Applied Mathematical Modelling 33 (2009) 1957–1969

Contents lists available at ScienceDirect

Applied Mathematical Modelling

journal homepage: www.elsevier.com/locate/apm

Air pollution control with semi-infinite programming

A. Ismael F. Vaz [a,*], Eugénio C. Ferreira [b]

[a] Department of Production and Systems, Engineering School, University of Minho, Campus of Gualtar, 4710-057 Braga, Portugal
[b] IBB-Institute for Biotechnology and Bioengineering, Centre of Biological Engineering, University of Minho, Campus of Gualtar, 4710-057 Braga, Portugal

ARTICLE INFO

Article history:
Received 28 July 2007
Received in revised form 30 April 2008
Accepted 1 May 2008
Available online 9 May 2008

Keywords:
Air pollution control
Gaussian dispersion model
Semi-infinite programming
SIPAMPL database
NSIPS solver

ABSTRACT

Environment issues are more than ever important in a modern society. Complying with stricter legal thresholds on pollution emissions raises an important economic issue. This paper presents some ideas in the use of optimization tools to help in the planning and control of stationary pollution sources.

Three main semi-infinite programming approaches are described. The first consists in optimizing an objective function while the pollution level in a given region is kept bellow a given threshold. In the second approach the maximum pollution level in a given region is computed and in the third an air pollution abatement problem is considered. These formulations allow to obtain the best control parameters and the maxima pollution positions, where the sampling stations should be placed.

A specific modeling language was used to code four academic problems. Numerical results computed with a semi-infinite programming solver are shown.

1. Introduction

Environment issues in general and atmospheric pollution in particular have deserved particular attention by decision makers either in planning and control.

Atmospherical dispersion models are programs that use mathematical algorithms to simulate the pollutants dispersion in the atmospherical environment and in some cases how they can chemically react in the atmosphere. Dispersion models can be used to estimate and predict the atmospherical pollutants concentration emitted from sources as industrial plants or vehicle traffic.

Such kind of models are important for governmental agencies in order to preserve and check the air quality. Models are typically used to insure that existent or future industrial plants fulfill legal thresholds imposed by law.

These models are also important in defining the best decision politics in order to fulfill the legal threshold while the economic impact is minimized.

Models can also be used in design of strategies for effective control on reducing the hazardous atmospheric pollutants.

There are several available dispersion models that can easily be obtained from the Internet (see, for example the EPA – Environment Protection Agency – website in [1]). Dispersion models can be classified in four major classes, according to the used methodology [2]. *Box models* are based on the mass conservation principle. The scenario is considered as a box and the air mass inside it is considered uniformly distributed. *Gaussian models* are based on a Gaussian distribution of the plume. The normal distribution on the plume depends on the vertical (σ_y) and on the horizontal (σ_z) standard deviations. The width of the plume is therefore determined by the σ_y and σ_z, which are defined according to stability classes or travel time from the source. *Lagrangian/Eularian models*, as in box models, consider a region of air containing an initial concentration of pollutants.

* Corresponding author. Tel.: +351 253604740; fax: +351 253604741.
E-mail addresses: aivaz@dps.uminho.pt (A. Ismael F. Vaz), ecferreira@deb.uminho.pt (E.C. Ferreira).

0307-904X/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.apm.2008.05.008

These models consider changes in concentration due to mean fluid velocity, turbulence of wind components and molecular diffusion. *Computational fluid dynamic models* provide a complex analysis of fluid flow based on the conservation of mass and momentum by solving the Navier–Stokes equation.

Dispersion models take as input some of the following data:

- Atmospheric conditions such as wind velocity and direction, turbulence (characterized by its stability class) and environment temperature;
- Emission parameters such as emission source position and height, source internal diameter, gas out temperature and emission rate;
- Land elevations, localization and dimensions of any obstacles in the region to be considered.

This work deals with air pollution control of stationary sources. The resulting optimization problems belong to a well known class of optimization problems denominated as semi-infinite programming problems [3–6]. A problem is considered as semi-infinite when either the number of variables or the number of constraints is finite (but not both). We postpone a formal definition until Section 3.

While the proposed approach for optimal air pollution control could be used with any of the described air dispersion models, we will focus our attention on the Gaussian model, since it is the most widely used. Gaussian model is the core of almost all regulatory dispersion models. In its simple version, the Gaussian model relies on several mathematical equations that can easily be coded in a mathematical language.

In this paper a simple Gaussian model is considered with the purpose of illustrating the proposed approach to air pollution control. The model is used to provide estimates of pollution in a region where mean weather conditions are assumed (see [7]). One of the proposed problems consists of optimizing an objective function (minimum stack height) while the air pollution is kept bellow a given threshold. Other proposed problem consists in computing the maximum air pollution attained in a given region. Finally an air pollution abatement problem where reduction in the air pollution emissions is to be minimized while air pollution is kept below a given threshold is shown.

We start in Section 2 by presenting the air pollution control with a Gaussian model, and the required notation. A brief introduction is carried out in Section 3 about semi-infinite programming and the used method to solve the proposed problems. In Section 4 four academic examples coded in a modeling language are described and the numerical results are shown in Section 5. We conclude in Section 6 and the appendix presents an example coded in the used modeling language.

2. Air pollution control – Gaussian model

The reader is pointed to [7,8] for a background reading in air pollution control and to [9,10] for some optimization models.

In order to describe the Gaussian model mathematical equations we consider a coordinate system where the origin is at ground level. The X and Y-axis extend horizontally and are perpendicular to each other. The Z-axis extends vertically perpendicular with the X and Y-axis (see Fig. 1). Let a and b be the x and y coordinates, respectively, of the pollution emission position. The stack pollution emission occurs at some height h above the ground ($z = 0$).

Assuming that the plume spread has a Gaussian distribution[1], the concentration, $\mathscr{C}$ (g m^{-3}), of gas or aerosols (particles less than about 20 μm diameter) at position x, y, and z from a continuous source with an effective emission height, H, is given by

$$\mathscr{C}(x,y,z,H) = \frac{\mathscr{Q}}{2\pi\sigma_y\sigma_z\mathscr{U}} e^{-\frac{1}{2}\left(\frac{\mathscr{Y}}{\sigma_y}\right)^2} \left(e^{-\frac{1}{2}\left(\frac{z-H}{\sigma_z}\right)^2} + e^{-\frac{1}{2}\left(\frac{z+H}{\sigma_z}\right)^2} \right), \tag{1}$$

where $\mathscr{Q}$ (g s^{-1}) is the uniform emission rate of pollutants, $\mathscr{U}$ (m s^{-1}) is the mean wind speed affecting the plume and σ_y (m) and σ_z (m) are the standard deviations of plume concentration distributed in the horizontal and vertical planes, respectively. $\mathscr{Y}$ is given by

$$\mathscr{Y} = (x-a)\sin(\theta) + (y-b)\cos(\theta), \tag{2}$$

where θ (rad) is the mean wind direction ($0 \leqslant \theta \leqslant 2\pi$).

In Eq. (1) the variable x does not appear explicitly in the formula, but the σ_y and σ_z standard deviations depend on the $\mathscr{X}$ variable given by

$$\mathscr{X} = (x-a)\cos(\theta) - (y-b)\sin(\theta). \tag{3}$$

Eqs. (2) and (3) provide a change of coordinates of the pollution emission point in the mean wind direction.

The second exponential term in (1) accounts for the pollutant reflection on the ground.

The Gaussian parameters σ_y and σ_z depend on the distance from the source and takes into account the atmospheric turbulence. The most common tabulated data is due to Pasquill [11] and Gifford [12]. Pasquill and Gifford data were summarized in [7] with additional graphs and tables.

The effective emission height, H, is the sum of the physical stack height, h (m), and the plume rise, ΔH (m).

[1] In fact the Gaussian distribution is an analytical solution to a simplified diffusion equation that considers a mass balance on a small volume.

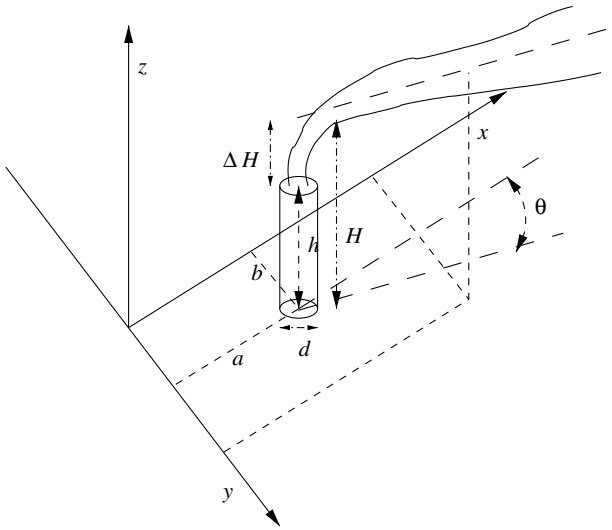

Fig. 1. Coordinate system and notation.

There are several equations to model the plume rise ΔH. The plume rise is directly proportional to the gas exit velocity (how fast the plume is coming out) and the plume buoyancy (in relation with the gas out temperature). It is also inversely proportional to the wind speed and to the stability class parameters (the more stable it is, the lower the plume will be). The most used formula for determining the plume rising height in stable atmosphere is due to Briggs [13], given by

$$\Delta H = 2.6\left(\frac{F}{\mathscr{U}s}\right)^{\frac{1}{3}},$$

where F is the buoyancy flux parameter and s is the parameter of atmospheric stability.

The buoyancy parameter F is determined by

$$F = \frac{d^2 V_o g(T_o - T_e)}{4T_o},$$

where d (m) is the internal stack diameter, V_o $(\mathrm{m\,s^{-1}})$ is the stack gas exit velocity, g is the acceleration of gravity $(9.806\ \mathrm{ms^{-2}})$, T_o (K) is the gas out temperature and T_e (K) is the environment temperature.

The parameter s of atmospheric stability is determined as follows:

$$s = \frac{g}{T_o}\frac{\mathrm{d}\theta}{\mathrm{dz}},$$

where $\mathrm{d}\theta/\mathrm{d}z$ is the potential temperature's gradient. As a default approximation, for a stable atmosphere, $\mathrm{d}\theta/\mathrm{d}z$ is taken as $0.020\ \mathrm{K\,m^{-1}}$.

To proceed with a multiple source scenario we need to further extend our notation. Let $\mathscr{C}_i$, $i = 1, \ldots, n$, be the contribution of source i for the total pollutant concentration, where n is the number of pollution sources distributed in a given region. The use of the i subscript also extends to the ith source specific parameters (a_i, b_i, h_i, d_i, $\mathscr{Q}_i$, $(V_o)_i$, ΔH_i and H_i).

By further assuming the pollutant as chemical inert, its concentration can be computed by superposition of the n pollution sources. The pollutant concentration at a point (x, y, z) can be computed by summing up individual source contribution $\sum_{i=1}^{n} \mathscr{C}_i(x, y, z, H_i)$.

With the optimal air pollution control in mind we can devise the following mathematical programming problems.

In the planning phase, the computation of the minimal stack height, while keeping the pollution level below some threshold $\mathscr{C}_0$, in a given area $\mathscr{R}$, at ground level, can be formulated as follows:

$$\begin{aligned} \min_{h=(h_1,\ldots,h_n)\in R^n} \quad & \sum_{i=1}^{n} f_i(h_i), \\ \text{s.t.} \quad & \sum_{i=1}^{n} \mathscr{C}_i(x, y, 0, H_i) \leqslant \mathscr{C}_0, \\ & h_{\mathrm{lb}} \leqslant h \leqslant h_{\mathrm{ub}} \\ & \forall (x, y) \in \mathscr{R}, \end{aligned} \tag{4}$$

where f_i, $i = 1, \ldots, n$, are construction cost functions, associated with the stack height. The simple bound constraint $h_{\text{lb}} \leqslant h \leqslant h_{\text{ub}}$ is to be understood as componentwise and it allows to consider legal and technical bound on the stacks, respectively.

For a given region $\mathscr{R}$, with fixed pollution sources the maximum air pollution concentration (l^*) can be estimated by solving the following mathematical programming problem.

$$\begin{aligned} \min_{l \in R} \quad & l, \\ \text{s.t.} \quad & \sum_{i=1}^{n} \mathscr{C}_i(x, y, 0, H_i) \leqslant l \\ & \forall (x, y) \in \mathscr{R}. \end{aligned} \tag{5}$$

The points $(x^*, y^*) \in \mathscr{R}$ where $\sum_{i=1}^{n} \mathscr{C}_i(x^*, y^*, 0, H_i) = l^*$ are global maximizers (where the maximum pollution concentration is attained) that make the constraint active and are the positions where the sampling stations should be placed.

Another possible formulation (see [14–17] for a similar formulation) is the determination of the pollution abatement that is necessary in a given region to fulfill the legal threshold requirement. The minimum production cost (minimum cost with cleaning), while the air pollution is kept bellow a given threshold, can be obtained by solving the following mathematical programming problem:

$$\begin{aligned} \min_{(r_1, \ldots, r_n) \in R^n} \quad & \sum_{i=1}^{n} f_i(r_i), \\ \text{s.t.} \quad & \sum_{i=1}^{n} (1 - r_i) \mathscr{C}_i(x, y, 0, H_i) \leqslant \mathscr{C}_0 \\ & \forall (x, y) \in \mathscr{R}, \end{aligned} \tag{6}$$

where $f_i(r_i)$, $i = 1, \ldots, n$, is what one pays for the abatement on source i (cleaning or not producing) and $r_1, \ldots, r_n$ are the percentage of the pollution abatement factor.

The proposed mathematical programming problems belong to a well known class of semi-infinite programming problems [3–6], where problem (6) is a linear SIP. In some real world applications these problems are characterized by possessing at least a constraint that must be satisfied for a given period (infinite) of time or space. A real application can also be found in the robust approach to portfolio optimization [6]. In the herein considered formulations the constraint considers a threshold value to the allowed pollution level for a given region.

3. Semi-infinite programming

Many engineering problems, such as robot trajectory planning, optimal signal sets, production planning, and digital filter design can be posed as semi-infinite programming (SIP) problems (see [4] for several SIP application). Air pollution control has also deserved some attention in the SIP context [4,17]. In this section we give a brief overview on semi-infinite programming (SIP) and about the method used to solve the proposed problems.

3.1. Semi-infinite programming overview

Semi-infinite programming problems can be described in the following mathematical form:

$$\begin{aligned} \min_{u \in R^n} \quad & f(u), \\ \text{s.t.} \quad & g_i(u, v) \leqslant 0, \quad i = 1, \ldots, m, \\ & u_{\text{lb}} \leqslant u \leqslant u_{\text{ub}} \\ & \forall v \in \mathscr{V} \subset R^p, \end{aligned} \tag{7}$$

where $f(u)$ is the objective function, $g_i(u, v)$, $i = 1, \ldots, m$, are the infinite constraint functions and u_{lb}, u_{ub} are the lower and upper (simple) bounds on u.

Problem (7) can be stated in a more general form, by including finite (constraints only depending on u) equality and inequality constraints, but this definition just suit our purpose.

These problems are called semi-infinite programming problems due to the constraints $g_i(u, v) \leqslant 0$, $i = 1, \ldots, m$. Each infinite constraint of this type must be satisfied for every $v \in \mathscr{V}$. These constraints could be represented by an equivalent notation as $g_{i,v}(u) \leqslant 0$. We can think of $\mathscr{V}$ as an infinite index set and therefore (7) is a problem with finitely many variables over an infinite set of constraints.

Herein, the set $\mathscr{V}$ is assumed to be a cartesian product of intervals ($[\alpha_1, \beta_1] \times \cdots \times [\alpha_p, \beta_p]$).

Natural ways to solve the SIP problem (7) is to replace the infinite set $\mathscr{V}$ by a finite one, or solving a sequence of finite subproblems (where the infinite set $\mathscr{V}$ is replaced by finite approximations). There are several ways of doing this. Discretization, exchange and reduction type methods (see [4], for a more detailed explanation) are the major classes.

Publicly available tools to deal with SIP problems are SIPAMPL [18] and the NSIPS [19] solver. AMPL [20] is a modeling language for mathematical programming problems. AMPL provides an interface that allows a wide variety of solvers to access problems coded in the AMPL language. Together with the simple and powerful modeling language, AMPL also provides automatic differentiation. Since AMPL is limited to finite programming, SIPAMPL was developed to allow the codification of SIP problems. SIPAMPL stands for SIP with AMPL and it comprises an extension to the AMPL modeling language to allow coding SIP problems.

SIPAMPL takes advantage of the AMPL modeling language to allow the codification of SIP problems. It provides also an interface to connect to any SIP solver, a database with over than 160 coded SIP problems, an interface that allows MATLAB [21] to use the SIP problems available in the database and a select tool for query the database for SIP problems with specific characteristics.

Although AMPL is a commercial software a limited student edition is available for free. SIPAMPL can be obtained from www.norg.uminho.pt/aivaz, but it relies on the availability of AMPL.

NSIPS stands for Nonlinear SIP Solver. NSIPS uses the SIPAMPL interface to obtain the problem to be solved. While some advanced skills are necessary to get NSIPS working, a full version is publicly available from the NEOS server (see www-neos.mcs.anl.gov).

To the best of our knowledge MATLAB is the only commercial software that provides a solver for SIP in its optimization toolbox [22] (`fseminf` function).

3.2. The discretization method

The following definitions are needed to describe the discretization method presented in this section:

Definition 3.1. A grid is a set of the form $\mathscr{V}[\xi = (\xi_1, \xi_2, \ldots, \xi_p)] = \mathscr{V} \cap \{v = (v_1, v_2, \ldots, v_p) : v_i = \alpha_i + j\xi_i, j = 0, \ldots, n_i; i = 1, \ldots, p\}$, where $n_i = (\beta_i - \alpha_i)/\xi_i$.

Definition 3.2. NLP($\mathscr{V}[\xi]$) is the following nonlinear programming subproblem:

$$\begin{aligned} \min_{u \in R^n} \quad & f(u), \\ \text{s.t.} \quad & g_i(u, v) \leqslant 0, \quad i = 1, \ldots, m, \\ & u_{\text{lb}} \leqslant u \leqslant u_{\text{ub}} \\ & \forall v \in \mathscr{V}[\xi] \end{aligned} \tag{8}$$

Discretization methods ([23–25]) try to solve a SIP problem by solving a sequence of finite subproblems where the infinite set $\mathscr{V}$ is replaced by a finite one that is, usually, a grid of points. These methods do not guarantee an exact or near exact solution to the SIP but will provide a solution in the finest (final) grid. These methods start with an initial approximation to the solution (in a coarse grid) and proceed to the final grid by solving a specified number of subproblems in intermediate grids. To keep the number of constraints in the subproblems in a manageable size the method use only a selected set of points in each grid.

Algorithm 3.1. Conceptual algorithm

- (Step 0:) Let r be the number of requested refinements. Given a set of grid parameters (ξ^0, ξ^1, …, ξ^r) define $\mathscr{V}[\xi^0]$ and set $\widetilde{\mathscr{V}}[\xi^0] = \mathscr{V}[\xi^0]$. Solve the NLP($\widetilde{\mathscr{V}}[\xi^0]$) and let u^0 be the solution found.
- (Step k:) If there exists points in the grid $\mathscr{V}[\xi^{k-1}]$ that makes a constraint violated, i.e., $\{\exists \bar{v} \in \mathscr{V}[\xi^{k-1}] : g_i(u^{k-1}, \bar{v}) > 0, i = 1, \ldots, m\}$
  then: let $\widetilde{\mathscr{V}}[\xi^{k-1}] \subseteq \mathscr{V}[\xi^{k-1}]$. Solve NLP($\widetilde{\mathscr{V}}[\xi^{k-1}]$) and let u^{k-1} be the solution found. Continue in step k.
  else: if $k > r$ then stop with u^{k-1} as an approximate solution to the SIP problem. Otherwise set $\widetilde{\mathscr{V}}[\xi^k] \subseteq \mathscr{V}[\xi^k]$. Solve NLP($\widetilde{\mathscr{V}}[\xi^k]$) and let x^k be the solution found. Go to step $k + 1$.

The algorithm is presented as conceptual since the rules for constructing the sets $\mathscr{V}[\xi]$ are not defined. The reader is pointer to [23–25] for further details.

The discretization methods are also known by outer approximation methods, since the SIP problem feasible region is a subset of the finite subproblems feasible region. Since the discretization method is providing a solution on the finest grid some infeasibility w.r.t. the SIP problem may be occurring. In order to validate the obtained solutions (u^*) we are reporting an infeasibility measure by solving m (finite) optimization problems (for each $i = 1, \ldots, m$).

$$\text{infeasibility} = \max_{i=1,\ldots,m} \max_{v \in \mathscr{V}} g_i(u^*, v). \tag{9}$$

4. Examples of air pollution control problems

In this section we describe four examples with data collected from the literature on air pollution control.

These problems were coded in the SIPAMPL modeling language and are publicly available in the SIPAMPL problems database. For the purpose of illustration a simple example coded in SIPAMPL is provided in Appendix A.

4.1. Minimal stack height

An air pollution control problem was formulated in [26] to show the reliability of an optimization procedure, in obtaining the global maximum of the sulfur dioxide concentration in a given region. The proposed problem data will be used herein in minimizing the total stack height while the pollutant (sulfur dioxide) is kept bellow a given threshold.

The problem consists of a region with 10 stacks. The environment temperature (T_e) is 283 K and the gas emission temperature is 413 K. A wind speed ($\mathscr{U}$) of 5.64 m s^{-1} and a wind direction (θ) of 3.996 rad ($\approx 229°$) are considered. The stack and emission data for the 10 stacks is given in Table 1.

The authors provide the mathematical expression for the σ_y and σ_z, obtained from curve fitting the figures appearing in [7].

The stack height in Table 1 was used as an initial guess for the SIP formulation and a squared region of 40*km* width is considered ($\mathscr{R} = [-20000, 20000] \times [-20000, 20000]$).

This problem is coded in the (SIP)AMPL format and is publicly available in the SIPAMPL database (file `vazl_briggs.mod`).

4.2. Maximum attained pollution and sampling stations planning

Hypothetical source data from [16] is used to illustrate the computation of the maximum pollution level l^* using formulation (5). The source data is shown in Table 2. The region considered was $\mathscr{R} = [0, 24140] \times [0, 24140]$ (a square of approximately 582 km^2). The environment air temperature was 284 K, with a wind speed of 5 m s^{-1} and direction of 3.927 rad

Table 1
Stack and emission data

Source	a_i (m)	b_i (m)	h_i (m)	d_i (m)	$\mathscr{Q}_i$ (g s^{-1})	$(V_o)_i$ (m s^{-1})
1	−3000	−2500	183	8.0	2882.6	19.245
2	−2600	−300	183	8.0	2882.6	19.245
3	−1100	−1700	160	7.6	2391.3	17.690
4	1000	−2500	160	7.6	2391.3	17.690
5	1000	2200	152.4	6.3	2173.9	23.404
6	2700	1000	152.4	6.3	2173.9	23.404
7	3000	−1600	121.9	4.3	1173.9	27.128
8	−2000	2500	121.9	4.3	1173.9	27.128
9	0	0	91.4	5.0	1304.3	22.293
10	1500	−1600	91.4	5.0	1304.3	22.293

Table 2
Stack and emission data for the maximum pollution level

Source	a_i (m)	b_i (m)	h_i (m)	d_i (m)	$\mathscr{Q}_i$ (g s^{-1})	$(V_o)_i$ (m s^{-1})	$(T_o)_i$ (K)
1	9190	6300	61.0	2.6	191.1	6.1	600
2	9190	6300	63.6	2.9	47.7	4.8	600
3	9190	6300	30.5	0.9	21.1	29.2	811
4	9190	6300	38.1	1.7	14.2	9.2	727
5	9190	6300	38.1	2.1	7.0	7.0	727
6	9190	6300	21.9	2.0	59.2	4.3	616
7	9190	6300	61.0	2.1	87.2	5.2	616
8	8520	7840	36.6	2.7	25.3	11.9	477
9	8520	7840	36.6	2.0	101.0	16.0	477
10	8520	7840	18.0	2.6	41.6	9.0	727
11	8050	7680	35.7	2.4	222.7	5.7	477
12	8050	7680	45.7	1.9	20.1	2.4	727
13	8050	7680	50.3	1.5	20.1	1.6	727
14	8050	7680	35.1	1.6	20.1	1.5	727
15	8050	7680	34.7	1.5	20.0	1.6	727
16	9190	6300	30.0	2.2	24.7	9.0	727
17	5770	10810	76.3	3.0	67.5	10.7	473
18	5620	9820	82.0	4.4	66.7	12.9	603
19	4600	9500	113.0	5.2	63.7	9.3	546
20	8230	8870	31.0	1.6	6.3	5.0	460
21	8750	5880	50.0	2.2	36.2	7.0	460
22	11240	4560	50.0	2.5	28.8	7.0	460
23	6140	8780	31.0	1.6	8.4	5.0	460
24	14330	6200	42.6	4.6	172.4	13.4	616
25	14330	6200	42.6	3.7	171.3	16.1	616

(≈225°). The same weather stability as in the maximum stack height example is used, while in [16] the numerical results were obtained with a Pasquill stability class D.

This problem is also coded in the SIPAMPL format and is publicly available in its problem database (file `vaz2_briggs.mod`).

4.3. Air pollution abatement

In [15] the authors describe an example of policy abatement in air pollution that uses the Sutton equation for the expected pollution concentration. A slightly different problem was used latter in a paper from Van Honstede [17]. This case is available in the SIPAMPL database included in the Watson set of problems (see [27]).

Although the problem data proposed in [15] is valid and can be used to illustrate the potential of this formulation, it was probably scaled. In the next paragraphs the problem proposed by Gustafson and Kortanek [15] is described and we postpone to the end of this subsection the use of the same problem formulation with the data presented in [26].

In a given city there are three plants $\mathscr{P}_1$, $\mathscr{P}_2$ and $\mathscr{P}_3$, emitting the amounts e_1, e_2 and e_3, with $0 \leqslant e_i \leqslant 2$, $i = 1,2,3$, of a certain pollutant. The city ordinance states that the expected pollution level must not exceed a standard $\mathscr{C}_0$ under the most common weather conditions, i.e., a steady westerly wind ($\theta = 0$ in the Gaussian model) of constant speed $\mathscr{U}$. The city would also like to know where to place the sampling stations and their number in order to check compliance with the ordinance. Assuming that the revenue is proportional to the emission rate and that the total revenue of the three plants is a linear combination of the emissions, the optimization problem is

$$\begin{aligned} \max_{e_1,e_2,e_3 \in R} \quad & 2e_1 + 4e_2 + e_3, \\ \text{s.t.} \quad & \sum_{i=1}^{3} e_i \mathscr{C}(x,y,0,H_i) \leqslant \mathscr{C}_0, \\ & 0 \leqslant e_i \leqslant 2, \quad i = 1,2,3 \\ & \forall (x,y) \in [-1,4] \times [-1,4]. \end{aligned}$$

By setting $r_i = 2 - e_i$, $i = 1,2,3$, the previous maximization problem can be rewritten as a minimization problem, yielding

$$\begin{aligned} \min_{r_1,r_2,r_3 \in R} \quad & 2r_1 + 4r_2 + r_3 \\ \text{s.t.} \quad & \sum_{i=1}^{3} (2 - r_i) \mathscr{C}(x,y,0,H_i) \leqslant \mathscr{C}_0 \\ & 0 \leqslant r_i \leqslant 2, \quad i = 1,2,3 \\ & \forall (x,y) \in [-1,4] \times [-1,4]. \end{aligned} \tag{10}$$

Instead of the Sutton, the Gaussian expression is used to formulate this problem. Using the equivalence between the Sutton ($n = 1$, $C_x = C_y = 1$) and Gaussion expressions (see [15]) we have,

$$\sigma_y = \sigma_z = \begin{cases} \sqrt{\frac{\mathscr{x}}{2}} & \text{for } \mathscr{x} > 0 \\ 0 & \text{otherwise,} \end{cases}$$

and $\mathscr{C} = 0$ g m^{-3} is considered whenever σ_y or σ_z are zero.

A wind speed of $\mathscr{U} = (\frac{1}{2\pi})^2$ m s^{-1}, emission rate $\mathscr{Q} = 1$ g s^{-1} and $\mathscr{C}_0 = \frac{1}{2}$ g m^{-3} were considered. The effective stack heights and coordinates are given in Table 3 (no plume rise is considered).

The file `vaz3.mod` in the SIPAMPL database refers to this example.

As already mentioned the data presented in Table 3 is not meaningful in the pollution control field. To illustrate this formulation in a more realistic scenario we have used the data from [26]. Using the reported data in Table 1, the objective function considered was the sum of the reductions in all sources ($\sum_{i=1}^{10} r_i$) while the pollution concentration, at ground level, is kept bellow the Portuguese limit ($\sum_{i=1}^{10} (1 - r_i) \mathscr{C}_i(x,y,0,H_i) \leqslant 350$ μg m^{-3}). The file `vaz4_briggs.mod` in the SIPAMPL database refers to this example.

Table 3
Stack data for `vaz3.mod`

Source	a_i	b_i	h_i
1	0	1	1
2	0	0	1
3	2	−1	$\sqrt{2}$

Table 4
Numerical results for minimum stack height problem

	Instance 1	Instance 2	Instance 3
h_1	16.53	16.53	92.54
h_2	160.81	161.12	299.45
h_3	17.09	17.07	277.77
h_4	192.89	192.95	300.00
h_5	112.19	112.10	279.91
h_6	29.58	28.30	100.79
h_7	0.00	10.00	196.50
h_8	148.30	146.43	267.76
h_9	128.39	122.65	269.49
h_{10}	3.51	10.00	258.89
$f(u^*)$	809.29	817.14	2343.10

5. Numerical results

The discretization method available in the NSIPS [19] software package was selected. As already mentioned, the discretization method computes a solution in the finest grid and if the grid is not fine enough some infeasibility in the infinite constraints can occur. If a finer grid is to be required then the dimension of the first grid or the number of grid refinements should be changed from their default values in the solver. The default options were considered, except for the `method` and `disc_h` options. `method` selects the used method and was set to `disc_hett`, which changes the default method to the Hettich version of the discretization method [23]. `disc_h` changes the space (and consequently the number of points used) in the initial grid (ξ^0 in Algorithm 3.1).

5.1. Minimum stack height

In this example NSIPS was used with the option `disc_h=1000` (setting ξ^0 to $(1000, 1000)$, meaning that the initial grid uses an equally spaced grid of 1 km step).

Numerical results are shown in Table 4. Two threshold values for the pollution level and two lower limits on the stack height are considered, originating three different instances of the problem. In the first one a limit of 771.14 μg m^{-3} is considered ($\mathscr{C}_0 = 771.14$ μg m^{-3}) while the lower limit on the stack height is zero. In order to obtain a practical solution a maximum of 300 m is imposed on each stack height. The bound constraints on the stack height leads to the following problem:

$$\begin{aligned} \min_{(h_1,\ldots,h_{10})\in R^{10}} \quad & \sum_{i=1}^{10} h_i, \\ \text{s.t.} \quad & \sum_{i=1}^{n} \mathscr{C}_i(x,y,0,H_i) \leqslant \mathscr{C}_0, \\ & 0 \leqslant h_i \leqslant 300, \quad i = 1,\ldots,10 \\ & \forall (x,y) \in [-20000, 20000] \times [-20000, 20000]. \end{aligned}$$

Numerical results are shown in the first column of Table 4 and one stack has height equal to zero. Portuguese legislation[2] imposes a minimum stack height of 10 m. The stack height can only be inferior to 10 m if some legal[3] requirements are met. One way to prove that the requirements are met is by simulation, using a proper air pollution dispersion model. In instance 2 the same limit $\mathscr{C}_0$ is considered while the lower limit on the stack height is 10 m ($10 \leqslant h_i \leqslant 300$, $i = 1,\ldots,n$). Instance 3 considers the Portuguese[4] one hour limit on sulfur dioxide $\mathscr{C}_0 = 350$ μg m^{-3}.

The constraint contour, for the solution found for instance 3, is presented in Fig. 2. The contour was obtained with the MATLAB interface to SIPAMPL [28].

In order to check the feasibility of the solutions found, in the three instances, the following global optimization problem (see problem (9)) has to be solved with high accuracy.

$$\max_{v\in\mathscr{R}} \left(\sum_{i=1}^{n} \mathscr{C}_i(x,y,0,H_i) - \mathscr{C}_0 \right) \tag{11}$$

Instead of solving problem (11) with a solver for global optimization, we opted to solve problem (11) with a solver for local optimization, using the points in the final grid that make the infinite constraint active at the solution as initial guesses

[2] Decree law number 352/90 from 9 November 1990.
[3] Decree law number 78/2004 from 3 April 2004.
[4] Decree law number 111/2002 from 16 April 2002.

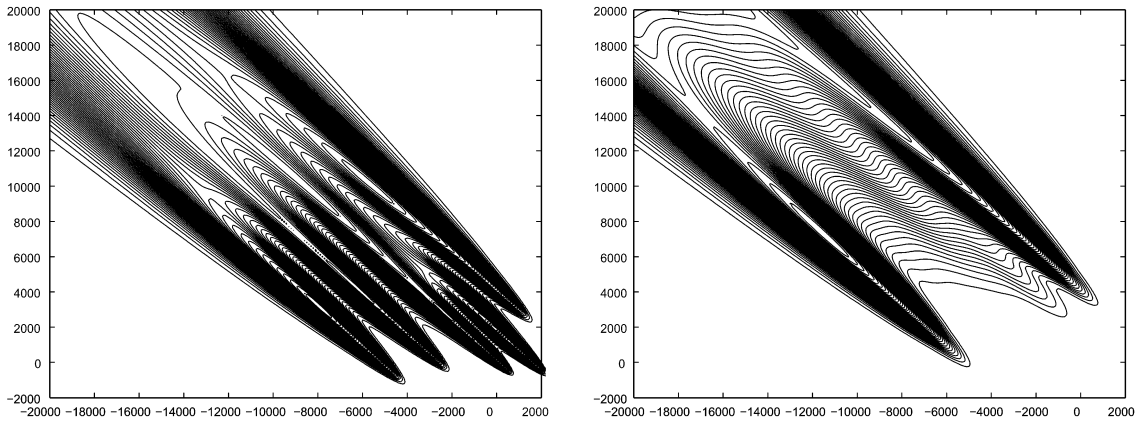

Fig. 2. Constraint contour of minimum stack height (instance 2 and 3, respectively).

for the local solver. The global solution to problem (11) is expected to be in the neighborhood of the point in the finest grid that makes the discretized constraint active.

The NPSOL [29] solver found the global maximum of problem (11), instance 1 and 2, at $(-7905.43, 3134.21)$ with infesibility $= 3.83 \times 10^{-5}$. For instance 3 the global maximum was attained at $(-11475.1, 7245.85)$ with infeasibility $= 3.89 \times 10^{-6}$.

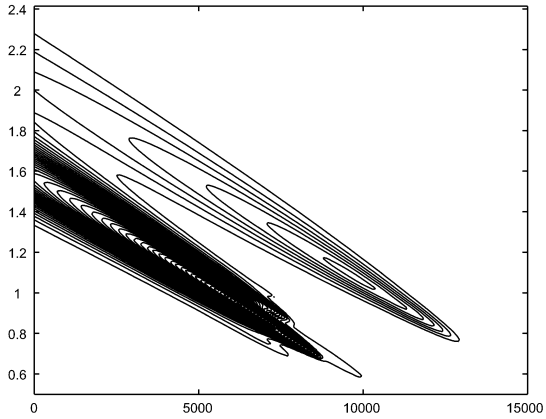

Fig. 3. Maximum pollution level contour.

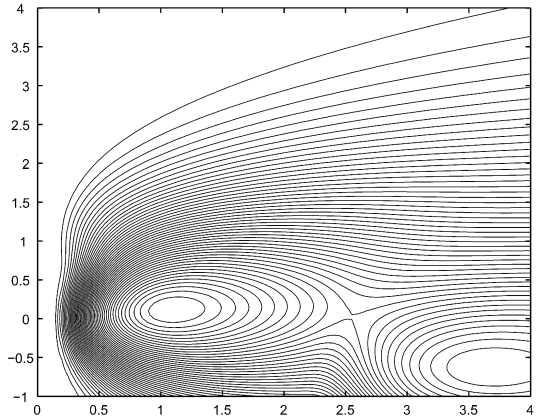

Fig. 4. Air pollution abatement contour (first example).

Table 5
Numerical results for pollution abatement, problem `vaz4_briggs.mod`

r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9	r_{10}	$f(u^*)$
0.42	1.00	0.54	1.00	1.00	0.38	0.69	0.42	1.00	1.00	7.44

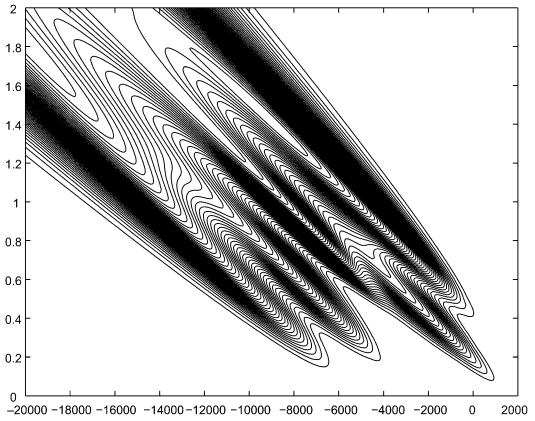

Fig. 5. Air pollution abatement contour, second example.

5.2. Maximum pollution level and sampling stations position

The same grid spacing of the previous formulation was used in the discretization method.

The results found by the discretization method was $l^* = 795\ \mu g\ m^{-3}$. The `vaz2_briggs` problem constraint maximum was attained at $(x, y) = (6000, 9500)$ (the only active point for the constraint in the final grid). While this point is a good position where the sampling station should be placed, other local maxima of the constraint could be considered, as it can be seen in the constraint contour Fig. 3.

When checking the infeasibility, NPSOL reports 1.72×10^{-4} attained at $(6611.96, 9070.38)$.

5.3. Air pollution abatement

For the first example (`vaz3.mod`), the numerical result found by the discretization method is $r^* = (0.987, 0.951, 0.943)$ with the option `disc_h` set to 0.05. The constraint maxima (active constraints) were attained at $(x, y)^1 = (1.100, 0.125)$, $(x, y)^2 = (1.100, 0.100)$ and $(x, y)^3 = (3.675, -0.625)$, where sampling stations should be placed to check the compliance with the ordinance.

The contour of the air pollution abatement problem constraint is presented in Fig. 4.

When checking the infeasibility, NPSOL reports 5.50×10^{-5} attained at $(1.10, 0.11)$.

Table 5 presents the numerical results obtained for the second example in this formulation. The option `disc_h` was set to 1000 and the initial guess is $r_i = 0$, $i = 1, \ldots, 10$, corresponding to no reduction in all sources.

Once again the obtained numerical results confirm that the scenario presented by Wang and Luus [26] would be not obvious to implement in practice. Five sources would need to use a technology with an 100% reduction and other two with a reduction of more than 50%. Fig. 5 presents the constraint contour of problem `vaz4.mod`.

When checking the infeasibility, NPSOL reports 5.60×10^{-7} attained at $(-9157.98, 14323.00)$.

6. Conclusions

Air pollution control problems can be posed as semi-infinite programming problems and efficiently solved by publicly available software. In these problems an objective function is to be optimized while a given threshold for the pollution, in a given region, is to be attained. In the present paper the plume spread was assumed to have a Gaussian distribution under mean weather conditions and the Briggs equation was used to compute the plume rise. In the presented examples, stack and emission data was collected from literature ([15,26]) to illustrate the proposed approach.

The formulation of air pollution control problems as SIP allows a great degree of freedom, since new objective and constraints can be easily introduced. The codification of the proposed problems in the SIPAMPL modeling language makes them publicly available to the research community, either to test other objectives and/or new constraints, or as SIP benchmark problems.

The discretization method implemented in the NSIPS solver was used to solve the coded problems and proved to be efficient.

Appendix A. An example coded in SIPAMPL

We have selected to present the problem described in (10) due to its simpleness. The following SIPAMPL code is self explanatory, since comments were added throughout the code (the AMPL comment character is an # mark).

```
###########################################################################
# Objective: Linear
# Constraints: Linear
###########################################################################
# Pollution control problem
#    Coded by A. Ismael F. Vaz 31/03/2004  aivaz@dps.uminho.pt
#    University of Minho, Portugal
#
# Policy abatement problem described by S.-A. Gustafson and K.O. Kortanek
#  in "Analytical properties of some multiple-source urban diffusion
#  models", Environment and Planning 4, 1972, pp. 31-41.
#
# Adapted for Gaussion model without plume rise.
#
###########################################################################
param pi := 4*atan(1);              # Pi
param nsources;                     # Number of pollution emission sources
param theta;                        # Wind direction
param U;                            # Mean wind speed

param xpos{1..nsources};            # Position of sources
param ypos{1..nsources};
param Q{1..nsources};               # Emission rates
param h{1..nsources};               # Stack height
param rinit{1..nsources};           # Abatement initial guess
param nt;                           # Number of coordinates

var r {i in 1..nsources}:=rinit[i];# Pollution abatement
var t {1..nt};                      # t variable t[1]-> x coordinate
                                    # t variable t[2]-> y coordinate
# Translation and rotation
var X{i in 1..nsources} =
                     (t[1]-xpos[i])*cos(theta)-(t[2]-ypos[i])*sin(theta);
var Y{i in 1..nsources} =
                     (t[1]-xpos[i])*sin(theta)+(t[2]-ypos[i])*cos(theta);

# Standard deviations
var sigmaY{i in 1..nsources} = if(X[i]>0) then (X[i]/2)^0.5 else 0;
var sigmaZ{i in 1..nsources} = if(X[i]>0) then (X[i]/2)^0.5 else 0;
var sigmaZY{i in 1..nsources} = if(X[i]>0) then (X[i]/2) else 0;

# Concentration of gas
var C {i in 1..nsources} = if(sigmaY[i]!=0 && sigmaZ[i]!=0)then
                  (Q[i]/(pi*sigmaZY[i]*U))*
                  exp(-0.5*((Y[i]/sigmaY[i])^2+(h[i]/sigmaZ[i])^2))
                else 0;
```

```
minimize fx:                                    # Minimize the abatement
    2*r[1]+4*r[2]+r[3];
subject to tcons:                               # Constraint
    sum {i in 1..nsources} ((2-r[i])*C[i]) <= 0.5;
subject to bounds {i in 1..nsources}:           # Simple bounds
    0 <= r[i] <= 2;
subject to tbound {i in 1..nt}:                 # Region
    -1 <= t[i] <= 4;
data;                                           # Provide problem data
param nsources=3;                               # Number of sources
param nt := 2;                                  # Region dimension
param theta := 0;                               # Wind direction in rad
param U     := .02533029591058444286;           # Mean wind speed  (1/(2*pi))^2
param xpos :=                                   # x position on the ground (m)
    1    0    2    0    3     2;
param ypos :=                                   # y position on the ground (m)
    1    1    2    0    3    -1;
param h :=                                      # Effective stack height
    1    1    2    1    3    1.41421356237309504880;
param Q :=                                      # Uniform emission rate
    1    1    2    1    3     1;
param rinit :=                                  # Initial guess
    1    1    2    1    3     1;
option solver nsips;                            # Select solver and set options
option nsips_options 'method=disc_hett disc_h=0.05';
option nsips_auxfiles rc;                       # Provide auxiliary files
solve;                                          # Solve the problem
```

References

[1] U.S. E.P.A., U.S. Environment Protection Agency. <www.epa.gov>.
[2] N.S. Holmes, L. Morawska, A review of dispersion modelling and its applications to the dispersion of particles: an overview of different dispersion models available, Atmos. Environ. 40 (2006) 5902–5928.
[3] M. Goberna, M.A. López (Eds.), Semi-Infinite Programming: Recent Advances, Nonconvex Optimization and its Applications, vol. 57, Kluwer Academics Publishers, Dordrecht, The Netherlands, 2001.
[4] R. Hettich, K.O. Kortanek, Semi-infinite programming: theory, methods, and applications, SIAM Rev. 35 (3) (1993) 380–429.
[5] R. Reemtsen, J.-J. Rückmann (Eds.), Semi-Infinite Programming, Kluwer Academic Publishers, 1998.
[6] M.A. López, G. Still, Semi-infinite programming, Eur. J. Oper. Res. 180 (2007) 491–518.
[7] D.B. Turner, Workbook of Atmospheric Dispersion Estimates, second ed., Lewis publishers, Boca Raton, 1994.
[8] N. de Nevers, Air Pollution Control Engineering, McGraw-Hill, 2000.
[9] W.W. Cooper, H. Hemphill, Z. Huang, S. Li, V. Lelas, D.W. Sullivan, Survey of mathematical programming models in air pollution management, Eur. J. Oper. Res. 96 (1) (1997) 1–35.
[10] H.J. Greenberg, Mathematical programming models for environmental quality control, Oper. Res. 43 (4) (1995) 578–622.
[11] F. Pasquill, The estimation of the dispersion of windborne material, Meteorol. Mag. 90 (1063) (1961) 33–49.
[12] F.A. Gifford, Use of routine observations for estimating atmospheric dispersion, Nucl. Safety 2 (1961) 47–57.
[13] G.A. Briggs, Plume rise predictions, in: Lectures on Air Pollution and Environmental Impact Analysis, American Meteorological Society, Boston, MA, 1975, pp. 59–111.
[14] W.L. Gorr, S.-Å. Gustafson, K.O. Kortanek, Optimal control strategies for air quality standards and regulatory policy, Environ. Plann. 4 (2) (1972) 183–192.
[15] S.-Å. Gustafson, K.O. Kortanek, Analytical properties of some multiple-source urban diffusion models, Environ. Plann. 4 (1972) 31–41.
[16] S.-Å. Gustafson, K.O. Kortanek, J.R. Sweigart, Numerical optimization techniques in air quality modeling: objective interpolation formulas for the spatial distribution of pollutant concentration, Appl. Meteorol. 16 (12) (1977) 1243–1255.
[17] W. Van Honstede, An approximation method for semi-infinite problems. In [30], 1979, pp. 127–136.
[18] A.I.F. Vaz, E.M.G.P. Fernandes, M.P.S.F. Gomes, SIPAMPL: semi-infinite programming with AMPL, ACM Trans. Math. Softw. 30 (1) (2004) 47–61.
[19] A.I.F. Vaz, E.M.G.P. Fernandes, M.P.S.F. Gomes, NSIPS v2.1: Nonlinear Semi-Infinite Programming Solver. Technical Report ALG/EF/5-2002, Universidade do Minho, Braga, Portugal, December 2002. <http://www.norg.uminho.pt/aivaz>.
[20] R. Fourer, D.M. Gay, B.W. Kernighan, A modeling language for mathematical programming, Manage. Sci. 36 (5) (1990) 519–554.
[21] MathWorks, MATLAB, The MathWorks Inc., 2001, Version 6.1, Release 12.1.
[22] MathWorks, MATLAB Optimization Toolbox, The MathWorks Inc., 2001. In [21].

[23] R. Hettich, An implementation of a discretization method for semi-infinite programming, Math. Program. 34 (3) (1986) 354–361.
[24] R. Hettich, G. Gramlich, A note on an implementation of a method for quadratic semi-infinite programming, Math. Program. 46 (1990) 249–254.
[25] R. Reemtsen, Discretization methods for the solution of semi-infinite programming problems, J. Optimiz. Theory Appl. 71 (1) (1991) 5–103.
[26] B.-C. Wang, R. Luus, Reliability of optimization procedures for obtaining global optimum, AIChE J. 24 (4) (1978) 619–626.
[27] C.J. Price, Non-Linear Semi-Infinite Programming. Ph.D. Thesis, University of Canterbury, New Zealand, August 1992.
[28] A.I.F. Vaz, E.M.G.P. Fernandes, A new interface between SIPAMPL and MATLAB to solve semi-infinite programming problems, WSEAS Trans. Inform. Sci. Appl. 3 (12) (2006) 2331–2338.
[29] P.E. Gill, W. Murray, M.A. Saunders, M.H. Wright, User's Guide for NPSOL: A Fortran Package for Nonlinear Programming, Stanford University, 1986.
[30] R. Hettich (Ed.), Semi-Infinite Programming, Springer-Verlag, 1979.