Int. J. Appl. Math. Comput. Sci., 2008, Vol. 18, No. 3, 307–318
DOI: 10.2478/v10006-008-0028-5

amcs

MOBILE SENSOR ROUTING FOR PARAMETER ESTIMATION OF DISTRIBUTED SYSTEMS USING THE PARALLEL TUNNELING METHOD

TOMASZ ZIĘBA, DARIUSZ UCIŃSKI

Institute of Control and Computation Engineering
University of Zielona Góra, ul. Podgórna 50, 65–246 Zielona Góra, Poland
e-mail: t.zieba@weit.uz.zgora.pl, d.ucinski@issi.uz.zgora.pl

The paper deals with the problem of optimal path planning for a sensor network with mutliple mobile nodes, whose measurements are supposed to be primarily used to estimate unknown parameters of a system modelled by a partial differential equation. The adopted framework permits to consider two- or three-dimensional spatial domains and correlated observations. Since the aim is to maximize the accuracy of the estimates, a general functional defined on the relevant Fisher information matrix is used as the design criterion. Central to the approach is the parameterization of the sensor trajectories based on cubic B-splines. The resulting finite-dimensional global optimization problem is then solved using a parallel version of the tunneling algorithm. A numerical example is included to clearly demonstrate the idea presented in the paper.

Keywords: sensor network, distributed parameter systems, optimum experimental design, tunneling algorithm, parallel computing.

1. Introduction

Since for distributed parameter systems (DPSs) it is most often impossible to observe their states over the entire spatial domain, a natural question arises of where to locate discrete sensors so as to accurately estimate the unknown system parameters. Both researchers and practitioners do not doubt that making use of sensors placed in an 'intelligent' manner may lead to dramatic gains in the achievable accuracy of the resulting parameter estimates, so efficient sensor location strategies are highly desirable. In turn, the complexity of the sensor location problem implies that there are very few sensor placement methods which are readily applicable to practical situations. What is more, they are not well known among researchers. This generates keen interest in the potential results, as the motivations to study the sensor location problem stem from practical engineering issues. The optimization of air quality monitoring networks is among the most interesting ones. One of the tasks of environmental protection systems is to provide expected levels of pollutant concentrations. But to produce such a forecast, a smog prediction model is necessary which is usually chosen in the form of an advection-diffusion partial-differential equation. Its calibration requires parameter estimation, e.g., the unknown spatially varying turbulent diffusivity tensor should be identfied based on the measurements from monitoring stations. Since measurement transducers are usually rather costly and their number is limited, we are faced with the problem of how to optimize their locations in order to obtain the most precise model. Other stimulating applications include, among other things, groundwater-resources management, the recovery of valuable minerals and hydrocarbon, model calibration in meteorology and oceanography, chemical engineering, automated inspection in static and active hazardous environments where trial-and-error sensor planning cannot be used (e.g., in nuclear power plants), or emerging smart material systems (Nehorai *et al.*, 1995; Porat and Nehorai, 1996; Jeremić and Nehorai, 1998; Jeremić and Nehorai, 2000; Navon, 1997; Daescu and Navon, 2004; Christofides, 2001; Banks *et al.*, 1996; Sun, 1994; Uciński, 2005). The operation and control of such systems usually requires precise information on the parameters which condition the accuracy of the underlying mathematical model, but such information is only available through a limited number of possibly expensive sensors.

The sensor placement problem was attacked from various angles, but the results communicated by most

authors are limited to the selection of stationary sensor positions (for reviews, see (Kubrusly and Malebranche, 1985; Uciński, 1999; Uciński, 2000; Uciński, 2005)). An intuitively clear generalization is to apply sensors which are capable of continuously tracking points providing at a given time moment the best information about the parameters (such a strategy is alternatively called continuous scanning). However, communications in this field are rather limited. Using mobile sensor network nodes, we can expect the minimal value of an adopted design criterion to be lower than the one with no mobility. In the seminal article (Rafajłowicz, 1986), the D-optimality criterion defined on the Fisher Information Matrix (FIM) associated with the estimated parameters is considered and an optimal time-dependent measure is sought, rather than the trajectories themselves. In (Porat and Nehorai, 1996), a single moving concentration sensor is used to detect and localize a vapour-emitting source for a diffusion equation. The sensor is guided to minimize the A-optimality criterion and parameter estimates are updated in subsequent stages of the experiment, i.e., a sequential experiment is conducted. However, no model of the sensor dynamics is considered, nor are restrictions imposed on sensor motions encountered in practice.

On the other hand, Uciński (Uciński, 2005; Uciński, 2000; Uciński and Korbicz, 2001), apart from generalizations, develops some computational algorithms based on the FIM. He reduces the problem to a state-constrained optimal control one for which solutions are obtained via the methods of successive linearizations, which is capable of handling various constraints imposed on sensor motions. In turn, the work (Uciński and Chen, 2005) was intended as an attempt to properly formulate and solve the time-optimal problem for moving sensors which observe the state of a DPS so as to estimate some of its parameters. In (Uciński and Chen, 2006), a similar technique was presented so as to make the Hessian of the parameter estimation well conditioned subject to an additional constraint imposed on the achievable D-efficiency of the solutions. In the vein of optimum experimental design, joint optimization of trajectories and measurement accuracies of heterogeneous mobile nodes in a sensor network was considered in (Tricaud *et al*., 2008). Finally, in (Uciński and Demetriou, 2008), a non-trivial generalization of the same approach was proposed for the problem of estimating the states of a stochastic DPS via the Kalman-Bucy filter.

The limited interest in mobile observations for parameter estimation in DPSs is in conflict with recent advances in hardware, sensor and networking technologies which enable large-scale deployment of superior data acquisition systems with adjustable resolutions, called *sensor networks*. Each sensor node has a sensing capability, as well as limited energy supply, computing power, memory and communication ability. These inexpensive, low-power communication devices can be deployed throughout the physical space, providing dense sensing close to physical phenomena, processing and communicating this information, and coordinating actions with other nodes. In modern applications, sensors can be located on various platforms and these platforms can be highly dynamic in motion. What is more, technological advances in communication systems and the growing ease in making small, low power and inexpensive mobile systems now render it feasible to deploy a group of networked vehicles in a number of environments (Cassandras and Li, 2005; Zhao and Guibas, 2004).

In a typical sensor network application, sensors are supposed to be deployed so as to monitor a region and collect the most valuable information from the observed system. The quality of sensor deployment can be quantified by the appropriate performance indices and optimum sensor node configurations can thus be sought. The resulting observation strategies concern optimally planning trajectories of mobile nodes. Up to now, approaches aiming at guaranteeing a dense region coverage or satisfactory network connectivity have dominated this line of research and abstracted away from the mathematical description of the physical processes underlying the observed phenomena. In this way, much information is lost which could potentially be used to make the operation of the sensor network more efficient and yield substantial gains in the functionality of the whole source localization system. Recent works (Uciński and Patan, 2007; Demetriou, 2006b; Demetriou, 2006a; Hussein and Demetriou, 2007; Demetriou, 2007) demonstrate that the inclusion of a DPS model into the optimization setting can substantially improve the quality of the information collected by the network.

To a certain extent, a barrier to widespread use of model-based sensor trajectory design techniques is the complexity and large scale of the attendant computations. This issue becomes of paramount importance in case the design is supposed to be performed *on-line*. This work is intended as an attempt to demonstrate that these computations can be parallelized with relative ease and even for fairly complex cases solutions can be obtained in reasonable times on a cluster of low-cost PCs using the Message Passing Interface (MPI). The approach adopted here can be called 'direct' in the sense that sensor trajectories are our design variables. This stands in contrast to the approach developed in (Uciński, 2005) and related works, where algorithmic optimal control was extensively used. Here we show that useful solutions can be efficiently obtained without resorting to such sophisticated machinery.

Specifically, in Section 2 we give a brief exposition of the sensor location problem in question. Section 3 provides an exposition of the tunneling algorithm. Its parallel version is set forth in Section 4. Section 5 includes a nu-

merical example to clearly demonstrate the ideas presented in the paper.

2. Experimental design problem in question

2.1. Optimal sensor routing for correlated observations. Consider a bounded spatial domain $\Omega \subset \mathbb{R}^d$ with a sufficiently smooth boundary Γ, a bounded time interval $T = (0, t_f]$, and a distributed parameter system (DPS) whose scalar state at a spatial point $x \in \bar{\Omega} \subset \mathbb{R}^d$ and a time instant $t \in \bar{T}$ is denoted by $y(x,t)$. Mathematically, the system state is governed by the partial differential equation (PDE)

$$\frac{\partial y}{\partial t} = \mathcal{F}\big(x,t,y,\theta\big) \quad \text{in } \Omega \times T, \tag{1}$$

where $\mathcal{F}$ is a well-posed, possibly nonlinear, differential operator which involves first- and second-order spatial derivatives and may include terms accounting for forcing inputs specified *a priori*. The PDE (1) is accompanied by the appropriate boundary and initial conditions

$$\mathcal{B}(x,t,y,\theta) = 0 \qquad \text{on } \Gamma \times T, \tag{2}$$

$$y = y_0 \qquad \text{in } \Omega \times \{t = 0\}, \tag{3}$$

respectively, $\mathcal{B}$ being an operator acting on the boundary Γ and $y_0 = y_0(x)$ a given function. Conditions (2) and (3) complement (1) so that the existence of a sufficiently smooth and unique solution is guaranteed. We assume that the forms of $\mathcal{F}$ and $\mathcal{B}$ are given explicitly up to an m-dimensional vector of unknown constant parameters θ which must be estimated using observations of the system. The implicit dependence of the state y on the parameter vector θ will be be reflected by the notation $y(x,t;\theta)$.

In what follows, we consider the discrete-continuous observations provided by n mobile sensors, namely,

$$z_{\mathrm{m}}^{\ell}(t) = y(x^{\ell}(t),t;\theta) + \varepsilon(x^{\ell}(t),t), \quad t \in T, \tag{4}$$

where $z_{\mathrm{m}}^{\ell}(\,\cdot\,)$ is the scalar output and $x^{\ell}(\,\cdot\,) \in X$ stands for the trajectory of the ℓ-th sensor ($\ell = 1,\ldots,n$), X signifies the part of the spatial domain Ω where the measurements can be made, and $\varepsilon(x^{\ell},t)$ denotes the measurement noise. The assumption that we are in a position to observe directly the system state is made only for simplicity of presentation. The approach outlined in what follows can easily be generalized to indirect observation of state variables.

Designers are often tempted to assume that the measurement noise is zero-mean, Gaussian, spatial uncorrelated and white (Quereshi *et al*., 1980; Kubrusly and Malebranche, 1985; Sun, 1994). But one of the characteristic properties of collecting spatial data is the presence of spatial correlations between observations made at different sites (Müller, 2007; Le and Zidek, 2006; Cressie, 1993; Fedorov and Hackl, 1997). This topic is, however, neglected in most works on sensor location for parameter estimation in dynamic DPSs. On the one hand, such a simplification is very convenient as it leads to elegant theoretical results, but on the other hand, it is rarely justified in a large number of applications. Environmental monitoring, meteorology, surveillance, some industrial experiments and seismology are the most typical areas where the necessity for taking account of this factor may emerge. Consequently, in what follows we assume that $\varepsilon(x,t)$ is a Gaussianly distributed measurement disturbance satisfying

$$\mathrm{E}\big[\varepsilon(x,t)\big] = 0, \tag{5}$$

$$\mathrm{E}\big[\varepsilon(x,t)\varepsilon(\chi,\tau)\big] = q(x,\chi,t)\delta(t-\tau), \tag{6}$$

$q(\,\cdot\,,\,\cdot\,,t)$ being a known continuous spatial covariance kernel and δ the Dirac delta function.

In this framework, the parameter identification problem is formulated in terms of minimizing the corresponding weighted output least-squares fit-to-data functional, see (Uciński, 2005) for details. Clearly, the resulting parameter estimate $\hat{\theta}$ depends on sensor trajectories $x^{\ell}(\,\cdot\,)$, which suggests that we may attempt to select the trajectories which lead to best estimates of the system parameters. To form a basis for the comparison of different trajectories, a quantitative measure of the 'goodness' of particular trajectories is required. A logical approach is to choose a measure related to the expected accuracy of the parameter estimates to be obtained from the data collected. Such a measure is usually based on the concept of the *Fisher Information Matrix* (FIM), which is widely used in optimum experimental design theory for lumped systems (Atkinson *et al*., 2007; Fedorov and Hackl, 1997; Pázman, 1986; Pukelsheim, 1993; Walter and Pronzato, 1997). In our setting, the FIM is given by (Uciński, 2005)

$$M(x^1,\ldots,x^n) = \sum_{i=1}^{n}\sum_{j=1}^{n}\int_T w_{ij}(t)g(x^i(t),t)g^{\mathsf{T}}(x^j(t),t)\,\mathrm{d}t, \tag{7}$$

where

$$g(x(t),t) = \left[\frac{\partial y(x(t),t;\vartheta)}{\partial \vartheta_1}, \quad \ldots, \quad \frac{\partial y(x(t),t;\vartheta)}{\partial \vartheta_m}\right]^{\mathsf{T}}_{\vartheta=\theta^0} \tag{8}$$

stands for the so-called *sensitivity vector*, θ^0 being a prior estimate to the unknown parameter vector θ (Uciński, 2005; Sun, 1994; Rafajłowicz, 1981; Rafajłowicz, 1983). Furthermore, we set

$$W(t) = \big[w_{ij}(t)\big] = C^{-1}(t), \tag{9}$$

$$C(t) = \big[c_{ij}(t)\big], \tag{10}$$

$$c_{ij}(t) = q(x^i(t),x^j(t),t). \tag{11}$$

Under some mild assumptions, the inverse of the FIM constitutes an approximation of the covariance matrix for the estimate of θ (Walter and Pronzato, 1997; Fedorov and Hackl, 1997).

The selection of the best measurement trajectories then rests on maximizing a scalar measure defined on the FIM. As for a specific form of Ψ, various choices exist (Atkinson *et al*., 2007; Walter and Pronzato, 1997; Fedorov and Hackl, 1997), but the most popular criterion, called the D-optimality criterion, is the log-determinant of the FIM:

$$\Psi(M) = \log\det(M). \tag{12}$$

The resulting D-optimum sensor configuration leads to the minimum volume of the uncertainty ellipsoid for the estimates.

The introduction of an optimality criterion renders it possible to formulate the sensor location problem as the maximization of the performance measure

$$\mathcal{R}(x^1,\dots,x^n) := \Psi\big[M(x^1,\dots,x^n)\big] \tag{13}$$

with respect to $x^\ell(\,\cdot\,)$, $\ell = 1,\dots,n$ belonging to the admissible set X. This apparently simple formulation may lead to the conclusion that the only question remaining is that of selecting an appropriate solver from a library of numerical optimization routines. Unfortunately, a careful analysis reveals dark sides of this naive way of thinking.

A major problem to address when attempting to plan optimal sensor paths regards the selection of their proper discretization. But even when applying a parsimonious approximation scheme, we must bear in mind that in sensor network settings encountered in practice we may have dozens of sensors and therefore the dimensionality of the optimization problem may be high (more than 100 dimensions is a rule). What is more, a desired global extremum is usually hidden among many poorer local extrema. Consequently, to directly find a numerical solution may be extremely difficult. These impediments make the recourse to large-scale global optimization techniques rather necessary.

2.2. Parametrization of sensor trajectories. Without loss of generality, from now on we make the assumption that we have only $d = 2$ spatial dimensions. In order to make our design problem finite-dimensional, we introduce a parametrization of the trajectories. Such a discretization becomes a necessity if we wish to make the design problem tractable (Sacks, 1998). Clearly, the selection of the appropriate finite-dimensional subspace affects both the accuracy of numerical integration and the accuracy with which the solutions of the original problem are approximated. But a thorough analysis of this choice falls far beyond the scope of this paper. In our approach, given a desired number of basis functions $p > 3$, we choose the set of discretization points $t_i = ih$, $i = 0,\dots,p-3$ for $h = t_f/(p-3)$, and for each $\ell = 1,\dots,n$ we expand admissible trajectories as linear combinations of B-splines of order three (Quarteroni *et al*., 2000):

$$x^\ell(t) = (x_1^\ell(t), x_2^\ell(t)), \tag{14}$$

$$x_1^\ell(t) = \sum_{i=1}^{p} \omega_{1i}^\ell \phi_i(t), \quad x_2^\ell(t) = \sum_{i=1}^{p} \omega_{2i}^\ell \phi_i(t), \tag{15}$$

where for each $i = 1,\dots,p$ we define

$$\phi_i(t) = g\left(\frac{t}{h} - i + 2\right), \tag{16}$$

$$g(\tau) = \begin{cases} \frac{2}{3} - \frac{1}{2}|\tau|^2(2-|\tau|) & \text{if } 0 \le |\tau| < 1, \\ \frac{1}{6}(2-|\tau|)^3 & \text{if } 1 \le |\tau| < 2, \\ 0 & \text{if } 2 \le |\tau|. \end{cases} \tag{17}$$

Our choice was dictated by the fact that cubic B-splines offer C^2-regularity and have excellent approximation properties.

In cosequence, the motions of all sensors are fully described by the vector of $2np$ coefficients

$$\begin{aligned} w = (&\omega_{11}^1, \omega_{21}^1, \dots, \omega_{1p}^1, \omega_{2p}^1, \dots \\ &\omega_{11}^n, \omega_{21}^n, \dots, \omega_{1p}^n, \omega_{2p}^n), \end{aligned} \tag{18}$$

which become our new decision variables. The ultimate form of the performance index is

$$\mathcal{J}(w) = \Psi[M(x(\,\cdot\,;w))], \tag{19}$$

where the notation $x(w)$ expresses the dependence of the trajectories on w through (14), (15) and (18).

Remark 1. As has already been mentioned, all sensor trajectories must remain within the admissible region X. This constitutes a constraint on the allowable values of the components of the vector w. If X is a square, i.e., $X = [x_{1\,\min}, x_{1\,\max}] \times [x_{2\,\min}, x_{2\,\max}]$, then the constraint set $\mathcal{W} = \big\{w \mid x^\ell(t;w) \in X, \forall t \in T\big\}$ is a box (i.e., $\mathcal{W} = \prod_{\ell=1}^{n}\prod_{k=1}^{p}[x_{1\,\min}, x_{1\,\max}] \times [x_{2\,\min}, x_{2\,\max}]$) which can be handled with relative ease. In more complicated cases, in order to avoid resorting to constrained optimization, we take account of them by adding to (19) an appropriate quadratic loss penalty function (Papalambros and Wilde, 2000). For example, if $X = \big\{x \in \bar{\Omega} : b_i(x) \le 0,\ i = 1,\dots,I\big\}$, where $b_i(\,\cdot\,)$ are some given functions, then the barrier function could be

$$P(x(\,\cdot\,;w)) = -c\sum_{\ell=1}^{n}\sum_{i=1}^{I}\int_T \big[\max(0, b_i(x^\ell(t;w)))\big]^2\,\mathrm{d}t \tag{20}$$

for some constant $c > 0$.

3. Solution by the tunneling algorithm

3.1. Outline. As a global optimizer to produce solutions to our trajectory design problem for which many local maxima and many design variables are a general rule, we selected the tunneling algorithm set forth in (Levy and Montalvo, 1985). Basically, it consists of two phases: a local maximization phase and a tunneling phase. These phases are alternately used to approximate a global maximizer of $\mathcal{J}(w)$. In the maximization phase, given a starting point w^0, we can use any nonlinear programming algorithm for local optimization to find a local maximum $w^\star$ of $\mathcal{J}(\,\cdot\,)$. In the tunneling phase, we introduce an auxiliary function $\mathcal{T}(w)$ which should be continuously differentiable. Starting from some point in a neighbourhood of $w^\star$, the role of this function is to help in producing a new point $\bar{w}$ such that $\mathcal{J}(\bar{w}) \geq \mathcal{J}(w^\star)$. If used properly, these two phases yield a sequence of local maximizers such that the function value at each of these maximizers is no less than any of the function values at the previously found maximizers.

3.2. Algorithm details. We formulate our problem as a classical maximization problem, i.e., we wish to find a global maximum $w_G^\star$ of

$$\max \mathcal{J}(w) \quad \text{s.t. } w \in \mathcal{W}, \tag{21}$$

where $\mathcal{W} = \big\{w \mid x^\ell(t;w) \in X,\, \forall t \in T\big\}$. At the global solution, we have

$$\mathcal{J}(w) \leq \mathcal{J}(w_G^\star), \qquad \forall w \in \mathcal{W}. \tag{22}$$

In order to find $w_G^\star$, a sequence of local maxima is constructed with nondecreasing function values, i.e., $\mathcal{J}(w^{\star(1)}) \leq \mathcal{J}(w^{\star(2)}) \leq \cdots \leq \mathcal{J}(w_G^\star)$, while ignoring all local maxima with objective function values lower than the best one found so far (it is here that the tunneling phase plays its key role).

Having found a local maximum $w^\star$ in the maximization phase, the tunneling phase uses a starting point $\tilde{w}$, which belongs to a vicinity of $w^\star$, to find a point w^{tun} satisfying

$$\mathcal{J}(w^{\mathrm{tun}}) \geq \mathcal{J}(w^\star) \tag{23}$$

and differing from the hitherto discovered local maxima. The point w^{tun} will be taken as the initial point w^0 for the next maximization phase.

3.2.1. Local maximization phase. In our work, in order to solve the problem

$$w^\star = \arg \max_{w \in \mathcal{W}} \mathcal{J}(w), \tag{24}$$

we use the coordinate ascent method (Bertsekas, 1999, p. 160). It is a nonderivative method for maximizing differentiable functions. Here the criterion is maximized along one coordinate direction at each iteration. The order in which coordinates are chosen may vary in the course of the algorithm. The method can also be used for the maximization of $\mathcal{J}$ subject to upper and lower bounds on the components of w, which is useful when the admissible set X is a rectangle. Note that the method generally has similar convergence properties to steepest ascent.

3.2.2. Tunneling phase. Principally, to solve the problem of finding a point w^{tun} such that

$$\mathcal{J}(w^{\mathrm{tun}}) \geq \mathcal{J}(w^\star) \tag{25}$$

means solving the inequality

$$\mathcal{J}(w) - \mathcal{J}(w^\star) \geq 0, \quad w \in \mathcal{W}, \tag{26}$$

but on the condition of avoiding the hitherto found local maxima. This is accomplished by introducing a tunneling function $\mathcal{T}$ which desirably would help us to find a novel maximum in another valley, and then solving the inequality

$$\mathcal{T}(w) \geq 0 \tag{27}$$

instead. In the literature (Levy and Montalvo, 1985; Gómez *et al*., 2003), the *exponential tunneling function*

$$\mathcal{T}_e(w) = (\mathcal{J}(w) - \mathcal{J}^\star) \exp\left(\frac{\lambda^\star}{\|w - w^\star\|}\right) \tag{28}$$

or the *classical tunneling function*

$$\mathcal{T}_c(w) = \frac{\mathcal{J}(w) - \mathcal{J}^\star}{\|w - w^\star\|^{\lambda^\star}} \tag{29}$$

were proposed as candidates for $\mathcal{T}$, where $\|\,\cdot\,\|$ signifies the Euclidean norm and $\lambda^\star$ defines the strength of the pole $w^\star$.

If a large value of $\lambda^\star$ is adopted, the tunneling function will be smoother and the danger of encountering critical points (i.e., undesirable local maxima or saddle points) during the search will be reduced. Note, however, that a large value of $\lambda^\star$ may lead to unnecessarily expensive computations.

To solve the inequality (27), we can use the same algorithm as in the minimization phase. For that purpose, in this work we therefore adopted the coordinate ascent method.

The function $\mathcal{J}$ may have many local maxima and, what is more, convergence to maxima at the same level is possible, that is, $\mathcal{J}(w^{\star(1)}) = \mathcal{J}(w^{\star(2)}) = \cdots = \mathcal{J}(w^{\star(t)})$, as $w^{\star(i)}$ would be acceptable solutions to the problem (27). To avoid cycling and going back to these maxima at the same level already found, it is important to store the local maxima, until a maximum $w^{\star(k+1)}$ with a greater value of the objective function is found. To realize

this goal, it is necessary to modify the definition of the tunneling function. This can be achieved by defining

$$\mathcal{T}_e(w) = (\mathcal{J}(w) - \mathcal{J}^\star) \prod_{i=1}^{k} \exp\left(\frac{\lambda_i^\star}{\|w - w_i^\star\|}\right) \quad (30)$$

and initializing $k = 1$ as soon as a point $w^{\star(k+1)}$ is found with a better value of the objective function than $\mathcal{J}(w^\star)$.

As soon as a local maximum of $\mathcal{J}(w)$ has been found, we have to generate an initial point $w_0^{\rm tun}$ to start the tunneling search. We find this point in a neighbourhood of the point $w^\star$. The distance between the points $w_0^{\rm tun}$ and $w^\star$ depends on the stage of the tunneling phase. In the first stage of the tunneling phase this distance is relatively small so we look for a new tunnel near the maximum that we have just found. In the second stage, the distance between the points is larger and we can find a new tunnel further than the nearest neighborhood. A new point $w_0^{\rm tun}$ is generated along a random direction. Parallelizing this phase is described in Section 4. The numbers of iterations in the first stage ($iter_1$) and the second stage ($iter_2$) are defined at the start of the algorithm.

3.2.3. Stopping conditions for the tunneling phase. The tunneling phase is terminated without finding a new tunnel when the maximum number of function evaluations allowed for this phase has been reached. It is also finished when the boundary of the admissible region $\mathcal{W}$ has been reached or the strength of the pole $\lambda^\star$ is greater than a given maximum value.

3.2.4. General stopping conditions. The algorithm stops when in the tunneling phase a given maximum number of initial points to start the search for $w^{\rm tun}$ has been reached. Before we start the algorithm, we define a maximum number of initial points to start the search for $w^{\rm tun}$. When the tunneling phase is unsuccessfull, we decrease the counter of the remaining iterations. Once the tunneling phase is successful, we set the counter to the initial value so that the number of iterations is the same as at the beginning.

Remark 2. In our implementation, we used a solution to a system of partial differential equations given on a finite grid of spatio-temporal points. In turn, the tunneling algorithm is a continuous optimization one, so the appropriate interpolation procedure for data between the grid nodes is necessary. Some approach towards it is exposed in Appendix G.2 of (Uciński, 2005). Here we used linear interpolation in time and bilinear interpolation in space (Press *et al.*, 1992).

4. Parallel tunneling algorithm

Before parallelizing the tunneling algorithm, we have to decide which scheme of parallelization will be implemented and which phase will be parallelized. In this work, we assume that we have a master-slave scheme where one of the cluster nodes perform a supervisory role while the other nodes are slaves and perform all computations. Thus our approach is very similar to that proposed in (Gómez *et al.*, 2003).

We decide to parallelize the tunneling phase because the parallelization of this part of the algorithm is very natural and will produce the biggest decrease in the computation time. If we parallelized the minimization phase, cluster nodes would exchange a lot of information, and therefore the overall speedup would be lower because of data transmission delay.

In the tunneling phase we seek a point $w^{\rm tun}$ in another valley such that

$$\mathcal{J}(w^{\rm tun}) \geq \mathcal{J}(w^\star). \quad (31)$$

To this end, we maximize the tunneling function along random directions formed as (Gómez *et al.*, 2003)

$$w = w^\star + e(\varsigma), \quad (32)$$

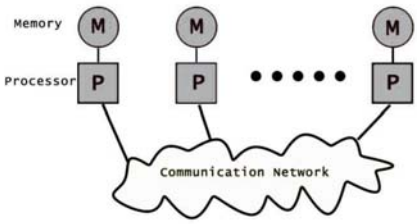
where $e(\,\cdot\,)$ is a function guaranteeing $w \in \mathcal{W}$ and ς constitutes a random vector.

Each slave node is supposed to maximize the tunneling function along a different random direction. To make this possible before the tunneling algorithm starts, the master node sets at random the initial seeds for random number generators at individual nodes.

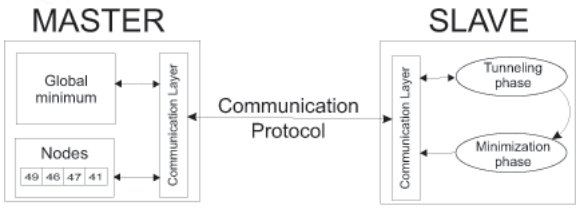
To implement a parallel version of the tunneling algorithm, we use an MPI (Message Passing Interface) environment (Scott *et al.*, 2005). MPI is a defining feature of the message passing model in which data transfer from the local memory of one process to the local memory of another requires operations to be performed by both processes (cf. Fig. 1(a)). There are many specific communication networks, e.g., Fast/GigaBit Ethernet, Infiniband or Myrinet, with different connection topologies. In practice, MPI is a communication library for both parallel computers and workstation networks, especially for Unix/Linux clusters.

4.1. Idea of master and slaves. In our work we defined two types of cluster nodes. The *master* is a node which stores the current approximation to the global maximum and other data. It also sends information to slave nodes. The master works in a loop and waits for requests from slaves. The loop ends when the tunneling algorithm has to be finished.

The *slaves* are nodes which run the tunneling algorithm using data from the master node. The slaves implement the tunneling phase and, after finding a tunnel, the minimization phase starts. After determining a maximum they send these data to the master and wait for data for the next iteration.



(a)



(b)

Fig. 1. Parallel environment: MPI model (a) and master-slave tasks (b).

The master exchanges data with slaves using operation codes. The corresponding operational codes describe what kind of operation is requested, as, e.g., OPER_SEND_GLOBAL_MIN_TO_MASTER, OPER_LET_NEXT_COUNTING, etc. When a slave node demands the current approximation to the global minimum, it sends a request with the appropriate operation code to the master, and the master acts on it. When the maximum number of iterations of the tunneling algorithm has been reached, the master sends a 'finish computations' command to slaves and announces the global maximum found as a result of computing.

4.2. Modifications. When a node finds a new maximum and sends it to the master, the latter sends this information to all nodes. All nodes check information from the master node in the tunneling phase. If the time of evaluating $\mathcal{J}(w)$ is short, so are the times of evaluating (28) and (29). Consequently, a slave node may check information from the master quite often. This means that the parallel tunneling algorithm is very flexible and can react very fast to an updated approximation to the global maximum.

As far as the problem of searching for optimal trajectories of mobile sensors for correlated observations is concerned, we must realize that the time of evaluating $\mathcal{J}(w)$ may sometimes reach several seconds (depending on the discretization and the number is sensors). To efficiently evaluate the determinant of an FIM whose entries depend on very time-consuming functions may be hard. When the optimization method calls this function very often (hundreds or thousands of times), the reaction time of the algorithm to an update in the approximation to the global maximum is very long. To avoid this situation, in the coordinate ascent method we implement a check for a new piece of information from the master. This modification makes it possible to avoid very long latency times on a slave node when another node finds a new approxmation to the global maximum.

4.3. Information exchange scheme. The parallelization of the tunneling algorithm is based on the idea of two types of nodes: a master and slaves. Each slave node runs the tunneling algorithm independently of other nodes and exchanges information with the master. The role of the master node is to store global data and share information about control parameters with slaves. A general scheme of the master node is reflected by Algorithm 1, and a detailed scheme of the actions by slave nodes is embodied by Algorithm 2.

At the beginning, the master node sets at random initial seeds for the random generators in slaves. Afterwards, the master initiates a loop in which it waits for actions from slaves and, as an answer, passes desired information. Upon receiving an initial seed from the master, each slave node starts a maximization algorithm for a random initial point. The maximum found is sent to the master node and the slave is waiting for an approximation to the global maximum. At this time, the master node waits for every slave node to receive an initial local maximum. After receiving all local maxima, the master determines an initial global optimum and sends this initial global maximum to each slave node. Simultaneously, slaves receive vectors ς which determine a neighbourhood where to start the tunneling phase. The size of the neighbourhood depends on the stage of the tunneling algorithm. In the first stage the vector ς is relatively small and the tunneling algorithm explores the neighbourhood of w. In the second stage we wish to explore the whole admissible region, which is attained by appropriately generating $e(\varsigma)$.

The master node stores an array which contains iteration counters per node and maximum numbers of iterations per part for every node. If the current number of iterations for one node is greater than the defined maximum number of iterations, then this node will be given another vector ς. An important assumption is that the iteration counter for a node is set to zero if this node finds a new global maximum. Summarizing, the parallel method implements the following rules:

- There is a master node (processor) that controls the process and broadcasts necessary information to all slave nodes.
- Each slave node will carry out both phases, tunnelization and maximization, and will have different seeds for the random number generator.

Algorithm 1 Master node scheme.

```
1: procedure MASTER
2:     INITIALIZE RANDOM SEEDS FOR SLAVE NODES
3:     SEND SEEDS TO SLAVE NODES
4:     repeat
5:         WAITFOROPERATION(operation,whichNode)
6:         PERFORMOPERATION(operation,whichNode)
7:     until (iter > max_iter)
8: end procedure
```

- The tunnelization phase will start from different initial points (along different random directions) at each processor.
- At the start each slave node runs the maximization phase to find an initial approximation to the local maximum.
- When a slave node finds a point in another valley (a successful tunnelization), it proceeds to find a local maximum.
- When a slave node finds a local maximum, it sends the result immediately to the master node.
- The master node checks if the new maximum is the best one found so far in which case it proceeds to sending a message with this information to all processors. Additionally, the master node resets iteration counters for each slave node.
- Each slave node checks if there is a message from the master node only during the tunneling phase. If it is already in a maximization phase, it continues this phase until it finds a local maximum, without checking messages from the master node.
- Each slave node checks if there is a message from the master node during the coordinate ascent method in the tunneling phase, cf. Section 4.2.
- When a slave node checks for a message at the tunneling phase, if it gets a new maximum, it re-starts the tunneling search from this new maximum.
- The master node checks the iteration counter for each slave node and sends the corresponding vectors ς.
- The master node checks the general stopping conditions and sends information to a slave to stop computing.

5. Computational results

The following example demonstrates advantages of applying the tunneling algorithm. Consider the diffusion process with two mobile sources over a spatial domain $\Omega = (0,1)^2$. The trajectories of the mobile sources are described by two functions:

$$s_1(x,t) = q^1 + v^1 t, \tag{33}$$

$$s_2(x,t) = q^2 + v^2 t, \tag{34}$$

where

$$q^1 = (0.095, 0.523), \quad v^1 = (0.00999, 0.0047), \tag{35}$$

$$q^2 = (0.523, 0.095), \quad v^2 = (0.0047, 0.00999). \tag{36}$$

The mobile sources generate an air pollutant in the following manner:

$$f_1(x,t) = 0.45\exp(-10.0\|x - s_1(x,t)\|), \tag{37}$$

$$f_2(x,t) = 0.45\exp(-10.0\|x - s_2(x,t)\|). \tag{38}$$

The evolution of the pollutant concentration $y = y(x,t)$ over the normalized observation interval $T = (0,1)$ and the spatial domain $\Omega = (0,1)^2$ is described by the diffusion equation

$$\begin{aligned}\frac{\partial y(x,t)}{\partial t} = &\frac{\partial}{\partial x_1}\left(\kappa(x)\frac{\partial y(x,t)}{\partial x_1}\right)\\ &+ \frac{\partial}{\partial x_2}\left(\kappa(x)\frac{\partial y(x,t)}{\partial x_2}\right) + f(x,t),\end{aligned} \tag{39}$$

subject to the boundary and initial conditions:

$$\frac{\partial y(x,t)}{\partial n} = 0 \qquad \text{on } \Gamma \times T, \tag{40}$$

$$y(x,0) = 0 \qquad \text{in } \Omega, \tag{41}$$

where $f(x,t) = f_1(x,t) + f_2(x,t)$, and $\partial y/\partial n$ stands for the partial derivative of y with respect to the outward normal to the boundary Γ. The assumed functional form of the spatial-varying diffusion coefficient is

$$\kappa(x) = \theta_1 + \theta_2 x_1 + \theta_3 x_2, \tag{42}$$

so that the constant parameters θ_1, θ_2 and θ_3 need estimation based on measurement data from two mobile sensors. The observations from the sensors are assumed to be correlated so that

$$c_{ij} = \sigma^2 \exp\left(-\beta\|x^i - x^j\|\right), \tag{43}$$

Algorithm 2 Slave node scheme.

```
1: procedure SLAVE
2:     RECEIVE RANDOM SEED FROM MASTER
3:     w = MINIMIZING ALGORITHM()
4:     SEND LOCAL MINIMUM TO MASTER(w)
5:     loop
6:         w = RECEIVE ACTUAL GLOBAL MINIMUM FROM MASTER
7:         RECEIVE ς FROM MASTER
8:         loop
9:             w^tun = TUNNEL MINIMIZATION()
10:            if (J(w^tun) < J(w)) then
11:                w^0 = MINIMIZATION ALGORITHM()
12:                SEND LOCAL MINIMUM TO MASTER(w^0)
13:                EXIT LOOP
14:            else
15:                w^0 = RECEIVE ACTUAL GLOBAL MINIMUM FROM MASTER()
16:                if ( J(w^0) < J(w) ) then
17:                    EXIT LOOP
18:                else
19:                    operation = SEND REQUEST FOR CONTINUING COUNTING()
20:                    if (operation == stopCounting) then
21:                        order = WAITING FOR ADDITIONAL ORDER()
22:                        if (order == endCounting) then
23:                            EXIT PROCEDURE SLAVE
24:                        end if
25:                        if (order == continueCounting) then
26:                            EXIT LOOP
27:                        end if
28:                    end if
29:                    if (operation == continueCounting) then
30:                        RECEIVE ς FROM MASTER
31:                    end if
32:                end if
33:            end if
34:        end loop
35:    end loop
36: end procedure
```

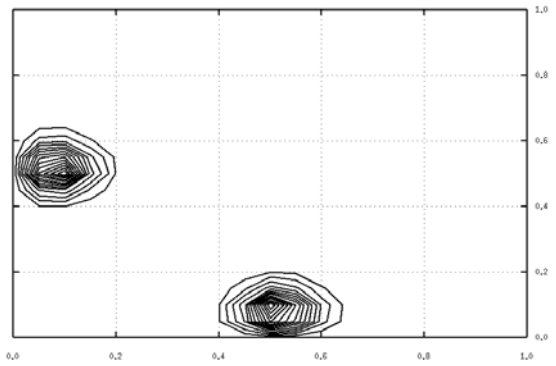
where $\beta = 0.2$.

Since we use local designs (Uciński, 2005), we have to set nominal values for the vector θ. Thus we assumed $\theta_1^0 = 0.01, \theta_2^0 = \theta_3^0 = 0.005$. The solution to (33)–(42) is displayed in Fig. 2, where the two mobile sources can be observed.

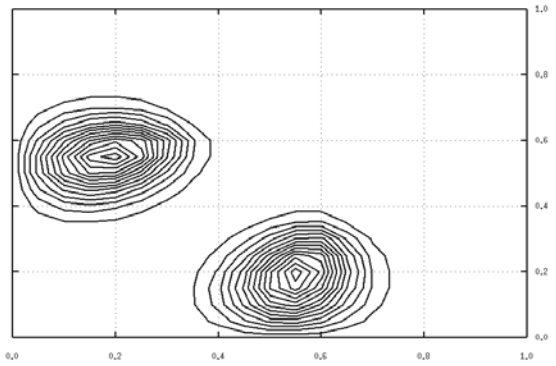
We solved the PDE (39) using a separate program. This application solves PDEs using the finite-element method and uses a parallel MPI environment to speed up a computing. We divided the spatial domain to 441 (21×21) blocks. Numerical integration required to evaluate the FIM was performed employing the trapezoidal rule for the time step equal to $1/80$, based on the sensitivity vector g interpolated at the nodes representing admissible locations x^i.

The program used for parallel solution of the discussed problem was written completely in Fortran 95 using *ifort* (Intel®Fortran Compiler v.8.1 for Linux 64-bit platforms) and the *mpich-1.2.6* implementation of MPI for message passing (Pacheco, 1997). The program uses Intel Math Kernel Library for matrix and vector operations. Computations were performed on a Linux cluster at the University of Zielona Góra, being part of the national Clusterix project (Wyrzykowski *et al*., 2004). This homogenous cluster is equipped with four SMP nodes with two 64-bit *Intel Itanium*^{2} 1.4GHz processors each, running under the control of GNU/Linux Debian for *ia64*. The connection between the nodes is realized via Gigabit Ethernet.

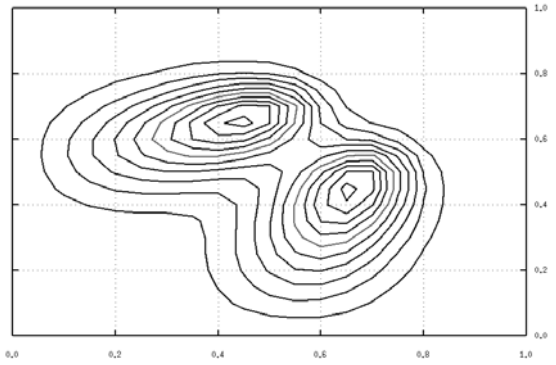
The D-optimal sensors trajectories are shown in Fig. 3. As was expected, the sensors follow the mobile pollutant sources. Each computational node started with



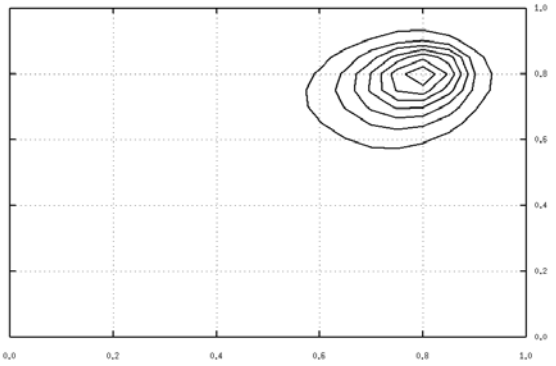
(a) $t = 0.1$



(b) $t = 0.4$



(c) $t = 0.7$



(d) $t = 1.0$

Fig. 2. Concentration of the pollutant at consecutive time instants.

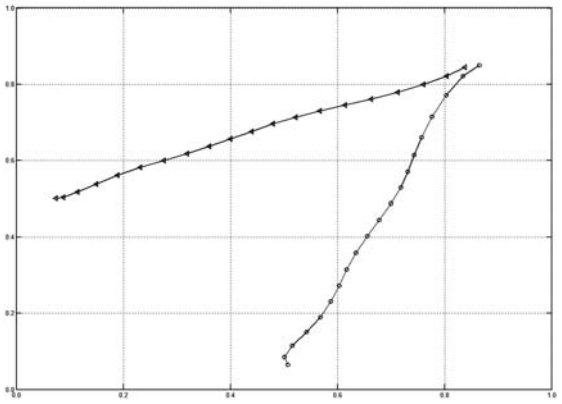


Fig. 3. D-optimal trajectories.

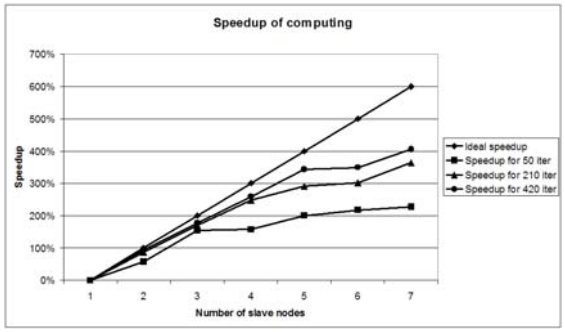


Fig. 4. Computation speedup.

randomly generated initial trajectories of moving sensors. The maximum number of iterations for the coordinate ascent method was set to 100.

The time of computations depends on the number of processors and iterations in the tunneling phase as shown in Table 1. Figure 4 displays the speedup of computing. A slowdown can be observed if computations involve more than three computational nodes. This bottleneck is likely to result from the architecture of the computational cluster. The computations were performed on four machines with two *Intel Itanium*^{2} processors. There may be a conflict in access to resources by each processor on each node. The resources of each node are shared between processors and this situation may cause a latency in access to data. If computations are performed on one, two or three nodes, the cluster splits the application such that one processor performs computations on each node. In this situation the speedup is almost linear. Latency is caused by a slight delay in transmission.

6. Conclusions

The problem of optimal observations for distributed parameter systems has been attacked from various angles since the mid-1970s. In this paper we formulated the problem of optimum trajectory design for mobile sensors as a multidi-

Table 1. Computation times for the proposed example [hh : mm : ss].

$iter_1$	$iter_2$	Number of computational processors						
		1	2	3	4	5	6	7
50	50	00:01:23	00:00:42	00:00:33	00:01:23	00:00:42	00:00:33	00:00:33
210	210	02:30:42	01:20:14	00:55:55	00:43:31	00:38:38	00:37:41	00:32:45
420	420	04:59:20	02:33:20	01:47:45	01:23:12	01:07:45	01:06:38	00:59:18

mensional optimization one. Apart from the fact that multidimensional optimization involves complications *per se*, the problem of optimal observation selection imposes additional high computational requirements. The objective function may have very complicated forms and determining their values may be very time consuming. Owing to the rapidly emerging computational environments, not least clusters and grids, in this paper we made an attempt to solve the problem of optimal observation design for mobile sensor networks by converting it to a multidimensional nonlinear programming problem to be solved using a parallel version of the tunneling algorithm. Simulation results demonstrate that the optimal observation problem has thus become much more affordable.

Note that no physical models of the vehicles conveing sensors were assumed. This implies that the designed trajectories of sensors do not depend on weight, size, acceleration or maximum speed of the vehicles. In the future, we are going to extend this approach to a formulation taking account of vehicle dynamics and kinematics.

Acknowledgment

The work of the first author was supported by the Integrated Regional Operational Programme (Measure 2.6: Regional innovation strategies and the transfer of knowledge) co-financed from the European Social Fund.

References

Atkinson A. C., Donev A. N. and Tobias R. D. (2007). *Optimum Experimental Designs, with SAS*, Oxford University Press, Oxford.

Banks H. T., Smith R. C. and Wang Y. (1996). *Smart Material Structures: Modeling, Estimation and Control*, Masson, Paris.

Bertsekas D. P. (1999). *Nonlinear Programming*, *2nd Edn.,* Athena Scientific, Belmont, MA.

Cassandras C. G. and Li, W. (2005). Sensor networks and cooperative control, *European Journal of Control* **11**(4–5): 436–463.

Christofides P. D. (2001). *Nonlinear and Robust Control of PDE Systems: Methods and Applications to Transport-Reaction Processes*, Birkhäuser, Boston, MA.

Cressie N. A. C. (1993). *Statistics for Spatial Data*, Revised Edn., John Wiley & Sons, New York, NY.

Daescu D. N. and Navon I. M. (2004). Adaptive observations in the context of 4D-Var data assimilation, *Meteorology and Atmospheric Physics* **85**(4): 205–226.

Demetriou M. A. (2006a). Detection and containment policy of moving source in 2D diffusion processes using sensor/actuator network, *Proceedings of the European Control Conference 2007,* Kos, Greece. Published on CD-ROM.

Demetriou M. A. (2006b). Power management of sensor networks for detection of a moving source in 2-D spatial domains, *Proceedings of the 2006 American Control Conference,* Minneapolis, MN, USA. Published on CD-ROM.

Demetriou M. A. (2007). Process estimation and moving source detection in 2-D diffusion processes by scheduling of sensor networks, *Proceedings of the 2007 American Control Conference,* New York, NY, USA. Published on CD-ROM.

Fedorov V. V. and Hackl P. (1997). *Model-Oriented Design of Experiments*, Lecture Notes in Statistics, Springer-Verlag, New York, NY.

Gómez S., del Castillo N., Castellanos L. and Solano J. (2003). The parallel tunneling method, *Parallel Computing* **29**(4): 523–533.

Hussein I. I. and Demetriou M. A. (2007). Estimation of distributed processes using mobile spatially distributed sensors, *Proceedings of the 2007 American Control Conference,* New York, NY, USA. Published on CD-ROM.

Jeremić A. and Nehorai A. (1998). Design of chemical sensor arrays for monitoring disposal sites on the ocean floor, *IEEE Transactions on Oceanic Engineering* **23**(4): 334–343.

Jeremić A. and Nehorai A. (2000). Landmine detection and localization using chemical sensor array processing, *IEEE Transactions on Signal Processing* **48**(5): 1295–1305.

Kubrusly C. S. and Malebranche H. (1985). Sensors and controllers location in distributed systems — A survey, *Automatica* **21**(2): 117–128.

Le N. D. and Zidek J. V. (2006). *Statistical Analysis of Environmental Space-Time Processes*, Springer-Verlag, New York, NY.

Levy A. V. and Montalvo A. (1985). The tunneling algorithm for the global minimization of functions, *SIAM Journal on Scientific and Statistical Computing* **6**(1): 15–29.

Müller W. G. (2007). *Collecting Spatial Data. Optimum Design of Experiments for Random Fields*, *3rd Revised and Extended Edn.,* Physica-Verlag, Heidelberg.

Navon I. M. (1997). Practical and theoretical aspects of adjoint parameter estimation and identifiability in meteorology and oceanography, *Dynamics of Atmospheres and Oceans* **27**(1): 55–79.

Nehorai A., Porat B. and Paldi E. (1995). Detection and localization of vapor-emitting sources, *IEEE Transactions on Signal Processing* **43**(1): 243–253.

Pacheco P. S. (1997). *Programming parallel with MPI*, Morgan Kaufmann, San Francisco, CA.

Papalambros P. Y. and Wilde D. J. (2000). *Principles of Optimal Design. Modeling and Computation*, *2nd Edn.,* Cambridge University Press, Cambridge.

Pázman A. (1986). *Foundations of Optimum Experimental Design*, D. Reidel Publishing Company, Dordrecht.

Porat B. and Nehorai A. (1996). Localizing vapor-emitting sources by moving sensors, *IEEE Transactions on Signal Processing* **44**(4): 1018–1021.

Press W. H., Teukolsky S. A., Vetterling W. T. and Flannery B. P. (1992). *Numerical Recipes in FORTRAN. The Art of Parallel Scientific Computing*, *2nd Edn.,* Cambridge University Press, Cambridge.

Pukelsheim, F. (1993). *Optimal Design of Experiments*, John Wiley & Sons, New York, NY.

Quarteroni A., Sacco R. and Saleri F. (2000). *Numerical Mathematics*, Springer-Verlag, New York, NY.

Quereshi Z. H., Ng T. S. and Goodwin G. C. (1980). Optimum experimental design for identification of distributed parameter systems, *International Journal of Control* **31**(1): 21–29.

Rafajłowicz E. (1981). Design of experiments for eigenvalue identification in distributed-parameter systems, *International Journal of Control* **34**(6): 1079–1094.

Rafajłowicz E. (1983). Optimal experiment design for identification of linear distributed-parameter systems: Frequency domain approach, *IEEE Transactions on Automatic Control* **28**(7): 806–808.

Rafajłowicz E. (1986). Optimum choice of moving sensor trajectories for distributed parameter system identification, *International Journal of Control* **43**(5): 1441–1451.

Sacks E. W. (1998). Semi-infinite programming in control, *in* R. Reemtsen and J.-J. Rückmann (Eds.), Semi-Infinite Programming, Kluwer Academic Publishers, Boston, MA, pp. 389–411.

Scott L. R., Clark T. and Bagheri B. (2005). *Scientific Parallel Computing*, Princeton University Press, Princeton, NJ.

Sun N.-Z. (1994). *Inverse Problems in Groundwater Modeling*, Kluwer Academic Publishers, Dordrecht.

Tricaud C., Patan M., Uciński D. and Chen Y. (2008). D-optimal trajectory design of heterogeneous mobile sensors for parameter estimation of distributed systems, *Proceedings of the 2008 American Control Conference,* Seattle, WA, USA. Published on CD-ROM.

Uciński D. (1999). *Measurement Optimization for Parameter Estimation in Distributed Systems*, Technical University Press, Zielona Góra.

Uciński D. (2000). Optimal sensor location for parameter estimation of distributed processes, *International Journal of Control* **73**(13): 1235–1248.

Uciński D. (2005). *Optimal Measurement Methods for Distributed-Parameter System Identification*, CRC Press, Boca Raton, FL.

Uciński D. and Chen Y. (2005). Time-optimal path planning of moving sensors for parameter estimation of distributed systems, *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005,* Seville, Spain. Published on CD-ROM.

Uciński D. and Chen Y. (2006). Sensor motion planning in distributed parameter systems using Turing's measure of conditioning, *Proceedings of the 45th IEEE Conference on Decision and Control,* San Diego, CA, USA. Published on CD-ROM.

Uciński D. and Demetriou, M. A. (2008). Resource-constrained sensor routing for optimal observation of distributed parameter systems, *Proceedings of the 18th International Symposium on Mathematical Theory of Networks and Systems,* Blacksburg, VA, USA. Published on CD-ROM.

Uciński D. and Korbicz J. (2001). Optimal sensor allocation for parameter estimation in distributed systems, *Journal of Inverse and Ill-Posed Problems* **9**(3): 301–317.

Uciński D. and Patan M. (2007). D-optimal design of a monitoring network for parameter estimation of distributed systems, *Journal of Global Optimization* **39**(2): 291–322.

Walter É. and Pronzato L. (1997). *Identification of Parametric Models from Experimental Data*, Communications and Control Engineering, Springer-Verlag, Berlin.

Wyrzykowski R., Meyer N. and Stroiński M. (2004). Clusterix: National cluster of linux systems, *Proceedings of the 2nd European Across Grids 2004 Conference*, Nicosia, Cyprus. Available at: http://grid.ucy.ac.cy/axgrids04/AxGrids.

Zhao F. and Guibas L. J. (2004). *Wireless Sensor Networks: An Information Processing Approach*, Morgan Kaufmann Publishers, Amsterdam.

Received: 15 December 2007
Revised: 26 May 2008