Specification and control synthesis for networked vehicle systems

João Borges de Sousa
and Fernando Lobo Pereira
Dept. Eng. Electrotécnica e de Computadores
Faculdade de Engenharia da Universidade do Porto
4200-465 Porto, Portugal
Email: {jtasso,flp}@fe.up.pt

Maria Bento Nunes
Laboratório de Aeronautica
Academia da Força Aérea Portuguesa
Granja do Marquês
Sintra, Portugal
E-mail: mfnunes@emfa.pt

***Abstract*— A framework for the representation, formal specification, and control synthesis for networked vehicle systems is presented. From dynamic optimization, this framework has inherited the concepts and theories of optimality, reach set computation and control, and the motivation to improve the performance of increasingly complex physical processes. From set theory, this framework borrowed the representational power of the language of sets to capture the relations among vehicles and controllers in a way that is consistent with control design. The ANTEX-M project is described to illustrate the challenges posed by networked vehicle systems and to illustrate how the framework addresses these challenges.**

I. Introduction

Over the last decade we have been designing and building multi-vehicle systems for underwater, sea, air, and ground applications [5], [4], [3]. In this process we developed a better understanding of the problem of bringing together technological development, theoretical underpinnings, and computational tools in the design and implementation of networked semi-autonomous and autonomous vehicles. This problem poses a new challenge to control engineering. The challenge comes from the distributed nature of the problem and from the nature of interactions. For example, in networked multi-vehicle systems, information and commands are exchanged among multiple vehicles, and the roles, relative positions, and dependencies of those vehicles change during operations. This challenge entails a shift in the focus of control theory – from prescribing and commanding the behavior of isolated systems to prescribing and commanding the behavior of interacting systems.

In this paper we outline a framework for the formal representation, specification, and control synthesis of networked vehicle systems.

For our purposes, the world consists basically of regions, physical objects, teams of physical objects, and networks of teams. While some objects have a physical existence, others are brought into existence as software agents. Examples of software agents are controllers that may be created, modified, and destructed in real-time. Examples of physical objects include vehicles and other devices. Vehicles have attributes and are capable of delivering atomic services (e.g. sensing), of executing tasks (e.g. fly a certain path), and of performing actions (e.g. launch a missile). A **complex service** is a service that cannot be delivered by a single physical object: it requires the composition of atomic services delivered by multiple physical objects. This is done with **atomic links**. An atomic link is a relation on the positions, motions and atomic services provided by two physical objects. An **atomic configuration** is a list of atomic links connecting a group of vehicles. Vehicles are teamed to deliver services, and to perform tasks that cannot be delivered by a single physical object.

In what concerns formal representation we represent all of the objects, their dynamic behavior, and the relations among themselves with simple concepts from set theory and from dynamic optimization. Relations of interest for us are: 1) services and their composition; 2) services and their physical implementation; 3) services and their order relations; 4) objects and modes of coordination; 5) objects and properties of their composition; 6) services and service providers; 7) objects and their control structure; 8) control structures and services. Some of the relations are static, others concern the dynamic behavior of vehicles under coordination constraints that change with time. In order to represent the last ones we use a set-valued description of the dynamic behavior of vehicles and teams. We use reach sets to describe the evolution of a dynamic system, invariant sets to describe the locations where the permanence of an object within a certain set is ensured, and solvability sets to describe the locations from which a system can evolve to reach a given set.

We specify operations on objects, and express the specification in a formal language. The key observation is that we can represent the objects, the relations they satisfy, and their operations in the language of sets. We will show that set-valued constraints express all of the relations of interest for systems of networked objects. This way we are able to represent the world of systems of networked objects with simple concepts from set theory. This is why we will be able to formally relate design and specification. In fact, this is the key idea behind our specification and control framework.

We write partial plan specifications and define a planning procedure that results in a data structure defining all of the controller specifications that precede controller design, and where all logical relations are already satisfied.

Finally, we use techniques from dynamic optimization to synthesize controllers that implement the plan, or that prove that the plan is not feasible.

The paper is organized as follows. In section II we discuss the ANTEX-M project to describe part of the motivation for our developments. In section III we introduce a mission example that we will use in the reminder of the paper to illustrate our framework. In section IV we discuss the issues of formal representation and specification. In section V we formulate the control problem for the mission example and in section VI we discuss the solution methodology. In section VII we draw some conclusions.

II. ANTEX-M

The *ANTEX-M* project concerns the design and the construction of a low cost unmanned air vehicle (UAV) platform for the Portuguese Air Force. The objectives of the project are:

- To design and build a low cost UAV platform for experimentation, development, and integration of sensing and communication technologies.
- To develop the technological and experimental expertise required to integrate UAV technology in the Portuguese Armed Forces.
- To demonstrate the operational capabilities of UAVs, either in isolated operation or integrated in a system.

The primary mission of the *ANTEX-M* UAV concerns surveillance. The specific applications are:

- Search and rescue operations in the Portuguese coastal waters.
- Monitor military activities in tactical operations.
- Anti-terrorist operations.
- Detection and tracking of maritime pollution.
- Fire detection.

The *ANTEX-M* UAV platform is an evolution of a Remotely Piloted Vehicle (RPV) developed by the Portuguese Air Force Academy to conduct research on adaptive aero-elastic structures [11]. The UAV will serve as a platform to mount sensors developed by the Portuguese Armed Forces and by the Directorate of Armament and Defense Equipment (Direcção Geral do Armamento e Equipamento e Equipamentos de Defesa). These sensors include infrared sensors with image processing, for detection and automatic tracking on board; laser emission detectors; and radar laser systems for three-dimensional image generation.

The preliminary design specifications for the UAV are: 1) empty weight – 5kg; 2) maximum take off weight – 8kg; 3) wing span – 2.4 m; 4) max level speed – 151 km/hr; 5) cruise speed – 139 km/hr.

III. Mission example

We illustrate our framework with one of the conceivable missions for the *ANTEX-M* type Unmanned Air Vehicle.

Fig. 1. ANTEX-M UAV

A conceivable mission for the *ANTEX-M* UAVs consists in the surveillance and mapping of selected regions. One such example consists in monitoring the evolution of oil spills[1].

Consider the following mission involving two vehicles, A and B, that coordinate their motions to execute a "mapping" task. The "mapping" task consists of having vehicle A following a prescribed path in the geographic (x,y) plane and taking measurements along that path without colliding with obstacles. There are no constraints on the z geographic coordinate except for those arising from unknown obstacles. Vehicle A has a mapping sensor and does not have any sensor for obstacle avoidance. Vehicle B surveys the area in front of vehicle A to identify the presence of potential obstacles. B is faster than A, and communicates the presence of obstacles to A. To do this B, carries an obstacle detection sensor. The problem is to coordinate the motions of the two vehicles so that, under mild assumptions on the topography of the world, the vehicles are able to execute the mapping task successfully, i.e. vehicle A does not collide with an obstacle before reaching its destination.

Hereafter, and unless stated otherwise, we will refer to this mission as our "example".

IV. Formal representation and specification

The world consists basically of **regions**, **physical objects**, **teams** of physical objects, and **networks** of teams. While some objects have a physical existence, others are brought into existence as software agents.

Regions are subsets of $\Re^n$. Physical objects are **vehicles**, and **devices**. Each vehicle has a **Type** and each physical object is located within at least one region.

A vehicle/device has **attributes** (e.g. range), it is capable of delivering **atomic services** (e.g. sensing), of executing **tasks** (e.g. fly a certain path), and of performing **actions** (e.g.

[1]This type of mission is particularly important for Portugal. The intense maritime traffic to and from Europe presents a considerable environmental threat, as demonstrated recently by the oil spill from the *Prestige* tanker.

launch a missile). A vehicle is controlled to move, and to deliver atomic services while moving. Physical objects have the potential to establish interactions among themselves. This is done with **atomic links**. An atomic link is a relation on the positions, motions and atomic services provided by two physical objects. An **atomic configuration** is a list of atomic links connecting a group of vehicles.

We use physical objects as the building blocks of **teams** and of **networks** of teams. Teams and networks of teams are brought into existence to deliver complex services, and to perform tasks that cannot be delivered by a single physical object.

A **complex service** is a service that cannot be delivered by a single physical object: it requires the composition of atomic services delivered by multiple physical objects, in particular vehicles. In order to do this, these vehicles have to be in a particular atomic configuration. In practice, complex services emerge from **modes of cooperation** among multiple objects, for example physical objects and software agents.

A **team** is a set of vehicles that is able to perform **team missions**. A team mission consists of team tasks and of task switching logic (also called a team **play**). A team task consists of the delivery of services and motions.

A **plan** is a data structure consisting of team tasks, controller specifications for each task, ordering constraints, variable binding constraints, and causal links. The plan is refined into team tasks. The refinement process involves team composition and tasking, resource allocation, and path planning.

Next we illustrate these concepts with the representation of the problem domain for our mission example.

In our example the set *Vehicles* is:

$$\textit{Vehicles}=\{A,B,C,D\}$$

There are two types of vehicles *Mapper* and *Scout*:

$$\textit{Type(A)=Mapper, Type(B)=Scout, Type(C)=Scout, Type(D)=Mapper}$$

The function *ProvideAtomicService* returns the list of atomic services provided by each vehicle type. The function *AttributeAtomicService* returns the list of attributes of an atomic service and the function *ValueAttribute(a,c)* returns the value of attribute *a* of the type *c* atomic service.

$$\begin{aligned}\textit{ProvideAtomicService(Mapper)}&=\{\textit{Coms,MapSensor,Motion}\}\\ \textit{ProvideAtomicService(Scout)}&=\\ \{\textit{Coms, ObstacleDetectionSensor,Motion}\}&\\ \textit{AttributeAtomicService(Coms)}&=\textit{Range}\\ \textit{ValueAttribute(Range,Coms)}&=R_{Coms}\end{aligned}$$

The equations of motion for all vehicles are given by:

$$\dot{x}_i(t) = f_i(t, x_i(t), u_i(t)) \quad u_i(t) \in \mathcal{U}_i, i = A, B, C, D$$

The function *Position(t,Z)* returns the geographic position (x,y,z) of vehicle *Z* at time t *Position(t,Z)* = $\Pi(x_Z(t))$. Π gives the projection of the state of vehicle Z onto the geographical position of the vehicle.

Atomic services are the building blocks of complex services. This is because some of the atomic services have the potential to establish interactions among the respective service providers. We call the atoms of these interactions **atomic links**: an atomic link is a relation on the relative motions, positions and atomic services provided by two different service providers. The predicate $AtomicLink(l, v_1, v_2)$ represents the fact that vehicles v_1 and v_2 are linked with a link of type *l*. The type defines the **role** – the atomic services and the list of commands accepted and issued – of each of the participants in the link and the **glue** – the way the two participants interact. The glue is a relation on the relative positions and motions of both service providers, and on the commands they exchange. The glue is determined from the attributes of the corresponding atomic services.

We represent the fact that any two vehicles in *Vehicles* are able to communicate under well-defined conditions with the atomic link of type *Coms*:

$$\begin{aligned}\textit{AtomicLink(Coms,}v_1, v_2\textit{)} \Leftrightarrow\\ \textit{Coms} \in \textit{ProvideAtomicService(}v_1\textit{)} \wedge\\ \textit{Coms} \in \textit{ProvideAtomicService(}v_2\textit{)} \wedge\\ \overline{\phi_{\textit{Coms}}}(\textit{Position(t,}v_1\textit{),Position(t,}v_2\textit{))} \leq 1\end{aligned}$$

where

$$\overline{\phi_{Coms}}(a, b) : \Re^3 \times \Re^3 \rightarrow \Re, \text{ s.t.}$$
$$\overline{\phi_{Coms}}(a, b) = \frac{d^2(a, b)}{R^2_{Coms}}, d(a, b) = \|a - b\|_2$$

It is convenient to express the function $\overline{\phi_{Coms}}$ in terms of the full state of both vehicles v_1 and v_2.

$$\phi_{Coms}(x_{v_1}, x_{v_2}) = \overline{\phi_{Coms}}(\Pi(x_{v_1}), \Pi(x_{v_2}))$$

We use the following predicates and functions to represent the complex service of type *s*:

- *RequiredVehicleType(s)* returns a list with the types of vehicles required to implement the service.
- *RequiredVehicles(s,c,V)* returns all the subsets of *V* capable of delivering the complex service of type *s* with the value of attributes specified in *c*.
- *RequiredConfigurationStyle(c,a)* returns the configuration style that each set of vehicles in *RequiredVehicles(c,a)* must satisfy to deliver the service *c* with the value of attributes as specified *a*.

For example, we represent the interactions between vehicles A and B in our mission example as the *ScoutedMapping* complex service. To do this we consider two generic *Mapper* and *Scout* vehicles, v_1 and v_2 respectively.

The vehicle of type *Scout*, v_2, evolves in a vicinity *P* of the current geographic position of v_1 and informs v_1 of the existence of obstacles so that v_1 can perform obstacle avoidance successfully. $P(Position(t, v_1))$ is given as a set-valued map from the current geographic position of v_1 to a subset of $\Re^3$.

$$P(a) : a \in \Re^3 \hookrightarrow P(a) \subset \Re^3$$

We represent this type of interactions between v_1 and v_2 with the atomic link of type *Inside*.

$$\begin{gathered}\textit{AtomicLink(Inside,}v_1, v_2\textit{)} \Leftrightarrow \\ (\textit{Type(}v_1\textit{)=Mapper}) \wedge (\textit{Type(}v_2\textit{)=Scout}) \wedge \\ (\overline{\phi_{Inside}}(\textit{Position(t, }v_1\textit{),Position(t, }v_2\textit{)}) \leq 1)\end{gathered}$$

where

$$\begin{gathered}\overline{\phi_{Inside}} : \Re^3 \times \Re^3 \rightarrow \Re \text{ s.t.} \\ \overline{\phi_{Inside}}(a,b) = d_c^2(b, P(a)) + 1, d_c(b,P) = \min_{s \in P} d(s,b)\end{gathered}$$

As before we define $\phi_{Inside}(x_{v_1}, x_{v_2})$ as follows:

$$\phi_{Inside}(x_{v_1}, x_{v_2}) = \overline{\phi_{Inside}}(\Pi(x_{v_1}), \Pi(x_{v_2}))$$

The implementation of the service *ScoutedMapping* also requires both vehicles to communicate. This means that they have to satisfy a configuration, i.e. a list of atomic links. We use an **atomic configuration style** as a compact representation of a set of atomic configurations sharing a common property. We represent the configuration style *y* with a predicate *ConfigurationStyle(y,c)*, where *c* is a team of vehicles.

$$\begin{gathered}\textit{ConfigurationStyle(ScoutMapper,V)} \Leftrightarrow \\ \exists X, Y \in V : \textit{Type(X)=Mapper} \wedge \textit{Type(Y)=Scout} \wedge \\ \textit{AtomicLink(Coms,X,Y)} \wedge \textit{AtomicLink(Inside,X,Y)}\end{gathered}$$

Finally we are able to represent the *ScoutedMapping* complex service:

$$\begin{gathered}\textit{RequiredVehicleType(ScoutedMapping)=}\{\textit{Scout,Mapper}\} \\ \textit{RequiredVehicles(ScoutedMapping,nil,Vehicles)=} \\ \{\{A,B\},\{A,C\},\{D,B\},\{D,C\}\} \\ \textit{RequiredConfigurationStyle(ScoutedMapping,Vehicles)=} \\ \textit{ScoutMapper}\end{gathered}$$

Single vehicles and teams of vehicles execute tasks. A task has a type. Consider, for example, the *Mapping* task. This task is defined as follows.

$$\begin{gathered}\textit{Task(Mapping, }\{X, Y\}\textit{, ScoutedMapping(X,Y),} \\ \textit{Path(}x_0, x_f, p, X\textit{), }\phi_0\textit{(Position(}t_0\textit{,X),Position(}t_0\textit{,Y)))}\end{gathered}$$

where *Mapping* is the type of the task, $\{X, Y\}$ is the team of vehicles executing the task while delivering the service *ScoutedMapping*, $p = \{(x,y) \in \Re^2 : (x,y) = p(t), t \in [t_0, t_f]\}$, and $Path(x_0, x_f, p, X)$ and ϕ_0*(Position(*t_0*,X),Position(*t_0*,Y))* are defined as follows (X, Y are vehicle variables):

$$\forall t \in [t_0, t_F] : \overline{\phi_{path}}\textit{(t,Position(t,X),p(t))} \leq 1$$

where

$$\begin{gathered}\overline{\phi_{path}}(t,a,b) : \Re \times \Re^3 \times \Re^3 \text{ s.t.} \\ \overline{\phi_{path}}(t,a,b) = d^2(a,b) - \delta + 1 \\ \phi_0(\textit{Position(}t_0, X\textit{),Position(}t_0, Y\textit{))} \leq 1\end{gathered}$$

δ is the path-tracking tolerance and the last equation defines the set of initial positions for vehicles X and Y. We define ϕ_{path} in the manner described above:

$$\phi_{path}(t, x_{v_i}, b) = \overline{\phi_{path}}(t, \Pi(x_{v_i}), b)$$

The plan specification is a data structure consisting of tasks and a partial order on these tasks. In our example the plan specification consists only of the mapping task:

$$\begin{gathered}\textit{Plan = }\{\textit{Task(Mapping, }\{X,Y\}\textit{, ScoutedMapping(X,Y),} \\ \textit{Path((0,10),(100,10),p, X), }\phi_0\textit{(Position(}t_0\textit{,X),Position(}t_0\textit{,Y)))}\}\end{gathered}$$

We need to transform this plan specification onto an implementable plan, i.e. we need a planner. Here we are not concerned with planning procedures and we assume that the planner produced the following plan.

$$\begin{gathered}\textit{Plan = }\{\textit{Task(Mapping, }\{A,B\}\textit{, ScoutedMapping(A,B),} \\ \textit{Path((0,10),(100,10),p,X)), }\phi_0\textit{(Position(}t_0\textit{,A),Position(}t_0\textit{,B))}\}\end{gathered}$$

At this point the plan consists of control specifications from which we derive a feasible structure of controllers in case it exists.

V. Formulation

The control problem formulation arises naturally from the previous specification and is expressed as follows.

$$\begin{aligned}\forall t \in [0,1] : \phi_{path}(t, x_A(t), p(t)) \leq 1 \wedge \\ \phi_{Inside}(x_A(t), x_B(t)) \leq 1 \ \wedge \\ \phi_{Coms}(x_A(t), x_B(t)) \leq 1 \wedge \\ \phi_0(x_A(t_0), x_B(t_0)) \leq 1\end{aligned} \tag{1}$$

We obtain this formulation from the instantiated plan, where the variables X and Y are bound to vehicles A and B.

Remark 1: We represent all of the state constraints as inequalities of the form $\phi(x) \leq 1$. We use this representation to simplify the notation. In fact, all state constraints can be represented in this form.

In what follows we consider the following hypotheses:

H1. The set-valued map P is closed, convex, and bounded.
H2. The path p is continuous in t.
H3. $\phi_0(x,y)$ is continuous in both variables.

Lemma 1: Under hypotheses $H1 - 2$ the functions $\phi_{path}(t,x,y)$, $\phi_{Inside}(x,y)$, and $\phi_{Coms}(x,y)$ are continuous.

Define $\phi(t,x,y)$, u and $\mathcal{U}$ as follows:

$$\phi(t,x,y) = \max\{\phi_{path}(t,x,p(t)), \phi_{Inside}(x,y), \phi_{Coms}(x,y)\}$$
$$u = \{u_A, u_B\}, \mathcal{U} = \mathcal{U}_\mathcal{A} \times \mathcal{U}_\mathcal{B}$$
$$f(t,x_A,x_B,u) = col(f_A(t,x_A,u_A), f_B(t,x_B,u_B))$$

We use the approach from [10] to formulate this control problem as an invariance problem (see [8], [9], [2], [1]). To do this, we introduce the following value function.

$$V(t,x_A,x_B) = \min_{u(.)} \max\{\{\max_{\tau\in[t_0,t]}\{\phi(\tau,x_A[\tau],x_B[\tau])\}, \quad (2)$$
$$\phi_0(x_A(t_0),x_B(t_0))\}, x_A[t] = x_A, x_B[t] = x_B\}$$

where u(.) is a feasible control function ($u(\tau) \in \mathcal{U}, \tau \in [t_0,t]$).

Now consider the sub-level set of this value function given by the following equation:

$$R(t,x_A,x_B) = \{(x_A,x_B) : V(t,x_A,x_B) \leq 1\} \quad (3)$$

At time t, R represents the set of all locations for A and B that satisfy equation 1.

The question now is how to calculate the value function. This is not a trivial matter. The idea is to transform this global problem into a local one. We do this by transforming the global problem onto a partial differential equation.

VI. Solution

In general the value function V can be calculated through the generalized Hamilton-Jacobi-Bellman (HJB) equation. We can only do this if the value function satisfies the principle of optimality.

Theorem 1: The value function V satisfies the principle of optimality.

Basically the principle of optimality states that the value function satisfies a semi-group property. The value function inherits this property from the semi-group property of the reach set.

Using the techniques from [10] we can derive the HJB equation for this problem. First we introduce some notation:

$$\mathcal{H}(t,x,y,V,u) = V_t(t,x,y) + \langle V_x(t,x,y) \cdot f(t,x,y,u)\rangle \quad (4)$$

An infinitesimal version of the principle of optimality leads to Hamilton-Jacobi-Bellman equation:

$$V_t(t,x,y) + \max_{u\in\mathcal{U}}\langle V_x(t,x,y)\cdot f(t,x,y,u)\rangle = 0 \quad (5)$$
$$\text{when } V(t,x,y) \neq \phi(t,x,y)$$
$$\max_{u\in\mathcal{U}}\{\min\{\mathcal{H}(t,x,y,V,u), \mathcal{H}(t,x,y,\phi,u)\}\}$$
$$\text{when } V(t,x,y) \neq \phi(t,x,y)$$
$$V(t_0,x,y) = \max\{\phi(t_0,x,y), \phi_0(t_0,x,y)\}$$

where V_t, V_x represent the corresponding sub-differentials. Since V is non-differentiable the usual notion of solution of a partial differential equation does not apply. We consider generalized "viscosity", or equivalent concepts, of solutions for this equation (see [6], [7]).

Given a solution V to the Hamilton-Jacobi-Bellman equation we are able to find the invariant set R from equation 3. Now we have all of the ingredients required to synthesize the controller for our problem (see [7]). Define $U(t,x_A,x_B)$ as the set of control values where the maximum of equation 5 is attained when x_A and x_B are the values of the state of vehicles A and B at time t. In the interior of R we can use any feasible control. On the boundary of R the control selection is restricted to the set-valued map $U(t,x_A,x_B)$.

VII. Conclusions

In this paper we propose a specification, planning, and control synthesis framework for networked vehicles systems. We use the language of set theory to uniformly represent vehicles, patterns of interactions among these vehicles, and the behavior of ordinary differential equations – such as the ones describing the motions of a vehicle – and techniques from dynamic optimization for the set-valued representation of this behavior and for control synthesis under set-valued constraints.

The calculation of the value function is not a trivial matter. We are investigating computational methods to do this.

Acknowledgment

The authors thank Professors Pravin Varaiya and Alexander Kurzhanski for stimulating discussions on dynamic optimization and reach set computation, Dr. Raja Sengupta for the motivation for the UAV example, and Tenente Coronel António Costa and Capitão Delfim Dores from the Portuguese Air Force for the discussions on the ANTEX-M project.

João Borges de Sousa and Fernando Pereira have been supported by Fundação da Ciência e Tecnologia under project Cordyal.

References

[1] Jean-Pierre Aubin. *Viability theory*. Birkhauser, 1991.

[2] Jean-Pierre Aubin and Helene Frankowska. *Set-valued analysis*. Birkhauser, 1990.

[3] J. Borges de Sousa, Aníbal C. Matos, and F. Lobo Pereira. Dynamic optimization in the coordination and control of autonomous underwater vehicles. In *Proceedings of Decision and Control Conference*. IEEE, 2002.

[4] J. Borges de Sousa and F. Lobo Pereira. Specification and design of coordinated motions for autonomous vehicles. In *Proceedings of Decision and Control Conference*. IEEE, 2002.

[5] J. Borges de Sousa and Raja Sengupta. Tutorial on autonomous and semi-autonomous networked multi-vehicle systems, decision and control conference. December 2001.

[6] L. C. Evans. *Partial Differential Equations*. Graduate Studies in Mathematics. American Mathematical Society, 1998.

[7] N.N. Krasovskii and A.I. Subbotin. *Game-theoretical control problems*. Springer-Verlag, 1988.

[8] A. B. Kurzhanskii. *Advances in nonlinear dynamics and control : a report from Russia*. Birkhauser, 1993.

[9] A. B. Kurzhanskii. *Ellipsoidal calculus for estimation and control*. Birkhauser, 1997.

[10] A. B. Kurzhanskii and P. Varaiya. Optimization methods for target problems of control. In *Proceedings of Mathematical Theory of Networks and Systems Conference*, 2002.

[11] A. Suleman, P. A. Moniz, and A. P. Costa. Design and testing of adaptive rpv aerolastic demonstrator. In *Proceedings of the AIAA*, pages 1361–71, 2001.