd2, Ce2, De2, Fe2, Cf2, Df2, Ff2, Stab);

 if (PenStr = 'y') then begin
 Concen(Qnew, Y, Nspeed, sigmay1, sigmaz1, Zrec, He, C6);
  end
```

```
else if (PenStr = 'n')then begin
Concen(Q, Y, Nspeed, sigmay1, sigmaz1, Zrec, He, C6);
end;

Exit:

if StartH > 0 then begin
//BackGrd(Co, RN, Cint);
Cint := Co[RN,0];
if X <= 0  then begin
Ct := C6;
end
else
Ct := C6 + Cint;
end
else
Ct := C6;
Writeln(outFile, Format('%6.2n', [Xrec]), ' ', Format('%6.2n', [Yrec]), ' ', Format('%6.2n',
          [Zrec]),' ', Format('%8.2n', [Ct]));

if StartH = 0 then begin
//Storing concentration in an array
Co[RN,0] := C6;
Avg[RN,0] := Xrec;
Avg[RN,1] := Yrec;
Avg[RN,2] := Ct;
RN := RN + 1;
end
else if StartH > 0 then begin
Co[RN,1] := C6;
CB := Co[RN, 1];
Co[RN, 0] := CB;
Cav := Avg[RN,2] + Ct;
Avg[RN,2] := Cav;
RN := RN + 1;
end;

//Getting out of the loop for user defined receptors
if (MetString = 'y') then begin
if I = Nrec then begin
goto Proceed;
end;

end;
//
```

```
until (Yrec = Yne) and (Xrec = Xne);//getting out of the loop for model defined grid

Proceed:
StartH := StartH + 1;
end;

for Ic := 0 to RN-1 do begin
Xrec := Avg[Ic,0];
Yrec := Avg[Ic,1];
Cav := Avg[Ic,2]/RS;
writeln(avgFile, Format('%6.2n', [Xrec]), '  ', Format('%6.2n', [Yrec]), '  ', Format('%8.2n',
        [Cav]));
end;
CloseFile(inFile);
CloseFile(outFile);
CloseFile(avgFile);
Application.MessageBox('CALCULATIONS COMPLETE', 'ABOUT...', MB_OK);
Nocalc:
end;
       {*********************************************************************************************************
                                    END OF THE MAIN PROGRAM
       *********************************************************************************************************}

Procedure Parameet(var Ka, Kb, Kc, Kd, Ke, Kf, Aa, Ab, Ac, Ad, Ae, Af, Bsy, Ca1, Da1, Fa1,
                       Cb1, Db1, Fb1, Cc1, Dc1, Fc1, Cd1, Dd1, Fd1, Ce1, De1, Fe1, Cf1, Df1,
                       Ff1, Ca2, Da2, Fa2, Cb2, Db2, Fb2, Cc2, Dc2, Fc2, Cd2, Dd2, Fd2,
                       Ce2, De2, Fe2, Cf2, Df2, Ff2: Real);
{*********************************************************************************************************
                SUBROUTINE PARAMETERS
 Purpose:       Initialising all the model constants
 Programmer:    T Tshukudu and JFC Friend
 Date:           June 2003
 Outputs:       Constants parameters
 *********************************************************************************************************}
 Begin

 Ka := 0.07; //Wind speed exponent for class A stability
 Kb := 0.07; //Wind speed exponent for class B stability
 Kc := 0.10; //Wind speed exponent for class C stability
 Kd := 0.15; //Wind speed exponent for class D stability
 Ke := 0.35; //Wind speed exponent for class E stability
 Kf := 0.55; //Wind speed exponent for class F stability
 Aa := 213;  //Constant A for Sigma Y Class A stability
 Ab := 156;  //Constant A for Sigma Y Class A stability
```

```
Ac := 104;  //Constant A for Sigma Y Class A stability
Ad := 68;   //Constant A for Sigma Y Class A stability
Ae := 50.5;  //Constant A for Sigma Y Class A stability
Af := 34;    //Constant A for Sigma Y Class A stability
Bsy := 0.894; //Constant B for Sigma Y
//Sigma z constants for X < 1 km
Ca1 := 440.8;
Da1 := 1.941;
Fa1 := 9.27;
Cb1 := 106.6;
Db1 := 1.149;
Fb1 := 3.3;
Cc1 := 61.0;
Dc1 := 0.911;
Fc1 := 0;
Cd1 := 33.2;
Dd1 := 0.725;
Fd1 := -1.7;
Ce1 := 22.8;
De1 := 0.678;
Fe1 := -1.3;
Cf1 := 14.35;
Df1 := 0.740;
Ff1 := -0.35;
//
//Sigma z constants for X > 1 km
Ca2 := 459.7;
Da2 := 2.094;
Fa2 := -9.6;
Cb2 := 108.2;
Db2 := 1.098;
Fb2 := 2.0;
Cc2 := 61.0;
Dc2 := 0.911;
Fc2 := 0;
Cd2 := 44.5;
Dd2 := 0.516;
Fd2 := -13.0;
Ce2 := 55.4;
De2 := 0.305;
Fe2 := -34.0;
Cf2 := 62.6;
Df2 := 0.180;
Ff2 := -48.6;
//
```

```
end;
{*****************************************************************************************************}
Procedure Concen(Q, Y, Nspeed, sigmay1, sigmaz1, Zrec, He: Real; var C6: Real);
{*****************************************************************************************************
        SUBROUTINE CONCENTRATION
Purpose:        Determine the concentration at each receptor for each period
Programmer:     T Tshukudu and JFC Friend
Date:           June 2003
Outputs:        Concentration
*****************************************************************************************************}
Var
C1, C2, C3, C4, C5: Real;

Begin

C6:= 0;
C1 := Q/(2*Pi* sigmay1 * sigmaz1 * Nspeed);
C2 := Y*1000*Y*1000/(sigmay1 * sigmay1);
C3 := ((Zrec - He)*(Zrec - He)/(sigmaz1 * sigmaz1));
C4 := ((Zrec + He)*(Zrec + He)/(sigmaz1 * sigmaz1));
//C5 is the 10 minutes concentration
C5 := C1 * exp(-C2/2) * (exp(-C3/2) + exp(-C4/2))* 1000000;
//C6 is the 1 hour concentration c = Co * (t/to)^^0.5

if C5 > 0 then begin
//Adjusting 10 minutes concetration to 1 hour
C6 := exp(ln(C5) + (0.2 * ln(1/6)));
end
else
C6 := 0;

end;
{*****************************************************************************************************}

Procedure Rsourc(var Q, Hs, Ts, Vi, D, Xsourc, Ysourc: Real; var StkStr: String);
{*****************************************************************************************************
        SUBROUTINE SOURCE INFORMATION
Purpose :       The routine reads in the flow of material into the smelter to determine the
                gas emissions rate.  It also uses the gas flow flow from the smelter to
                determine exit gas velocity.  Other stack parameters are read as constants
Programmer:     T Tshukudu and JFC Friend
Date:           June 2003
Inputs:         Flow of material into the smelter; and
                Mass flow of gases from the smelter
Outputs : Main Stack Characteristics
****************************************************************************************************}
Var
ConcTrate, PFCoal, LCoal: Real;
SConc, SPFC, SLC: Real;
STot: Real;
FFG, CVG, FNG, Ftot, Dst, Density, NTs: Real;
SArea, Qf: Real;
code : Integer;

 //Getting parameters from the main form
Begin

Density := 0;
Q := 0;
Vi := 0;
ConcTrate := StrToFloat(frmBCL.edtConc.Text);
PFCoal := StrToFloat(frmBCL.edtPFCoal.Text);
LCoal := StrToFloat(frmBCL.edtLCoal.Text);
SConc := StrToFloat(frmBCL.edtConcS.Text);
SPFC := StrToFloat(frmBCL.edtPFCoalS.Text);
SLC := StrToFloat(frmBCL.edtLCoalS.Text);
STot := (ConcTrate * SConc/100) + (PFCoal * SPFC/100) + (LCoal * SLC/100);
//Emission rate or source strength in g/s
Q := 2 * STot * 0.91 * 1000000 / (365 * 24 * 3600);
// 9% of the sulphur inflow is left in the output material

D := StrToFloat(frmBCL.edtDiameter.Text);  //Stack diameter m
Hs := StrToFloat(frmBCL.edtHeight.Text);   //stack height m

Ts := StrToFloat(frmBCL.edtTemp.Text);    //Stack gas temperature K (296 C)
NTs := (Ts - 273) * 1.8 + 32; //Stack Gas Temperature in F
if NTs > 400 then begin
 if NTs < 500 then begin
 Dst := 0.0435 + ((0.0412 - 0.0435) / 100 * (NTs - 400));
 Density := Dst * 35.31 / 2.205; //Air Density at Stack Temperature kg/m3
 end
 else if NTs < 600 then begin
 Dst := 0.0412 + ((0.0373 - 0.0412)/100*(NTs - 500));
 Density := Dst * 35.31 / 2.205; //Air Density at Stack Temperature kg/m3
 end;
end;

FFG := StrToFloat(frmBCL.edtFurnace.Text);
CVG := StrToFloat(frmBCL.edtConverter.Text);
```

```
FNG := StrToFloat(frmBCL.edtFNA.Text);
Ftot := (FFG + CVG + FNG) * 1000;
Qf := (Ftot / Density) / (365 * 24 * 3600);  //Max flow is about 411 m3/s
SArea := Pi * (D * D / 4);
//Vi := 7.3;
Vi := Qf / SArea; //Stack gas velocity m/s
 StkStr := frmBCL.edtStkStr.Text;
//StkStr:= InputBox('Input Box','Is Stack located at the origin (y or n)', '');
if (StkStr = 'n') then begin
Xsourc := 0;
Ysourc := 0;
end
else if (StkStr = 'y') then begin
Val(InputBox('ENTER X', 'Enter the X coordinate of the source in Km',''))
          , Xsourc, code);
Val(InputBox('ENTER Y', 'Enter the Y coordinate of the source in Km',''))
          , Ysourc, code);
end;
 end;
{**************************************************************************************************}

 Procedure Penet(DH, Hs, NWspeed, Q: Real; var Qnew, He: Real);
{*************************************************************************************************
          SUBROUTINE PENETRATION
Purpose:        Calculate the plume material left in the mixing layer and the effective
                stack height
Programmer:     T Tshukudu and JFC Friend
Date:           June 2003
Inputs:         Plume Rise, DH
                Height of the mixing layer, Hmix
                Stack height and emission rate (source strength)
Outputs:        Effective plume height and new source strength
**************************************************************************************************}
Var
zi, Pn, nHe, Hmix: Real;

Begin

zi := 0;
Pn := 0;
nHe:= 0;
//Hmix = 320 * U10
Hmix := 320 * NWspeed;

if Hmix < Hs then begin
```

```
Pn := 0;
end
else if Hmix > Hs then begin
zi := Hmix - Hs;
Pn := (1.5*DH - zi) /(1.5*DH - (0.5*DH));
nHe := Hs + (((2/3) + (Pn/3))*zi);
end;

 if Pn <= 0 then begin
 Qnew := Q;
 Pn := 0;
 nHe := Hs + (((2/3) + (Pn/3))*zi);
 if nHe < He then begin
 He := nHe;
 end;
 end;
 If Pn > 0 then begin
 Qnew := Q*(1-Pn);
 if nHe < He then begin
 He := nHe;
 end;
 end;
end;
{*****************************************************************************************************}

Procedure BackGrd(scr: BackConc; var RN: Integer; Cint: Real);

Begin
//Cint := scr[RN, 0];
end;

Procedure UserGrid(scr: Receptor; Nrec: Real; var I: Integer; var Xrec, Yrec, Zrec: Real);
{******************************************************************************************************
          SUBROUTINE USERGRID
Purpose:        Generate receptor points from user defined coordinates
Programmer:     T Tshukudu and JFC Friend
Date:           June 2003
Inputs:         Receptor coordinates array
Outputs:        X,Y coordinates of the receptor
******************************************************************************************************}

 Begin

 Xrec := scr[I,0]; //Getting X coordinates from the array
 Yrec := scr[I,1]; //Getting Y coordinates from the array
```

```
Zrec := scr[I,2]; //Getting Z coordinates from the array
I := I + 1;      //Integer to move to the next array
end;
{*******************************************************************************************************}

 Procedure XYGrid(var Xsw, Ysw, Xne, Yne, GridInt, Xrec, Yrec, Ycnt: Real);
{*******************************************************************************************************
              SUBROUTINE XYGRID
Purpose:         Generate the receptor points
Programmer:      T Tshukudu and JFC Friend
Date:            May 2003
Inputs:          X,Y coordinates (in Km) of the South-West corner of Grid
                 X,Y coordinates (in Km) of the North-East corner of Grid
                 Grid Interval
Outputs:         X,Y coordinates of the receptor
*******************************************************************************************************}

Var
count: Real;

Begin

Xrec := Xsw;
Yrec := Ysw;
Ysw := Ysw + GridInt;
 if Yrec = Yne then begin
 Xsw := Xsw + GridInt;
 count := Yne - Ycnt + 1;
 Ysw := Ysw - count;
 end;
end;
{*******************************************************************************************************}


Procedure Interpol(scr: CStation; Xrec, Yrec, Wdir1, Ws1, Wdir2, Ws2, Wdir3, Ws3: Real;
                   var RecDir, NWspeed: Real);
{*******************************************************************************************************
              SUBROUTINE INTERPOLATION
Purpose:          Interpolate wind direction and wind speed at a receptor
Programmer:      T Tshukudu and JFC Friend
Date:            May 2003
Inputs:          X, Y coordinates of the stations
                 Wind speed and direction at stations
                 X, Y coordinates of the receptor
Outputs:         Wind speed and direction at a receptor
*************************************************************************************************}
Var
X1, X2, Y1, X3, Y2: Real;
RX1, RX2, RX3, RY1, RY2, RY3: Real;
r11, r12, r13: Real;
A1, B1, C1, D, E, F, XX: Real;
A2, B2, C2, A3, B3, C3: Real;
Xs, Ys: Real;
Wdirr: Real;
Xs1, Xs2, Xs3, Ys1, Ys2, Ys3: Real;

label StationDir;

Begin

RecDir := 0;
NWspeed := 0;
if Ws1 < 0.6 then begin
  Ws1 := 0.6;
 end;
if Ws2 < 0.6 then begin
  Ws2 := 0.6;
 end;
 if Ws3 < 0.6 then begin
  Ws3 := 0.6;
 end;

 Xs1 := scr[0,0];
 Ys1 := scr[0,1];
 Xs2 := scr[1,0];
 Ys2 := scr[1,1];
 Xs3 := scr[2,0];
 Ys3 := scr[2,1];

If Xrec = Xs1 then begin
 if Yrec = Ys1 then begin
  RecDir := Wdir1;
  NWspeed := Ws1;
  goto StationDir;
 end;
 end;

If Xrec = Xs2 then begin
 if Yrec = Ys2 then begin
  RecDir := Wdir2;
 NWspeed := Ws2;
 goto StationDir;
end;
end;
If Xrec = Xs3 then begin
if Yrec = Ys3 then begin
 RecDir := Wdir3;
 NWspeed := Ws3;
 goto StationDir;
end;
end;

 Wdir1 := Wdir1 / 180 * Pi;
 Wdir2 := Wdir2 / 180 * Pi;
 Wdir3 := Wdir3 / 180 * Pi;
RX1 := Ws1 * sin(Wdir1);
RY1 := Ws1 * cos(Wdir1);
RX2 := Ws2 * sin(Wdir2);
RY2 := Ws2 * cos(Wdir2);
RX3 := Ws3 * sin(Wdir3);
RY3 := Ws3 * cos(Wdir3);

  A1 := (Xrec - Xs1) * (Xrec - Xs1);
  B1 := (Yrec - Ys1) * (Yrec - Ys1);
  C1 := A1 + B1;
  r11 := exp((1/2)*ln(C1));
    A2 := (Xrec - Xs2) * (Xrec - Xs2);
    B2 := (Yrec - Ys2) * (Yrec - Ys2);
    C2 := A2 + B2;
    r12 := exp((1/2)*ln(C2));
      A3 := (Xrec - Xs3) * (Xrec - Xs3);
      B3 := (Yrec - Ys3) * (Yrec - Ys3);
      C3 := A3 + B3;
      r13 := exp((1/2)*ln(C3));

X1 := (RX1/(r11*r11)) + (RX2/(r12*r12)) + (RX3/(r13*r13));
X2 := (1/(r11*r11)) + (1/(r12*r12)) + (1/(r13*r13));
X3 := X1 / X2;
Y1 := (RY1/(r11*r11)) + (RY2/(r12*r12)) + (RY3/(r13*r13));
Y2 := Y1 / X2;
XX := X3 / Y2;
Wdirr := arctan(XX);

//RecDir := 270 - (2 * Wdirr * 180 / Pi);
if (X3 > 0) then begin
```

```
  if (Y2 > 0) then begin
   RecDir := Wdirr * 180 / Pi;
  end;
  if (Y2 < 0) then begin
   RecDir := 180 + (Wdirr * 180 / Pi);
  end;
 end;
 if (X3 < 0) then begin
  if (Y2 > 0) then begin
   RecDir := 360 + (Wdirr * 180 / Pi);
  end;
  if (Y2 < 0) then begin
   RecDir := 180 + (Wdirr * 180 / Pi);
 end;
 end;
 D := X3 * X3;
 E := Y2 * Y2;
 F := D + E;
 NWspeed := exp((1/2)*ln(F));
 StationDir:

end;
{********************************************************************************************************}

 Function DStab(TowerSpeed, Nrad, DT6: Real; Hour: Word): String;
{********************************************************************************************************
                    SUBROUTINE DSTAB
Purpose:            Determine the stability class for each meteorological period
Programmer:         T Tshukudu and JFC Friend
 Date:              May 2003
 Inputs:            Time of day, Wind speed, Net Radiation and Vertical
                    temperature difference
 Outputs:           Stability class
 ********************************************************************************************************}

Var
Theday : String;
label StabDet;

Begin

DStab := '';
If Hour >= 6 then begin
 if Hour <= 18 then begin
 Theday := 'DayTime';
 end
 else if Hour > 18 then begin
 TheDay := 'NightTime';
 end;
end
else if Hour < 6 then begin
TheDay := 'NightTime';
end;
 if (TheDay = 'DayTime') then begin
  //
  if TowerSpeed < 2 then begin
    if Nrad >= 675 then begin
    DStab := 'a';
    goto StabDet;
    end
    else if Nrad < 175 then begin
    DStab := 'd';
    goto StabDet;
    end
    else
    DStab := 'b';
    goto StabDet;
  end;
  //
  if TowerSpeed < 3 then begin
   if TowerSpeed >= 2 then begin
     If Nrad >= 925 then begin
     DStab := 'a';
     goto StabDet;
     end
     else if Nrad < 175 then begin
     DStab := 'd';
     goto StabDet;
     end
     else if Nrad >=175 then begin
       if Nrad >= 675 then begin
       DStab := 'b';
       goto StabDet;
       end
       else
       DStab := 'c';
       goto StabDet;

     end;
  end;
  end;
  //
 if TowerSpeed < 5 then begin
  if TowerSpeed >= 3 then begin
    if Nrad >= 675 then begin
    DStab := 'b';
    goto StabDet;
    end
    else if Nrad < 175 then begin
    DStab := 'd';
    goto StabDet;
    end
    else
    DStab := 'c';
    goto StabDet;

  end;
 end;
  //
if TowerSpeed < 6 then begin
  if TowerSpeed >= 5 then begin
    if Nrad >= 675 then begin
    DStab := 'c';
    goto StabDet;
    end
    else if Nrad < 675 then begin
    DStab := 'd';
    goto StabDet;
    end;
  end;
 end;
  //
 if TowerSpeed >= 6 then begin
   if Nrad >= 925 then begin
    DStab := 'c';
    goto StabDet;
    end
    else if Nrad < 925 then begin
    DStab := 'd';
    goto StabDet;
    end;
  end;
  //
end;
if (TheDay = 'NightTime') then begin
```

```
 //
 If TowerSpeed < 2 then begin
  if DT6 < 0.0 then begin
  DStab := 'e';
  goto StabDet;
  end
  else if DT6 >= 0.0 then begin
  DStab := 'f';
  goto StabDet;
  end;
 end;
 //
If TowerSpeed >= 2 then begin
 if TowerSpeed < 2.5 then begin
  if DT6 < 0.0 then begin
  DStab := 'd';
  goto StabDet;
  end
  else if DT6 >= 0.0 then begin
  DStab := 'e';
  goto StabDet;
  end;
 end
end;
 //
 if TowerSpeed >= 2.5 then begin
 DStab := 'd';
 goto StabDet;
 end;
 //
end;
StabDet:

end;
{**************************************************************************************************************}

  Procedure Distance(Xr, Yr, Xs, Ys, Wdir: Real; var X, Y: Real);
{**************************************************************************************************************
                    SUBROUTINE DISTANCES
Purpose:             Calculate crosswind and downwind distances
Programmer:          T Tshukudu and JFC Friend
 Date:               May 2003
 Inputs:             X,Y coordiantes of the source
                     X,Y coordinates of the receptor
                     Wind direction at the receptor
Outputs:        Downwind distance, X, and Crosswind distance, Y.
*****************************************************************************************************}

var
A1, A2, A3, A4: Real;

begin
// Xdist = -(Xr - Xs)sin(wdir) - (Yr - Ys)cos(wdir)
// Ydist = (Xr - Xs)cos(wdir) - (Yr - Ys)sin(wdir)
X := 0;
Y := 0;
A1 := Xr - Xs;
A2 := Yr - Ys;
Wdir := Wdir / 180 * Pi;
A3 := sin(Wdir);
A4 := cos(Wdir);
X := -(A1 * A3) - (A2 * A4);
Y := (A1 * A4) - (A2 * A3);
 end;
{****************************************************************************************************}

 Function Wspeed(NWspeed, Hs, Zo, Ka, Kb, Kc, Kd, Ke, Kf: Real; Stab: String): Real;
{*****************************************************************************************************
            SUBROUTINE WINDSPEED
Purpose:        Adjusts Wind Speed from Anemometer Height to Stack Height
Programmer:     T Tshukudu and JFC Friend
Date:           May 2003
Inputs:         Anemometer height
                Wind speed at anemometer height
                Stability class and stack height
Outputs:        Wind speed at stack height
*****************************************************************************************************}
Var
Z, base: Real;
IntSpeed: Real;

Begin

//u = exp[ln(Uo) + k*Ln(dz/dzo)]
IntSpeed := 0;

if (Stab = 'a') then begin
Z := Hs;
base := (Z / Zo);
IntSpeed := exp(ln(NWspeed) + (Ka * ln(base))); //new wind speed
```

```
 end
else if (Stab = 'b') then begin
 Z := Hs;
base := (Z / Zo);
IntSpeed := exp(ln(NWspeed) + (Kb * ln(base))); //new wind speed
 end
else if (Stab = 'c') then begin
 Z := Hs;
base := (Z / Zo);
IntSpeed := exp(ln(NWspeed) + (Kc * ln(base))); //new wind speed
 end
else if (Stab = 'd') then begin
 Z := Hs;
base := (Z / Zo);
IntSpeed := exp(ln(NWspeed) + (Kd * ln(base))); //new wind speed
 end
else if (Stab = 'e') then begin
Z := Hs;
base := (Z / Zo);
IntSpeed := exp(ln(NWspeed) + (Ke * ln(base))); //new wind speed
 end
else if (Stab = 'f') then begin
 Z := Hs;
base := (Z / Zo);
IntSpeed := exp(ln(NWspeed) + (Kf * ln(base))); //new wind speed
end;

//Do Not Allow Stack Height Wind Speed < 1.0 m/s
if IntSpeed < 1.0 then begin
 Wspeed := 1.0;
end
else
Wspeed := IntSpeed;

end;
{*********************************************************************************************************}

 Function sigmay(X, Aa, ab, Ac, Ad, Ae, Af, Bsy: Real; Stab: String): Real;
{*********************************************************************************************************
                SUBROUTINE SIGMAY
Purpose:        Calculate the horizontal dispersion parameter, sigmay
Programmer:     T Tshukudu and JFC Friend
Date:           May 2003
Inputs:         Downwind distance, X, and Stability class
Outputs:        sigmay
**************************************************************************************************}

Var
Nsigmay: Real;

Begin

//sigmay = a * X ** b where A and B are constants, X is the downwind distance in Km
Nsigmay := 0;

if (Stab = 'a') then begin
Nsigmay := exp(ln(Aa) + (Bsy * ln(X)));
 end
else if (Stab = 'b') then begin
Nsigmay := exp(ln(Ab) + (Bsy * ln(X)));
 end
else if (Stab = 'c') then begin
Nsigmay := exp(ln(Ac) + (Bsy * ln(X)));
 end
else if (Stab = 'd') then begin
Nsigmay := exp(ln(Ad) + (Bsy * ln(X)));
 end
else if (Stab = 'e') then begin
Nsigmay := exp(ln(Ae) + (Bsy * ln(X)));
 end
else if (Stab = 'f') then begin
Nsigmay := exp(ln(Af) + (Bsy * ln(X)));
 end;
sigmay := Nsigmay;
 end;
{**************************************************************************************************}

 Function sigmaz(X, Ca1, Da1, Fa1, Cb1, Db1, Fb1, Cc1, Dc1, Fc1, Cd1, Dd1, Fd1, Ce1,
                 De1, Fe1, Cf1, Df1, Ff1, Ca2, Da2, Fa2, Cb2, Db2, Fb2, Cc2, Dc2, Fc2,
                 Cd2, Dd2, Fd2, Ce2, De2, Fe2, Cf2, Df2, Ff2: Real; Stab: String): Real;
{**************************************************************************************************
                SUBROUTINE SIGMAZ
Purpose:        Calculate the vertical dispersion parameter, sigmaz
Programmer:     T Tshukudu and JFC Friend
Date:           May 2003
Inputs:         Downwind distance, X, and Stability class
Outputs:        sigmaz
**************************************************************************************************}
Var
Nsigmaz: Real;
```

Begin

```
//sigmaz = (c * X ** d) + f
Nsigmaz := 0;
if X < 1 then begin
if (Stab = 'a') then begin
 Nsigmaz := (exp(ln(Ca1) + (Da1 * ln(X)))) + Fa1;
 end
else if (Stab = 'b') then begin
 Nsigmaz := (exp(ln(Cb1) + (Db1 * ln(X)))) + Fb1;
 end
else if (Stab = 'c') then begin
 Nsigmaz := (exp(ln(Cc1) + (Dc1 * ln(X)))) + Fc1;
 end
else if (Stab = 'd') then begin
 Nsigmaz := (exp(ln(Cd1) + (Dd1 * ln(X)))) + Fd1;
 end
else if (Stab = 'e') then begin
 Nsigmaz := (exp(ln(Ce1) + (De1 * ln(X)))) + Fe1;
 end
else if (Stab = 'f') then begin
 Nsigmaz := (exp(ln(Cf1) + (Df1 * ln(X)))) + Ff1;
 end;

 end;

if X >= 1 then begin
if (Stab = 'a') then begin
Nsigmaz := (exp(ln(Ca2) + (Da2 * ln(X)))) + Fa2;
 end
else if (Stab = 'b') then begin
Nsigmaz := (exp(ln(Cb2) + (Db2 * ln(X)))) + Fb2;
 end
else if (Stab = 'c') then begin
 Nsigmaz := (exp(ln(Cc2) + (Dc2 * ln(X)))) + Fc2;
 end
else if (Stab = 'd') then begin
Nsigmaz := (exp(ln(Cd2) + (Dd2 * ln(X)))) + Fd2;
 end
else if (Stab = 'e') then begin
Nsigmaz := (exp(ln(Ce2) + (De2 * ln(X)))) + Fe2;
 end
else if (Stab = 'f') then begin
Nsigmaz := (exp(ln(Cf2) + (Df2 * ln(X)))) + Ff2;
 end;
```

```
end;
//Do not allow sigmaz to be more than 5 km
if Nsigmaz > 5000 then begin
Nsigmaz := 5000;
end;
sigmaz := Nsigmaz;
end;
{*******************************************************************************************************}

Procedure DeltaH(Vi, D, Ts, Ta, xd, u:Real; Stab: String; var DH: Real);
{*******************************************************************************************************
                    SUBROUTINE DELTAH
Purpose:        Calculation of the Plume Rise as a function of downwind distance
Programmer:     T Tshukudu and JFC Friend
Date:           May 2003
Inputs:         Stack exit gas velocity, Diameter, and exit gas temperature
                Downwind distance, wind speed at stack height and Stability class
Outputs:        Plume rise, deltaH
*******************************************************************************************************}

Const g = 9.8;
Var
 Fb, Xs, Xf, R: Real;
 F1, X1, S: Real;
 X2, DH1: Real;
 Answer: String;

 Begin

 //For Neutral or Unstable conditions (A, B, C, D)
 //DH = 1.6 * Fb**(1/3) * Xf**(2/3) / u for x >= xf
 //DH = 1.6 * Fb**(1/3) * X**(2/3) / u for x < xf
 //Xs = 14 * Fb**(5/8) for Fb < 55
 //Xs = 34 * Fb**(2/5) for Fb >= 55
 //Fb = g * Vi * R**2 * (Ts - Ta) / Ts
 //For Stable conditions
 //DH = 2.6 * (Fb / (u S))**(1/3) for x > xf
 //DH = 1.6 * Fb**(1/3) * X**(2/3) / u for x < xf
 //S = g / Ta * (DT / DZ)
 //DT/Dz = 0.02 for E and 0.035 for F
 DH := 0;
 R := D / 2;
 Fb := (g*Vi*R*R*(Ts - (Ta + 273))) / Ts;
 if (Stab = 'a') then begin
 Answer := 'y';
```

```
end
else if (Stab = 'b') then begin
Answer := 'y';
end
else if (Stab = 'c') then begin
Answer := 'y';
end
else if (Stab = 'd') then begin
Answer := 'y';
end;

if (Answer = 'y') then begin
 if Fb < 55 then begin
  Xs := exp(ln(14) + ((5/8)*ln(Fb)));
   Xf := 3.5 * Xs;
    if xd * 1000 >= Xf then begin
    F1 := exp((1/3)*ln(Fb));
    X1 := exp((2/3)*ln(Xf));
    DH := (1.6*F1*X1) / u
    end
    else if xd * 1000 < Xf then begin
    F1 := exp((1/3)*ln(Fb));
    X1 := exp((2/3)*ln(xd * 1000));
    X2 := exp((2/3)*ln(Xf));
    DH1 := (1.6*F1*X2) / u;
    DH := (1.6*F1*X1) / u;
     if DH1 < DH then begin
     DH := DH1;
     end;
    end;
 end
 else if Fb >= 55 then begin
  Xs := exp(ln(34) + (0.4*ln(Fb)));
   Xf := 3.5 * Xs;
   if xd * 1000 >= Xf then begin
   F1 := exp((1/3)*ln(Fb));
   X1 := exp((2/3)*ln(Xf));
   DH := (1.6*F1*X1) / u;
   end
   else if xd * 1000 < Xf then begin
   F1 := exp((1/3)*ln(Fb));
   X1 := exp((2/3)*ln(xd * 1000));
   X2 := exp((2/3)*ln(Xf));
   DH1 := (1.6*F1*X2) / u;
   DH := (1.6*F1*X1) / u ;
```

```
      if DH1 < DH then begin
      DH := DH1;
      end;
     end;
   end;
  end
  else if (Stab = 'e') then begin
   X1 := 0.02;
   S := g * X1 / (Ta + 273);
   Xf := 2.0715 * u /(sqr(S));
    if xd * 1000 < Xf then begin
    F1 := exp((1/3)*ln(Fb));
    X1 := exp((2/3)*ln(xd * 1000));
    DH := (1.6*F1*X1) / u ;
    end
    else if xd * 1000 >= Xf then begin
    F1 := Fb /(u*S);
    DH := exp(ln(2.6) + ((1/3)*ln(F1)));
    end;
  end
  else if (Stab = 'f') then begin
   X1 := 0.035;
   S := g * X1 / (Ta + 273);
   Xf := 2.0715 * u /(sqr(S));
    if xd * 1000 < Xf then begin
    F1 := exp((1/3)*ln(Fb));
    X1 := exp((2/3)*ln(xd * 1000));
    DH := (1.6*F1*X1) / u;
    end
    else if xd * 1000 >= Xf then begin
    F1 := Fb /(u*S);
    DH := exp(ln(2.6) + ((1/3)*ln(F1)));
    end;
  end;

 end;
{****************************************************************************************************}
  Procedure TfrmBCL.Execute(Sender: TObject);
*****************************************************************************************************

  Purpose:        Opening the input file
  Programmer:     T Tshukudu and JFC Friend
  Date:           August 2003
*****************************************************************************************************

  Begin
```

```
dlgInput.Execute;
If dlgInput.FileName = '*.pm' then begin
Application.MessageBox('Please select the meteorological file',
'ABOUT...', MB_OK);
end;
end;
****************************************************************************************************

Procedure TfrmBCL.Load(Sender: TObject);
****************************************************************************************************
Purpose:        Opening the output file
 Programmer:    T Tshukudu and JFC Friend
 Date:          August 2003
****************************************************************************************************
Begin

dlgOutput.Execute;
If dlgOutput.FileName = '*.dat' then begin
Application.MessageBox('Please select the output file','ABOUT...', MB_OK);
end;
end;
****************************************************************************************************

Procedure TfrmBCL.btnExitClick(Sender: TObject);
****************************************************************************************************
Purpose:        Closing the program
 Programmer:    T Tshukudu and JFC Friend
 Date:          August 2003
****************************************************************************************************

Begin

close;
end;
****************************************************************************************************

Procedure TfrmBCL.btnSaveClick(Sender: TObject);
****************************************************************************************************
Purpose:        saving to the run file
 Programmer:    T. Tshukudu and JFC Friend
 Date:          August 2003
****************************************************************************************************
Var
RFile: TextFile;
r1,r2,r3,r4,r5,r6,r7,r8,r9: String;
```

```
s1, s2, s3, s4: String;
g1,g2,g3,g4,g5: String;
m1,m2,m3,m4,m5,m6: String;
d1,d2,d3,d4: String;
f1,f2,f3,f4,f5,f6,f7,f8,f9,f10: String;
f11,f12,f13,f14,f15,f16,f17,f18: String;
f19,f20,f21,f22: String;
Label Nosave;

Begin

dlgRun.Execute;
if dlgRun.FileName = '*.tfa' then begin
Application.MessageBox('Run file not saved','ABOUT...', MB_OK);
goto Nosave;
end
else
AssignFile(RFile, dlgRun.FileName);
Rewrite(RFile);

r1 := frmBCL.edtConc.Text;
r2 := frmBCL.edtPFCoal.Text;
r3 := frmBCL.edtLCoal.Text;
r4 := frmBCL.edtConcS.Text;
r5 := frmBCL.edtPFCoalS.Text;
r6 := frmBCL.edtLCoalS.Text;
r7 := frmBCL.edtFurnace.Text;
r8 := frmBCL.edtConverter.Text;
r9 := frmBCL.edtFNA.Text;
s1 := frmBCL.edtDiameter.Text;
s2 := frmBCL.edtHeight.Text;
s3 := frmBCL.edtTemp.Text;
s4 := frmBCL.edtWH.Text;
g1 := frmBCL.edtSWX.Text;
g2 := frmBCL.edtSWY.Text;
g3 := frmBCL.edtNEX.Text;
g4 := frmBCL.edtNEY.Text;
g5 := frmBCL.edtInterval.Text;
m1 := frmBCL.edtX1.Text;
m2 := frmBCL.edtY1.Text;
m3 := frmBCL.edtX2.Text;
m4 := frmBCL.edtY2.Text;
m5 := frmBCL.edtX3.Text;
m6 := frmBCL.edtY3.Text;
d1 := frmBCL.edtStkStr.Text;
```

```
d2 := frmBCL.edtAstn.Text;
d3 := frmBCL.edtMetString.Text;
d4 := frmBCL.edtPenStr.Text;
f1 := frmBCL.edtds.Text;
f2 := frmBCL.edtdl.Text;
f3 := frmBCL.edtts.Text;
f4 := frmBCL.edttl.Text;
f5 := frmBCL.edtws1.Text;
f6 := frmBCL.edtwl1.Text;
f7 := frmBCL.edtwd1.Text;
f8 := frmBCL.edtwdl1.Text;
f9 := frmBCL.edtrs.Text;
f10 := frmBCL.edtrl.Text;
f11 := frmBCL.edtTes.Text;
f12 := frmBCL.edttel.Text;
f13 := frmBCL.edtdts.Text;
f14 := frmBCL.edtdtl.Text;
f15 := frmBCL.edtwds2.Text;
f16 := frmBCL.edtwdl2.Text;
f17 := frmBCL.edtwss2.Text;
f18 := frmBCL.edtwsl2.Text;
f19 := frmBCL.edtwds3.Text;
f20 := frmBCL.edtwdl3.Text;
f21 := frmBCL.edtwss3.Text;
f22 := frmBCL.edtwsl3.Text;
writeln(RFile, r1);
writeln(RFile, r2);
writeln(RFile, r3);
writeln(RFile, r4);
writeln(RFile, r5);
writeln(RFile, r6);
writeln(RFile, r7);
writeln(RFile, r8);
writeln(RFile, r9);
writeln(RFile, s1);
writeln(RFile, s2);
writeln(RFile, s3);
writeln(RFile, s4);
writeln(RFile, g1);
writeln(RFile, g2);
writeln(RFile, g3);
writeln(RFile, g4);
writeln(RFile, g5);
writeln(RFile, m1);
writeln(RFile, m2);
writeln(RFile, m3);
writeln(RFile, m4);
writeln(RFile, m5);
writeln(RFile, m6);
writeln(RFile, d1);
writeln(RFile, d2);
writeln(RFile, d3);
writeln(RFile, d4);
writeln(RFile, f1);
writeln(RFile, f2);
writeln(RFile, f3);
writeln(RFile, f4);
writeln(RFile, f5);
writeln(RFile, f6);
writeln(RFile, f7);
writeln(RFile, f8);
writeln(RFile, f9);
writeln(RFile, f10);
writeln(RFile, f11);
writeln(RFile, f12);
writeln(RFile, f13);
writeln(RFile, f14);
writeln(RFile, f15);
writeln(RFile, f16);
writeln(RFile, f17);
writeln(RFile, f18);
writeln(RFile, f19);
writeln(RFile, f20);
writeln(RFile, f21);
writeln(RFile, f22);
closeFile(RFile);
Nosave:
end;
***************************************************************************************************************

procedure TfrmBCL.OpenDatasubform(Sender: TObject);
***************************************************************************************************************
Purpose:       Opening the Average file
  Programmer:  T Tshukudu and JFC Friend
  Date:        August 2003
***************************************************************************************************************
Var
File1: TextFile;
```

```
Begin

frmData.visible := True;
frmData.SetFocus;
AssignFile(File1, dlgAverage.FileName);

end;
*********************************************************************************************************

Procedure TfrmBCL.btnRunClick(Sender: TObject);
*********************************************************************************************************
Purpose:        Opening the input file
 Programmer:    T Tshukudu and JFC Friend
 Date:          August 2003
*********************************************************************************************************
Begin

frmRunFile.Visible := True;
frmRunFile.SetFocus;
end;
*********************************************************************************************************

Procedure TfrmBCL.btnAverageClick(Sender: TObject);
*********************************************************************************************************
Purpose:        Opening the Average file
 Programmer:    T Tshukudu and JFC Friend
 Date:          August 2003
*********************************************************************************************************
Begin

dlgAverage.Execute;
If dlgAverage.FileName = '*.tfv' then begin
Application.MessageBox('Please select the Average file','ABOUT...', MB_OK);
end;
end;
*********************************************************************************************************

Procedure TfrmBCL.FormActivate(Sender: TObject);
*********************************************************************************************************
Purpose:        Showing the splash screen
Programmer:     T Tshukudu and JFC Friend
Date:           October 2003
*********************************************************************************************************
Begin
```

```
 If Tstart <> 1 then begin
 frmScreen.Show ;
 end;

 end;
*************************************************************************************************************

 Procedure TfrmBCL.btnModelClick(Sender: TObject);

 Begin

 frmModel.Visible := true;
 frmModel.SetFocus;
 end;
*************************************************************************************************************

 Procedure TfrmBCL.popAbout(Sender: TObject);

 Begin

 frmModel.Visible := true;
 frmModel.SetFocus;
 end;
*************************************************************************************************************

end.
********************************************************************************************************************
                Written by:     T Tshukudu and JFC Friend
                Date:           May 2003
                Filename:       BCLmodel.dpr
                Description:    Displays information about the EEGAIR-BCL1 model
********************************************************************************************************************
Unit BCLmodel;
Interface
Uses
 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
 Dialogs, StdCtrls;
Type
 TfrmModel = class(TForm)
  memModel: TMemo;
  btnOk: TButton;
  procedure btnOkClick(Sender: TObject);
 Private
  { Private declarations }

 Public
  { Public declarations }
 end;

Var
 frmModel: TfrmModel;

Implementation

Uses BCLVersion1;
{$R *.dfm}

Procedure TfrmModel.btnOkClick(Sender: TObject);

Begin
close;
end;
```

****

Written by:	T Tshukudu and JFC Friend
Date:	May 2003
Filename:	BCLsub2.dpr
Description:	Load the run file

****

```
Unit BCLsub2;
Interface
Uses
 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
 Dialogs, StdCtrls;

Type
 TfrmRunFile = class(TForm)
  btnNew: TButton;
  btnLoad: TButton;
  dlgRunFile: TOpenDialog;
  procedure btnNewClick(Sender: TObject);
  procedure btnLoadClick(Sender: TObject);

 Private
  { Private declarations }
 Public
  { Public declarations }
 end;

Var
 frmRunFile: TfrmRunFile;

Implementation

Uses BCLVersion1 ;
 {$R *.dfm}

Procedure TfrmRunFile.btnNewClick(Sender: TObject);

Begin

 frmBCL.SetFocus;
 frmBCL.edtConc.Text:= '';
 frmBCL.edtPFCoal.Text := '';
 frmBCL.edtLCoal.Text := '';
 frmBCL.edtConcS.Text := '';
 frmBCL.edtPFCoalS.Text := '';
 frmBCL.edtLCoalS.Text := '';
 frmBCL.edtFurnace.Text := '';
 frmBCL.edtConverter.Text := '';
 frmBCL.edtFNA.Text := '';
 frmBCL.edtDiameter.Text := '';
 frmBCL.edtHeight.Text := '';
 frmBCL.edtTemp.Text := '';
 frmBCL.edtWH.Text := '';
 frmBCL.edtSWX.Text := '';
 frmBCL.edtSWY.Text := '';
 frmBCL.edtNEX.Text := '';
 frmBCL.edtNEY.Text := '';
 frmBCL.edtInterval.Text := '';
 frmBCL.edtX1.Text := '';
 frmBCL.edtY1.Text := '';
 frmBCL.edtX2.Text := '';
 frmBCL.edtY2.Text := '';
 frmBCL.edtX3.Text := '';
 frmBCL.edtY3.Text := '';
 frmBCL.edtStkStr.Text := '';
 frmBCL.edtAstn.Text := '';
 frmBCL.edtMetString.Text := '';
 frmBCL.edtPenStr.Text := '';
```

```
  frmBCL.edtds.Text := '';
  frmBCL.edtdl.Text := '';
  frmBCL.edtts.Text := '';
  frmBCL.edttl.Text := '';
  frmBCL.edtws1.Text := '';
  frmBCL.edtwl1.Text := '';
  frmBCL.edtwd1.Text := '';
  frmBCL.edtwdl1.Text := '';
  frmBCL.edtrs.Text := '';
  frmBCL.edtrl.Text := '';
  frmBCL.edtTes.Text := '';
  frmBCL.edttel.Text := '';
  frmBCL.edtdts.Text := '';
  frmBCL.edtdtl.Text := '';
  frmBCL.edtwds2.Text := '';
  frmBCL.edtwdl2.Text := '';
  frmBCL.edtwss2.Text := '';
  frmBCL.edtwsl2.Text := '';
  frmBCL.edtwds3.Text := '';
  frmBCL.edtwdl3.Text := '';
  frmBCL.edtwss3.Text := '';
  frmBCL.edtwsl3.Text := '';

end;

Procedure TfrmRunFile.btnLoadClick(Sender: TObject);
  Var
  RFile: TextFile;
  r1,r2,r3,r4,r5,r6,r7,r8,r9: String;
  s1,s2,s3,s4: String;
  g1,g2,g3,g4,g5: String;
  m1,m2,m3,m4,m5,m6: String;
  d1,d2,d3,d4: String;
  f1,f2,f3,f4,f5,f6,f7,f8,f9,f10: String;
  f11,f12,f13,f14,f15,f16,f17,f18: String;
  f19,f20,f21,f22: String;
  label EndCalc;

  Begin

  frmBCL.SetFocus;
  dlgRunFile.Execute;
  if dlgRunFile.FileName = '*.tfa' then begin
  Application.MessageBox('Please select the run file','ABOUT...', MB_OK);
  goto EndCalc;
```

```
end
else
AssignFile(RFile, dlgRunFile.FileName);
Reset(RFile);
Readln(RFile, r1);
frmBCL.edtConc.Text := r1;
Readln(RFile, r2);
frmBCL.edtPFCoal.Text:= r2 ;
Readln(RFile, r3);
frmBCL.edtLCoal.Text := r3;
Readln(RFile, r4);
frmBCL.edtConcS.Text := r4;
Readln(RFile, r5);
frmBCL.edtPFCoalS.Text := r5;
Readln(RFile, r6);
frmBCL.edtLCoalS.Text := r6;
Readln(RFile, r7);
frmBCL.edtFurnace.Text := r7;
Readln(RFile, r8);
frmBCL.edtConverter.Text := r8;
Readln(RFile, r9);
frmBCL.edtFNA.Text := r9;
Readln(RFile, s1);
frmBCL.edtDiameter.Text := s1;
Readln(RFile, s2);
frmBCL.edtHeight.Text := s2;
Readln(RFile, s3);
frmBCL.edtTemp.Text := s3;
Readln(RFile, s4);
frmBCL.edtWH.Text := s4;
Readln(RFile, g1);
frmBCL.edtSWX.Text := g1;
Readln(RFile, g2);
frmBCL.edtSWY.Text := g2;
Readln(RFile, g3);
frmBCL.edtNEX.Text := g3;
Readln(RFile, g4);
frmBCL.edtNEY.Text := g4;
Readln(RFile, g5);
frmBCL.edtInterval.Text := g5;
Readln(RFile, m1);
frmBCL.edtX1.Text := m1;
Readln(RFile, m2);
frmBCL.edtY1.Text := m2;
```

```
Readln(RFile, m3);
frmBCL.edtX2.Text := m3;
Readln(RFile, m4);
frmBCL.edtY2.Text := m4;
Readln(RFile, m5);
frmBCL.edtX3.Text := m5;
Readln(RFile, m6);
frmBCL.edtY3.Text := m6;
Readln(RFile, d1);
frmBCL.edtStkStr.Text := d1;
Readln(RFile, d2);
frmBCL.edtAstn.Text := d2;
Readln(RFile, d3);
frmBCL.edtMetString.Text := d3;
Readln(RFile, d4);
frmBCL.edtPenStr.Text := d4;
Readln(RFile, f1);
frmBCL.edtds.Text := f1;
Readln(RFile, f2);
frmBCL.edtdl.Text := f2;
Readln(RFile, f3);
frmBCL.edtts.Text := f3;
Readln(RFile, f4);
frmBCL.edttl.Text := f4;
Readln(RFile, f5);
frmBCL.edtws1.Text := f5;
Readln(RFile, f6);
frmBCL.edtwl1.Text := f6;
Readln(RFile, f7);
frmBCL.edtwd1.Text := f7;
Readln(RFile, f8);
frmBCL.edtwdl1.Text := f8;
Readln(RFile, f9);
frmBCL.edtrs.Text := f9;
Readln(RFile, f10);
frmBCL.edtrl.Text := f10;
Readln(RFile, f11);
frmBCL.edtTes.Text := f11;
Readln(RFile, f12);
frmBCL.edttel.Text := f12;
Readln(RFile, f13);
frmBCL.edtdts.Text := f13;
Readln(RFile, f14);
frmBCL.edtdtl.Text := f14;
Readln(RFile, f15);
```

```
 frmBCL.edtwds2.Text := f15;
 Readln(RFile, f16);
 frmBCL.edtwdl2.Text := f16;
 Readln(RFile, f17);
 frmBCL.edtwss2.Text := f17;
 Readln(RFile, f18);
 frmBCL.edtwsl2.Text := f18;
 Readln(RFile, f19);
 frmBCL.edtwds3.Text := f19;
 Readln(RFile, f20);
 frmBCL.edtwdl3.Text := f20;
 Readln(RFile, f21);
 frmBCL.edtwss3.Text := f21;
 Readln(RFile, f22);
 frmBCL.edtwsl3.Text := f22;
 closefile(RFile);
 EndCalc:
end;

end.
********************************************************************************************************************
              Written by:       T Tshukudu and JFC Friend
              Date:             October 2003
              Filename:         BCLscreen.dpr
              Description:      displays the splash screen for 5 seconds
********************************************************************************************************************
Unit BCLscreen;
 Interface
 Uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls;
 Type
 TfrmScreen = class(TForm)
  memScreen: TMemo;
  tmrScreen: TTimer;
  procedure Timer1(Sender: TObject);
 Private
  { Private declarations }
 Public
  { Public declarations }
 end;

 Var
 frmScreen: TfrmScreen;
```

```
Implementation

Uses BCLVersion1;
{$R *.dfm}

Procedure TfrmScreen.Timer1(Sender: TObject);

Begin

frmBCL.Tstart := 1;
Close ;
end;
end.
```

****