Partially distributed optimization for mobile sensor path-planning

Lars Imsland[1]

***Abstract*— Mobile sensor platforms may provide low-cost, versatile means for obtaining high resolution information from spatially distributed processes. The aim of this paper is to efficiently calculate receding horizon-type motion plans that optimize information retrival by mobile sensors from distributed processes. Formulating the problem in a rather general framework as minimization of entropy, gives a huge, non-convex, in general intractable optimization problem for path planning. Based on this formulation, using approximation and decomposition strategies, we propose a new, more computationally tractable framework. In the case of multiple mobile sensors, the framework allows a partially distributed setup where each mobile sensor optimize its path using an economic-type Model Predictive Controller, where the "economic" (in the sense of information) objective is constructed based on simulation of covariances for the distributed process, performed by a central entity.**

I. INTRODUCTION

Using mobile sensor platforms such as unmanned aerial vehicles for information retrieval has been studied widely over the last decades, and a plethora of applications and approaches exist. In this paper, we will be interested in using mobile sensors to monitor distributed processes that involve transport and possibly generation of substance, typically governed by advection-diffusion-reaction (ADR) mechanisms. Examples of applications of such technology may range from metocean purposes to transport of physical quantities at or under sea, or in the air (monitoring of transport of sea ice, oil spill at sea, gaseous plumes in air, etc.), e.g. [1]–[5].

We will assume that the process of interest is monitored using a state estimation scheme, perhaps updated using a mixture of stationary measurements and measurements from mobile sensors. From this state estimation scheme, typically of Kalman-filter type, we assume there is an accompanying state uncertainty measure in the form of a covariance matrix. We will use this measure of uncertainty distribution, together with a linear(-ized) prediction model for the process, to plan sensor paths that maximize future information retrieval in the form of minimizing a prediction of the future covariance matrix (entropy). As such, our starting point is similar to e.g. [1], [6]–[8]. In an information-theoretic setting, this can be interpreted as minimization of mutual information between sensor measurements and predicted states of interest [9], [10].

The problem formulation is related to approaches minimizing the Fischer Information Matrix (FIM) for parameter estimation in distributed systems, both for placement of stationary sensors and mobile sensors, see e.g. [11], [12] and references therein. The approach in [13] estimates diffusion in advection-diffusion field by a distributed online passive identifier approach, where sensor paths are decided by a concensus algorithm based on performance of the distributed filters.

On other approaches in similar problem settings, we mention [4], [5] that also uses a state-estimator framework based on advection-diffusion equations, to estimate concentration from a moving gaseous source. The motion of the single aerial sensor is given by a Lyapunov-based control law improving the performance of a Luenberger state estimator.

We also mention [3] that avoids the complexity of propagating the covariance matrix dynamics (the Riccati equations) in a receding horizon framework, by instead propagating an uncertainty measure using the advection-diffusion process directly. Motion planning is done by formulating a receding horizon optimization problem for possibly multiple sensors.

After the problem definition in Section II, we propose a reformulation and approximation in Section III, which, integrated with model predictive motion control in Section IV, allows the decomposition strategy in Section V. In Section VI we propose a partially distributed algorithm for multiple sensors, before we illustrate the algorithm in a simple ADR example using two sensors, in Section VII.

II. PROBLEM FORMULATION

We will use mobile sensors to obtain information about spatially distributed processes, in a state-estimation framework. The typical example is a process governed by conservation laws of advection-diffusion-reaction (ADR)-type,

$$\frac{\partial c}{\partial t} + \nabla^{\mathsf{T}} f = s, \tag{1}$$

where, generally, the flux vector f consists of advection and diffusion,

$$f = uc + d\nabla c.$$

Here, c is the conserved variable to be estimated, s is a source (reaction) term, u is the velocity field (assumed known) and d is the diffusion coefficient.

We assume that the process has a state estimator attached, with an uncertainty measure in the form of a covariance matrix for the state estimate. Typically this will be a form of Kalman filter. The present value of the covariance matrix will be an input to a receding horizon path planning optimization.

We assume that the process to be estimated is linear (or linearized for planning purposes). Furthermore, we assume the partial differential equations are discretized to a finite spatial dimension and described by a linear ODE,

$$\dot{\xi}(t) = F\xi(t) + G\nu(t) + w(t) \tag{2}$$

where $\xi \in \mathbb{R}^{n_\xi}$ is the discretized state variable (c in the case above), F and G are (large) matrices from spatial

*In part sponsored by the Research Council of Norway through the Centres of Excellence funding scheme grant number 223254 - AMOS.

[1]Lars Imsland is with the Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim, Norway. lars.imsland@itk.ntnu.no

discretization of (1), and $\nu(t)$ is a (known) vector of time-varying boundary conditions and sources. To conform with Kalman filter setups, we have added Gaussian white process noise, $w(t) \sim \mathcal{N}(0, Q)$.

The mobile sensor will navigate the field and take measurements

$$z(t) = H(x(t))\xi(t) + v(t) \tag{3}$$

where $H(x(t))$ is the time-varying measurement matrix induced by the placement of the sensor(s) at time t, $x(t)$, and $v(t)$ is Gaussian white measurement noise, $v(t) \sim \mathcal{N}(0, R)$.

Similarly to e.g. [1], [14], our goal will be to construct paths that maximize the information we get of our process (2), or alternatively, minimize the uncertainty, or entropy. In an information-theoretic framework [1], the uncertainty reduction by one random variable (e.g. the state in (2)) by another random variable (e.g. the measurement (3)) can be quantified by the mutual information between them. In our case, the random variables of interest will be the state at some future time T_2, $\xi(T_2)$, and the information gathered by measurements from now until time T_1, $\mathcal{Z}_{T_1} = \{z(t), \quad t \in [0, T_1)\}$. Herein we will consider the case where $T_1 = T_2 = T$, unlike [1] which considers $T_2 > T_1$.

Under the assumptions here (linear, Gaussian), the mutual information is given as

$$\mathcal{I}(\xi(T), \mathcal{Z}_T) = \frac{1}{2}\log\det P(T) - \frac{1}{2}\log\det \bar{P}(T) \tag{4}$$

where $P(T)$ and $\bar{P}(T)$ are endpoint solutions of the Riccati equations

$$\dot{P}(t) = FP(t) + P(t)F^{\mathsf{T}} + Q - P(t)H^{\mathsf{T}}(x(t))R^{-1}H(x(t))P(t), \tag{5a}$$

$$\dot{\bar{P}}(t) = F\bar{P}(t) + \bar{P}(t)F^{\mathsf{T}} + Q. \tag{5b}$$

We will approach this as an optimal control problem, that is, we want to optimize over sensor paths to maximize the mutual information. As the sensor paths will not affect the second term in (4), this term can be removed and we can somewhat informally state our optimal control problem as[1]

$$\begin{aligned} \min_{\text{sensor paths}} \quad & \frac{1}{2}\log\det P(T) \\ \text{subject to} \quad & \text{(5a)} \end{aligned} \tag{6}$$

where the sensor path x influences the objective through the measurement matrix $H(x(t))$. The problem should be augmented with dynamic sensor motion models and measurement models, which will be addressed later. We will employ a direct transcription (first-discretize-then-optimize) approach to solving this optimal control problem [15], that is, the ODEs involved will be discretized such that the resulting (large) optimization problem can be formulated in a mathematical programming framework.

[1]This is the "filter form" considered in [1]. If we consider information reward in the far future of the sensor path in near future, according to [1] using a smoother form instead is advantageous.

III. AN APPROXIMATION AND REFORMULATION

The problem (6) is nonlinear both in objective and constraints. Coupled with the huge dimension of the optimization variables (mainly the discretized $P(\cdot)$), this makes this formulation rather untractable. We will introduce an approximation to remove the nonlinearity in the objective, and we will use an equivalent formulation of the Riccati dynamics to remove some of the nonlinearities in the constraints. We start with the latter, which is based on a classical theorem, found e.g. in [16], [17]:

Theorem 1 *Consider the matrix differential equations*

$$\dot{P}(t) = F(t)P(t) + P(t)F^{\mathsf{T}}(t) + Q(t) - P(t)H^{\mathsf{T}}(t)R^{-1}(t)H(t)P(t), \quad P(t_0) = V_0U_0^{-1} \tag{7}$$

and

$$\begin{pmatrix}\dot{U}(t)\\ \dot{V}(t)\end{pmatrix} = \begin{pmatrix}-F^{\mathsf{T}}(t) & H^{\mathsf{T}}(t)R^{-1}(t)H(t)\\ Q(t) & F(t)\end{pmatrix}\begin{pmatrix}U(t)\\ V(t)\end{pmatrix}, \quad \begin{pmatrix}U(t_0)\\ V(t_0)\end{pmatrix} = \begin{pmatrix}U_0\\ V_0\end{pmatrix} \tag{8}$$

where $P(t)$, $F(t)$, $Q(t)$, $U(t)$, $V(t)$ are all in $\mathbb{R}^{n\times n}$ and $H(t)$ and $R(t)$ are real matrices of appropriate dimensions. If the solution of (7) *exists on a time interval, then the solution of* (8) *exists on the same interval with $U(t)$ nonsingular and*

$$P(t) = V(t)U^{-1}(t). \tag{9}$$

Conversely, if the solution of (8) *exists on some interval with $U(t)$ nonsingular, then the solution of* (7) *exist on the same interval, and is given by* (9)*.*

Replacing (5a) with (8) in (6), the dynamic constraint is no longer nonlinear in the covariance matrix, but the objective is still nonlinear,

$$\begin{aligned}\log\det P(T) &= \log\det V(T)U^{-1}(T)\\ &= \log\det V(T)\det U^{-1}(T)\\ &= \log\det V(T) + \log\det U^{-1}(T)\\ &= \operatorname{tr}\log V(T) + \operatorname{tr}\log U^{-1}(T)\\ &= \operatorname{tr}\log V(T) - \operatorname{tr}\log U(T)\end{aligned} \tag{10}$$

where we have used $\det AB = \det A\det B$ for square matrices, $\log ab = \log a + \log b$, $\log\det A = \operatorname{tr}\log A$ for invertible matrices [18], and $\log A^{-1} = -\log A$.

For a quadratic matrix Γ with $\lambda_i(\Gamma) \le 1$, we have that

$$\log\Gamma = -(I-\Gamma) - \frac{1}{2}(I-\Gamma)^2 - \frac{1}{3}(I-\Gamma)^3 - \dots. \tag{11}$$

We use the first term[2] in this series together with (10) for an approximation, to obtain

$$\log\det P(T) \approx \operatorname{tr} V(T) - \operatorname{tr} U(T) \tag{12}$$

which is linear in the elements of $V(T)$ and $U(T)$.

[2]We should divide by the largest eigenvalue to fulfill the requirement that the spectral radius should be less than 1, but since we use only the first, linear, term, this is redundant: $\log\det\alpha\Gamma = n\log\alpha + \log\det\Gamma$.

Remark 1 *Note that* (12) *in general is a very coarse approximation (unless $P(T)$ is close to the identity matrix), not even of first order. However, we can also view this approximation as an alternative measure on the size of the covariance matrix. The function* $\log\det P$ *is monotone increasing with respect to the positive semidefinite cone, as is* $\operatorname{tr} P$*, and minimizing* $\operatorname{tr} V - \operatorname{tr} U$ *gives a small V and large U, which gives a small* $\operatorname{tr} P = \operatorname{tr} VU^{-1}$*. Note also that the* tr *function (modulo a constant) is an upper bound of the* $\log\det$ *function,*

$$\log\det\Gamma \leq \operatorname{tr}(\Gamma - I),$$

where the upper bound is tight in the sense that it holds with equality for $\Gamma = I$.

Remark 2 *An alternative approximation of first order could be based on*

$$\delta\log\det P = \delta\log\det VU^{-1} = \operatorname{tr} V^{-1}\delta V - \operatorname{tr} U^{-1}\delta U$$

but as this would require inversion of the large matrices V and U, we choose to not pursue this.

Let us now discretize the dynamic Riccati equation, using some linear discretization scheme (explicit or implicit), giving V_k, U_k, $k = 1, \ldots, N$, such that V_N (U_N) corresponds to $V(T)$ ($U(T)$). Assume (for simplicity) that the mobile sensor dynamics is discretized on the same grid, into x_k, $k = 1, \ldots, N$. We proceed to stack the matrices U_k and V_k into a large matrix $W = (U_0^\mathsf{T}, V_0^\mathsf{T}, U_1^\mathsf{T}, V_1^\mathsf{T}, \ldots, U_N^\mathsf{T}, V_N^\mathsf{T})^\mathsf{T}$, and similarly stack x_k into $\mathbf{x}$ (see next section). The optimization problem (6) can then compactly be written

$$\min_{W,\mathbf{x}} \operatorname{tr} C^\mathsf{T} W \quad \text{s.t.} \quad \Phi(\mathbf{x})W = B \tag{13}$$

where Φ is determined by the discretization scheme, $C = (0, 0, \ldots, 0, -I, I)^\mathsf{T}$ and B is a function of the initial covariance. Note that $\Phi(\mathbf{x})W = B$ is nothing else than a compact way to write the discretized Riccati simulation, and that for fixed $\mathbf{x}$, this is linear in W. In the next section, we will augment this problem with mobile sensor dynamics to get implementable sensor paths.

IV. MOBILE SENSOR MOTION PLANNING

We will assume linear discrete-time mobile sensor dynamics, on the form

$$x_{k+1} = Ax_k + Bu_k, \tag{14}$$

where $x_k \in \mathbb{R}^n$ and $u_k = \mathbb{R}^m$. The motion is subject to linear constraints (e.g. feasible region for the sensor, actuation constraints, etc.) summarized as

$$Cx_k + Du_k \leq d. \tag{15}$$

We stack the path variables in $\mathbf{x} = (x_0, x_1, \ldots, x_N)^\mathsf{T}$ and all variables related to the sensor motion in $\mathbf{z} = (x_0, u_0, x_1, u_1, \ldots, u_{N-1}, x_N)^\mathsf{T}$, where we note that $\mathbf{x}$ is a subset of $\mathbf{z}$. A MPC-type control problem for one sensor (not yet taking uncertainty/entropy reduction into account) can then be stated as

$$\begin{aligned}\min_{\mathbf{z}} \; & \sum_{k=0}^{N-1} x_{k+1}^\mathsf{T} Q_{\text{MPC}} x_{k+1} + u_k^\mathsf{T} R_{\text{MPC}} u_k \\ \text{s.t.} \; & x_{k+1} = Ax_k + Bu_k, \quad k = 0, \ldots, N-1 \\ & Cx_{k+1} + Du_k \leq d, \quad k = 0, \ldots, N-1\end{aligned}$$

which can compactly be formulated as

$$\begin{aligned}\min_{\mathbf{z}} \; & \mathbf{z}^\mathsf{T} S \mathbf{z} \\ \text{s.t.} \; & A_{\text{eq}}\mathbf{z} = b_{\text{eq}} \\ & A_{\text{ineq}}\mathbf{z} \leq b_{\text{ineq}}\end{aligned}$$

We assume Q_{MPC} positive semi-definite (typically the zero-matrix) and R_{MPC} positive definite, which makes S block-diagonal positive semi-definite.

The overall motion planning problem (for one sensor) is then formulated as joint minimization of uncertainty and control performance, subject to sensor dynamics:

$$\min_{W,\mathbf{z}} \; \operatorname{tr} C^\mathsf{T} W + \mathbf{z}^\mathsf{T} S \mathbf{z} \tag{16a}$$

$$\text{s.t.} \; \Phi(\mathbf{x})W = B \tag{16b}$$

$$A_{\text{eq}}\mathbf{z} = b_{\text{eq}} \tag{16c}$$

$$A_{\text{ineq}}\mathbf{z} \leq b_{\text{ineq}} \tag{16d}$$

The magnitude of S trades off uncertainty minimization vs control performance. This problem is nonlinear and nonconvex (due to (16b)), and huge (due to the dimension of W), but highly structured. We will exploit this structure to construct efficient solvers for the problem using decomposition.

V. A PRIMAL DECOMPOSITION STRATEGY

If we could fix the variable $\mathbf{x}$, the optimization problem (16) (both objective and constraints) would become separable, that is, it could be solved by solving two different problems. Moreover, it separates into convex problems that in principle are tractable to solve, even for large dimensions. For such problems, a common decomposition strategy (see e.g. [19] for an introduction) amounts to iteratively fixing the "complicating variable" ($\mathbf{x}$ in this case) and solving the two slave problems for W and the remaining parts of $\mathbf{z}$, and thereafter use gradient information from the slave problems to update the complicating variable in a master problem. However, straightforward decomposition is not a good avenue for our problem, due to the somewhat large dimension of the "complicating variable" and since the MPC-type sub-problem is likely to become infeasible during iterates. Therefore, we choose to merge the MPC slave problem with the master problem using a gradient update algorithm for $\mathbf{z}$.

The solution to (13) for fixed $\mathbf{x}$ is the solution of a huge linear system, since $\Phi(\mathbf{x})$ is quadratic and invertible (if a suitable integration scheme for the Riccati dynamics is used). We can in theory write this solution explicitly as a function of $\mathbf{x}$ as

$$\pi(\mathbf{x}) = \operatorname{tr} C^\mathsf{T} \Phi^{-1}(\mathbf{x}) B, \tag{17}$$

noting that in practice we do not want to invert the huge matrix $\Phi(\mathbf{x})$ to obtain an explicit form of this function.

If we had this function explicitly available, we could write (16) as

$$\begin{aligned}\min_{\mathbf{z}} \; & \pi(\mathbf{x}) + \mathbf{z}^\mathsf{T} S \mathbf{z} \\ \text{s.t.} \; & A_{\text{eq}}\mathbf{z} = b_{\text{eq}} \\ & A_{\text{ineq}}\mathbf{z} \leq b_{\text{ineq}}\end{aligned} \tag{18}$$

which is a much smaller optimization problem. Our decomposition strategy will be to iteratively find the gradient of $\pi(\mathbf{x})$ and use this to solve a linearization of (18). The procedure for obtaining the gradient of the $\pi(\mathbf{x})$-problem is given by Lemma 1. Since we are essentially finding the gradient of a scalar function with respect to many parameters, it is intuitive that the adjoint gradient method (see e.g. [20]) should be the most efficient method:

Lemma 1 *The gradient of $\pi(\mathbf{x})$ can be calculated as*

$$\frac{\mathrm{d}\pi(\mathbf{x})}{\mathrm{d}\mathbf{x}_i} = -\operatorname{tr}\Lambda^{\mathsf{T}}\frac{\mathrm{d}\Phi(\mathbf{x})}{\mathrm{d}\mathbf{x}_i}W \tag{19}$$

where Λ and W are found by solving

$$\Phi(\mathbf{x})W = B \quad \textit{and} \quad \Phi^{\mathsf{T}}(\mathbf{x})\Lambda = C. \tag{20}$$

Proof: Rewrite (17) as

$$\pi(\mathbf{x}) = \operatorname{tr} C^{\mathsf{T}}W \quad \text{where} \quad \Phi(\mathbf{x})W = B,$$

and by direct differentiation, we find that the ("forward") derivative with respect to element i of $\mathbf{x}$ is

$$\frac{\mathrm{d}\pi(\mathbf{x})}{\mathrm{d}\mathbf{x}_i} = \operatorname{tr} C^{\mathsf{T}}\Pi_i \text{ where } \Phi(\mathbf{x})\Pi_i = -\frac{\mathrm{d}\Phi(\mathbf{x})}{\mathrm{d}\mathbf{x}_i}W.$$

where $\Pi_i = \frac{\mathrm{d}W}{\mathrm{d}\mathbf{x}_i}$. Now, use the second equation to eliminate Π_i in the first, to obtain

$$\frac{\mathrm{d}\pi(\mathbf{x})}{\mathrm{d}\mathbf{x}_i} = -\operatorname{tr} C^{\mathsf{T}}\Phi^{-1}(\mathbf{x})\frac{\mathrm{d}\Phi(\mathbf{x})}{\mathrm{d}\mathbf{x}_i}W,$$

and use $\Phi^{\mathsf{T}}(\mathbf{x})\Lambda = C$ to replace $C^{\mathsf{T}}\Phi^{-1}(\mathbf{x})$ with Λ^{T}. ■

Note that by using the adjoint formulation we avoid having to calculate the Π_i matrices of the forward derivative, each the solution of a huge linear system (namely, a covariance simulation).

The interpretation of this result is that to calculate the gradient of the (scalar) objective, we have to perform two covariance simulations. One simulation forward in time, $\Phi(\mathbf{x})W = B$, and as inspection of $\Phi^{\mathsf{T}}(\mathbf{x})\Lambda = C$ reveals, one simulation backwards in time, initialized at $\Lambda_N = (-I, I)^{\mathsf{T}}$. Noteworthy, due to the linear structure of the Riccati dynamics and objective of (13), the two simulations are (for given $\mathbf{x}$) independent of each other, and can be performed in parallel.

Remark 3 *The problem structure implies that* (19) *can be further simplified. If $\mathbf{x}_i$ corresponds to an element in x_k, and we split $\Lambda = (\Lambda_0^{\mathsf{T}}, \Lambda_1^{\mathsf{T}}, \ldots, \Lambda_N^{\mathsf{T}})^{\mathsf{T}}$ corresponding to the backward (adjoint) simulation, then, if explicit Euler integration is used for simulating the Riccati dynamics,*

$$\frac{\mathrm{d}\pi(\mathbf{x})}{\mathrm{d}\mathbf{x}_i} = -\operatorname{tr}\Lambda_{k+1}^{\mathsf{T}}\begin{pmatrix}\Psi_i V_k \\ 0\end{pmatrix} \tag{21}$$

where $\Psi_i = h\frac{\mathrm{d}}{\mathrm{d}\mathbf{x}_i}H^{\mathsf{T}}(x_k)R^{-1}H(x_k)$, and h is the integration timestep used in the Euler algorithm. Other linear (explicit or implicit) integration schemes will give similar simplifications.

Having a procedure for finding the gradient of $\pi(\mathbf{x})$, we can outline an algorithm to solve (18) based on sequential linearization, see Algorithm 1. This is similar to the use of Sequential Quadratic Programming (SQP) for nonlinear programming. However, differently from SQP, we only use first-order information about $\pi(\mathbf{x})$, so local convergence properties will be inherited from first-order methods for unconstrained optimization (that is, linear). This could be improved by maintaining an approximation to the Hessian of $\pi(\mathbf{x})$ using some Quasi-Newton method [21], but this is not investigated further here.

while *Not converged* **do**

1. Solve (20): $\Phi(\mathbf{x})W = B$ and $\Phi^{\mathsf{T}}(\mathbf{x})\Lambda = C$.
2. Use (19) to find $f = \frac{\mathrm{d}\pi(\mathbf{x}^i)}{\mathrm{d}\mathbf{x}}$.
3. Solve QP
$$\begin{aligned}\min_{\Delta\mathbf{z}} \quad & f^{\mathsf{T}}(\mathbf{x}^i + \Delta\mathbf{x}) + (\mathbf{z}^i + \Delta\mathbf{z})^{\mathsf{T}}S(\mathbf{z}^i + \Delta\mathbf{z}) \\ \text{s.t.} \quad & A_{\text{eq}}(\mathbf{z}^i + \Delta\mathbf{z}) = b_{\text{eq}} \\ & A_{\text{ineq}}(\mathbf{z}^i + \Delta\mathbf{z}) \leq b_{\text{ineq}}\end{aligned}$$
4. Use a linesearch method to update
$$\mathbf{z}^{i+1} = \mathbf{z}^i + \alpha\Delta\mathbf{z}$$
5. $i = i + 1$

end

Algorithm 1: Algorithm for SQP-type solution

To ensure convergence far from the optimal solution, some type of line-search (as indicated in Algorithm 1) or trust-region method should be used to ensure sufficient decrease in each iteration. A trust-region constraint is easily added to (18), and can be monitored and updated as in traditional trust-region methods [21].

However, we will not use Algoritm 1 directly at each time-step. Instead we will, to limit computations, take one iteration per time-step. Moreover, we will extend the framework to be able to use several mobile sensors.

VI. A PARTIALLY DISTRIBUTED FRAMEWORK FOR MULTIPLE SENSORS

Extending the problem above to several sensors in a centralized setup is straightforward: Add more rows to the measurement matrix $H(x)$, typically one row for each sensor, and include dynamics of all mobile sensors in the MPC-type problem. Note that the Riccati dynamics does not become larger, meaning the main complexity of the gradient computation (Lemma 1) remains the same.

However, the framework lends itself easily to a partially distributed setup where each sensor can solve its own MPC problem, and the centralized computation consist of simulating the Riccati dynamics forward and backward in time, and use of Lemma 1 to calculate gradients for each sensor.

In doing this, we will, as mentioned above, take one SQP-type iteration per time-step. This can be interpreted as a simplified version of a "real-time iteration scheme", see e.g. [22]. We will also dispense with the use of linesearch or trust region-methods, basically assuming that the sensor motion constraints are restrictive enough to act as a type of trust-region constraint. If this is not the case, it is rather straightforward to augment the algorithm with a "real" trust region which can be adjusted online based on the objective function decrease in each time-step. Note that no additional covariance simulations are necessary for this, as the objective function is computed while computing the gradient of $\pi(\mathbf{x})$.

In the following, assume that $\mathbf{x}(t_k) = \mathbf{x}_k$ contains the (planned) sensor trajectories for all sensors at time t_k, and that the trajectory for sensor j is $\mathbf{x}_k^j$ (and similarly for $\mathbf{z}_k$). We also assume for simplicity that all the sensors have identical dynamics and constraints. Our algorithm for multiple sensors is then summarized in Algorithm 2. Step 2 is the central computation, mainly consisting of forward and backward covariance simulation, while the distributed optimizations consist of the MPC problems in step 3.

for *each timestep* t_k **do**

1. Obtain a covariance $P(t_k)$ for (2), for instance from a state estimator. Find U_0 and V_0 such that $P(t_k) = V_0 U_0^{-1}$.
2. Solve $\Phi(\mathbf{x}_{k-1})W = B$ and $\Phi^{\mathsf{T}}(\mathbf{x}_{k-1})\Lambda = C$.
3. **for** $j = 1$ **to** $n_{sensors}$ **do**

   3a. Use (19) to find $f^j = \frac{\mathrm{d}\pi(\mathbf{x})}{\mathrm{d}\mathbf{x}^j}$.

   3b. Measure sensor j position, x_k^j, and solve

$$\begin{aligned} \min_{\mathbf{z}^j} \quad & (f^j)^{\mathsf{T}}\mathbf{x}^j + (\mathbf{z}^j)^{\mathsf{T}} S \mathbf{z}^j \\ \text{s.t.} \quad & A_{\text{eq}}\mathbf{z}^j = b_{\text{eq}} \\ & A_{\text{ineq}}\mathbf{z}^j \leq b_{\text{ineq}} \end{aligned}$$

   3c. Implement first part of the trajectory for sensor j. Update $\mathbf{x}_k$.

   end

end

Algorithm 2: Partially distributed optimization for path-planning for n_{sensors} mobile sensors

If boundedness of trajectories and recursive feasibility of the distributed MPC-type algorithms is of concern (e.g. due to a tight admissible region for sensor movement), it is rather straightforward to add terminal constraints to the MPC problem (15) that guarantee that an infinite horizon solution without constraint violations can be constructed.

VII. SIMULATION EXAMPLE

As an illustration, consider an advection-diffusion system (1) in one spatial dimension, with drift velocity $u = 0.05$ and diffusion coefficient $d = 0.0001$. We discretize over the spatial domain $[0, 1]$ using the explicit finite volume method in [23] using $n_\xi = 100$ volumes, to obtain a system on the form (2).

We will use two mobile sensors, both with dynamics

$$\dot{x} = u, \quad -1 \leq u \leq 1,$$

discretized using explicit Euler with time-step $h = 0.05$. Each sensor is controlled by a distributed MPC controller as in Section VI with $Q = 0$ and $R = 0.1$. We use a prediction horizon of 1, discretized into 20 elements.

To obtain a continuous measurement function from the spatially discretized measurements, we use spatial interpolation based on sums of Gaussian functions,

$$y(x) = \alpha(x) \sum_{i=1}^{n_\xi} \exp\left(\frac{(x - x_i)^2}{s^2}\right) \xi_i =: H(x)\xi$$

where $\alpha(x)$ is a normalization factor, and s is a "variance" parameter deciding the size of the sensor's "field-of-view" (FOV). In the optimizations herein, we used $s = 0.2$. Similar measurement functions are used in e.g. [1], [3].

We implement a discrete-time Kalman filter to provide the sensor path-planning with initial covariance uncertainty. This Kalman filter is updated using measurements from the mobile sensors. The covariance matrix is initialized at time $t = 0$ with a sinusoid plus bias on the diagonal, and exponentially decaying elements off-diagonal. The Kalman filter covariance evolution without mobile sensor measurements is shown in Figure 1.

The Kalman filter covariance matrix when using the two mobile sensor measurements is illustrated in Figure 2, with the sensor trajectories resulting from path planning using Algorithm 2 superimposed.

The distributed MPC optimizations at each time-step (Step 3b. in Algorithm 2) are solved in ca. 10 ms on a recent laptop computer, using a standard QP solver. The main work per time step is the central forward and backward simulation of the Riccati dynamics (Step 2 in Algorithm 2), which each takes (for $n_\xi = 100$) about 100 ms.

As a measure of complexity savings, using a state-of-the-art nonlinear programming solver to solve the open loop problem (6) subject to one sensor[3], optimizations take about 4 minutes for $n_\xi \approx 20$, and this increase very quickly for increasing n_ξ. It should be noted that using such coarse spatial discretizations, in addition to giving discretization errors of the advection-diffusion problem, is problematic for being able to use measurement functions that give reasonable FOV-approximations.

VIII. CONCLUDING REMARKS

We have put forward a partially distributed approach to path planning for information retrieval for distributed systems. The main complexity is the central computations consisting of simulating Riccati dynamics forward and backward in time, while the distributed optimization that compute the actual sensor paths, consists of small MPC-type optimization problems. The information from the central entity to the mobile sensors are gradients computed using an adjoint-based method. In this respect, the approach has similarities with adjoint PDE-constrained optimization [26, e.g.].

The approach could easily be extended to account for nonlinear (nonholonomic) sensor dynamics, resulting in Nonlinear Model Predictive Controllers for each sensor. It is also easy to imagine variants of the approach, for instance doing several iterations at each time-step, or updating the Riccati simulations after each sensor has completed its optimization.

The approximation (12) for the objective gives reasonable solutions for the simple case studied herein, but the generality of this is an issue for further investigation. In this regard, it is relevant to note that the alternative approximation/bound $\log\det P \approx \operatorname{tr} P$, along with using the standard Riccati

[3]This was done using the AD-framework CasADi [24], and the solver IPOPT [25]. To implement the $\log\det$-function, we used Cholesky decompositions, which gives complex gradient computations for large n_ξ. Using the symbolic derivative (not presently implemented in CasADi for the $\log\det$-function), would give similar complexity issues, as it involves the inverse of the covariance matrix. It would likely be faster for large n_ξ to use finite difference approximations for the gradient, but this has its own issues.

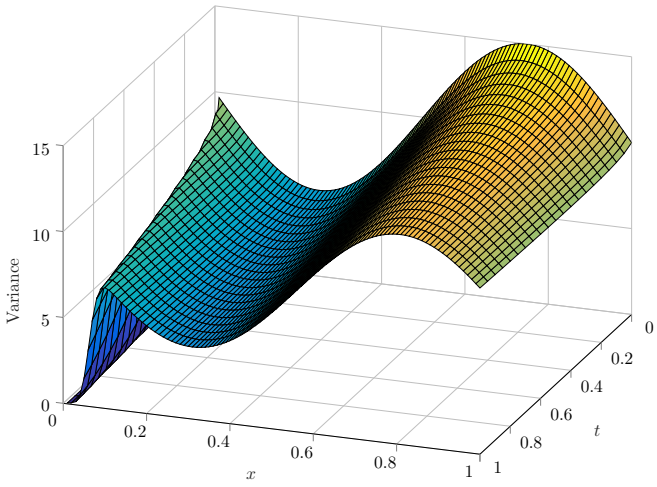

Fig. 1. Time evolution of the diagonal elements of the Kalman filter covariance matrix, without measurements. The sharp gradient on the left corresponds to advection of small uncertainty "boundary conditions" into the domain.

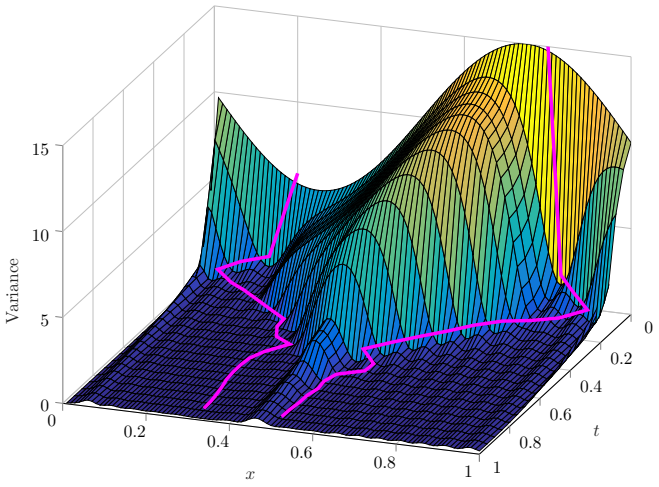

Fig. 2. Time evolution of the diagonal elements of the Kalman filter covariance matrix. The two mobile sensor trajectories are plotted just above the corresponding covariance matrix value.

equation (7) instead of (8), will benefit from using adjoint gradients in a similar way to that presented here. However, the resulting simulations forward and backward in time will be nonlinear and no longer independent in this case.

REFERENCES

[1] H.-L. Choi and J. P. How, "Continuous trajectory planning of mobile sensors for informative forecasting," *Automatica*, vol. 46, no. 8, pp. 1266–1275, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0005109810002104

[2] Y. Wang, R. Tan, G. Xing, J. Wang, and X. Tan, "Profiling aquatic diffusion process using robotic sensor networks," *IEEE Transactions on Mobile Computing*, vol. 13, no. 4, pp. 880–893, April 2014.

[3] J. Haugen and L. Imsland, "Monitoring an advection-diffusion process using aerial mobile sensors," *Unmanned Systems*, vol. 3, no. 3, pp. 221–238, 2015.

[4] T. Egorova, N. A. Gatsonis, and M. A. Demetriou, "Estimation of gaseous plume concentration with an unmanned aerial vehicle," *Journal of guidance, control, and dynamics*, vol. 39, no. 6, pp. 1314–1324, 2016.

[5] M. A. Demetriou, N. A. Gatsonis, and J. R. Court, "Coupled controls-computational fluids approach for the estimation of the concentration from a moving gaseous source in a 2-d domain with a lyapunov-guided sensing aerial vehicle," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 3, pp. 853–867, May 2014.

[6] I. Hussein, "Kalman filtering with optimal sensor motion planning," in *Proc. Amer. Contr. Conf.*, Seattle, WA, 2008, pp. 3548–3553.

[7] J. A. Burns, E. M. Cliff, and C. Rautenberg, "A distributed parameter control approach to optimal filtering and smoothing with mobile sensor networks," in *Proceedings of 17th Mediterranean Conference on Control and Automation*, Thessaloniki, Greece, 2009.

[8] J. A. Burns, E. M. Cliff, C. Rautenberg, and L. Zietsman, "Optimal sensor design for estimation and optimization of PDE systems," in *Proc. Amer. Contr. Conf.*, Baltimore, MD, USA, 2010.

[9] B. Grocholsky, A. Makarenko, and H. Durrant-Whyte, "Information-theoretic coordinated control of multiple sensor platforms," in *IEEE international conference on robotics and automation, Taipei, Taiwan*, 2003, pp. 1521–1526.

[10] G. M. Hoffmann and C. J. Tomlin, "Mobile sensor network control using mutual information methods and particle filters," *IEEE Transactions on Automatic Control*, vol. 55, no. 1, pp. 32–47, Jan 2010.

[11] D. Uciński and M. Patan, "Sensor network design for the estimation of spatially distributed processes," *Int. J. Appl. Math. Comput. Sci.*,, vol. 20, no. 3, pp. 459–481, 2010.

[12] C. Tricaud and Y. Chen, *Optimal Mobile Sensing and Actuation Policies in Cyber-physical Systems*. Springer London, 2012.

[13] J. You and W. Wu, "Online passive identifier for spatially distributed systems using mobile sensor networks," *IEEE Transactions on Control Systems Technology*, vol. PP, no. 99, pp. 1–9, 2017.

[14] B. Grocholsky, "Information-theoretic control of multiple sensor platforms," Ph.D. dissertation, University of Sydney, 2002.

[15] J. T. Betts, *Practical methods for optimal control using nonlinear programming*, ser. Advances in design and control. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2010.

[16] W. T. Reid, *Riccati differential equations*, ser. Mathematics in Science and Engineering. Burlington, MA: Elsevier, 1972.

[17] C. H. Choi and A. J. Laub, "Efficient matrix-valued algorithms for solving stiff riccati differential equations," *IEEE Transactions on Automatic Control*, vol. 35, no. 7, pp. 770–776, July 1990.

[18] C. S. Withers and S. Nadarajah, "log det a = tr log a," *International Journal of Mathematical Education in Science and Technology*, vol. 41, no. 8, pp. 1121–1124, 2010.

[19] S. Boyd, L. Xiao, A. Mutapcic, and J. Mattingley, "Notes on decomposition methods," 2008, notes for EE364B, Stanford University.

[20] Y. Cao, S. Li, L. Petzold, and R. Serban, "Adjoint sensitivity analysis for differential-algebraic equations: The adjoint DAE system and its numerical solution," *SIAM J. Sci. Comput*, vol. 24, no. 3, pp. 1076–1089, 2003.

[21] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York: Springer, 2006.

[22] M. Diehl, H. Bock, J. P. Schlder, R. Findeisen, Z. Nagy, and F. Allgwer, "Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations," *Journal of Process Control*, vol. 12, no. 4, pp. 577–585, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0959152401000233

[23] J. ten Thije Boonkkamp and M. Anthonissen, "The finite volume-complete flux scheme for advection-diffusion-reaction equations," *Journal of Scientific Computing*, vol. 46, no. 1, pp. 47–70, 2011.

[24] J. Andersson, "A General-Purpose Software Framework for Dynamic Optimization," PhD thesis, Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering (ESAT/SCD) and Optimization in Engineering Center, Kasteelpark Arenberg 10, 3001-Heverlee, Belgium, October 2013.

[25] A. Wächter and L. T. Biegler, "On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

[26] M. Gunzburger, *Perspectives in Flow Control and Optimization*, ser. Advances in design and control. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2003.