Appropriate Design Guided by Simulation: An Hovercraft Application

Julien Alexandre Dit Sandretto, Douglas Piccani de Souza, Alexandre Chapoutot

Appropriate Design Guided by Simulation: An Hovercraft Application

Julien Alexandre dit Sandretto,
Douglas Picciani de Suza, and Alexandre Chapoutot

U2IS, ENSTA ParisTech, Université Paris-Saclay
828 bd des Maréchaux 91762 Palaiseau cedex France
{alexandre,piccani,chapoutot}@ensta.fr

March 2016

Abstract

A design methodology based on simulation of dynamical behavior is presented in this paper. The particularity of our method is that it exploits the set-membership simulation. Indeed, this method allows one to consider an interval of values for each parameter of the dynamical model. Finding the parameters validating the requirements is then a filter, based on a Branch and Prune algorithm. We prefer an approach of appropriate design that is a design which validates the physical constraints coming from requirements, to an optimal design which does not lead to satisfy imperative requirements. Our method is described and applied on the complex problem of design of an hovercraft under dynamic requirements.

1 Introduction

During the design process of a complex system such as a vehicle (aircraft, ship, etc.), a chemistry device or a production chain, it is usual to define constraints on the behavior of the futur system (see [8] for a list of design applications), *i.e.*, on its temporal evolution or its dynamics. These constraints come from the requirements defined during the engineering step. The most common approach to model the behavior of a complex system uses differential equations such as Ordinary Differential Equations (ODEs). On contrary, there is no standard method to validate the requirements over the dynamics of the system.

One well known method, consisting in Monte-Carlo simulations for estimating the behavior of a system, is often used to solve design problems. This method cannot try all the possible configurations with continuous valued parameters and quickly leads to a combinatorial problem. Moreover, it is not easy to manipulate uncertainties with this method, which are important in design of complex systems [7].

In a Multidisciplinary Design Optimization (MDO) process, one may consider the influence of the physical process variability, information uncertainty and the choice of modeling on the design. If it is different than a behavior constrained approach, it focuses on the uncertainties [2]. A common solution to the issue of uncertainties is to use a probabilistic approach [15]. Unfortunately, this approach does not provide guarantee on the requirement validation.

Finally, two main approaches can be used to solve a design problem

- *Optimal design*, which tries to find the best design w.r.t. a given cost, even if not feasible in the sense of requirements;
- *Appropriate design*, which tries to find one feasible design, but does not consider the cost of the solution.

We can notice that some methods mix both approaches such as [8]. If optimization approach is not directly linked to the possible values of design parameters [12], the appropriate design based on interval analysis searches design parameters into intervals [16].

Another similar method uses intervals and Branch and Bound algorithm to find a feasible and optimal solution [21]. Both of these interval-based methods do not consider the dynamics of the studied system. We propose in this paper an interval-based method using set-membership simulation and constraint

satisfaction problem (CSP) formalism. Our method is tested on the design of an hovercraft, which is an application very close to [19].

After some preliminary *definitions*, we present our *design methodology* in three sections. We describe two tools in sections 2 and 3, before presenting the complete algorithm in Section 4. Then we apply our method on the design of an hovercraft and present the results in Section 5. Section 6 concludes.

Notation and Definitions

The main underlying tool, used in our design methodology, is interval analysis [17]. In the following, we will often use the notation $[x] \in \mathbb{IR}$ (the set of intervals with real bounds) with $[x] = [\underline{x}, \overline{x}] = \{x \in \mathbb{R} \mid \underline{x} \leqslant x \leqslant \overline{x}\}$ denoting an interval. By abuse of notation, $[x]$ will also denote a vector of intervals, *i.e.*, a Cartesian product of intervals, a.k.a. a *box*. We now introduce two fundamental definitions.

Definition 1 *Consider an* Ordinary Differential Equation *(ODE) with a given initial condition*

$$\dot{x}(t) = f(t, x(t), p) \quad with \quad x(0) \in [x_0],\ p \in [p], \tag{1}$$

with $f : \mathbb{R}^+ \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ assumed to be continuous in t and p and globally Lipschitz in x. We assume that parameters p are bounded. An Initial Value Problem *(IVP) consists in finding a function $x(t)$ described by the ODE* (1) *for all p lying in $[p]$ and for all the initial conditions in $[x_0]$.*

Definition 2 *A numerical (or continuous)* Constraint Satisfaction Problem *(CSP) $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ is defined as follows:*

- $\mathcal{X} = \{x_1, \ldots, x_n\}$ *is a set of variables;*
- $\mathcal{D} = \{[x_1], \ldots, [x_n]\}$ *is a set of domains ($[x_i]$ contains all possible values of x_i);*
- $\mathcal{C} = \{c_1, \ldots, c_m\}$ *is a set of constraints.*

A valuation $(v_1, \ldots, v_n)$, $v_i \in [x_i]$ with $i = 1, \ldots, n$, of a CSP can be classified as *i) accepted* if it satisfies all the constraints; *ii) rejected* if it does not satisfies at least one constraint; or *iii) uncertain* if it partially satisfies, in a set-theoretic sense, at least one constraint and satisfies all the others constraints. Many methods have been developed to solve CSPs, such as [14] and [20].

In order to be able to consider ODEs in CSP formalism, two main approaches can be distinguished. First, [10] has shown that it is possible to consider ODE directly in CSP in an elegant manner by considering the solution $x(t)$ of (1) as a function involved into a constraint c. Second, [6] has defined a new extended formalism named Constraint Satisfaction Differential Problem (CSDP). In this formalism, the differential constraint is clearly identified. We place our approach in the first and classical formalism of CSP, and we do not give more details, the implementation being out of the scope of this paper.

2 Set-based Simulation

Several methods exist to solve (1) involving interval values such as [18, 1]. In this section, we describe our approach for validated simulation based on Runge-Kutta methods [5, 1].

A numerical integration method computes a sequence of approximations (t_n, x_n) of the solution $x(t; x_0)$ of the IVP defined in Equation (1) such that $x_n \approx x(t_n; x_{n-1})$. The simplest method is Euler's method in which $t_{n+1} = t_n + h$ for some step-size h and $x_{n+1} = x_n + h \times f(t_n, x_n, p)$; so the derivative of x at time t_n, $f(t_n, x_n, p)$, is used as an approximation of the derivative on the whole time interval to perform a linear interpolation. This method is very simple and fast, but requires small step-sizes. More advanced methods coming from the Runge-Kutta family use a few intermediate computations to improve the approximation of the derivative. The general form of an s-stage Runge-Kutta formula, that is using s evaluations of f, is

$$x_{n+1} = x_n + h \sum_{i=1}^{s} b_i k_i \ , \tag{2a}$$

$$k_i = f\Big(t_n + c_i h,\ x_n + h \sum_{j=1}^{s} a_{ij} k_j, p\Big) \ , \quad i = 1, 2, \ldots, s \ . \tag{2b}$$

The coefficients c_i, a_{ij} and b_i fully characterize the method. To make Runge-Kutta validated, the challenging question is how to compute a bound on the distance between the true solution and the numerical solution, defined by $x(t_n; x_{n-1}) - x_n$. This distance is related to the *local truncation error* (LTE) of the numerical method.

To bound the LTE, we rely on *order condition* [11] respected by all Runge-Kutta methods. This condition states that a method of this family is of order O iff the $O+1$ first coefficients of the Taylor expansion of the solution and the Taylor expansion of the numerical methods are equal. In consequence, LTE is proportional to the Lagrange remainders of Taylor expansions. Formally, LTE is defined by (see [5]):

$$x(t_n; x_{n-1}) - x_n = \frac{h^{O+1}}{(O+1)!}\left(f^{(O)}\left(\xi, x(\xi; x_{n-1}), p\right) - \frac{d^{O+1}\phi}{dt^{O+1}}(\eta)\right) \quad \xi \in]t_n, t_{n+1}[\text{ and } \eta \in]t_n, t_{n+1}[\ . \quad (3)$$

The function $f^{(n)}$ stands for the n-th derivative of function f w.r.t. time t that is $\frac{d^n f}{dt^n}$ and $h = t_{n+1} - t_n$ is the step-size. The function $\phi : \mathbb{R} \to \mathbb{R}^n$ is defined by $\phi(t) = x_n + h\sum_{i=1}^{s} b_i k_i(t)$ where $k_i(t)$ are defined as Equation (2b).

The challenge to make Runge-Kutta integration schemes safe w.r.t. the true solution of IVP is then to compute a bound of the result of Equation (3). In other words we have to bound the value of $f^{(O)}\left(\xi, x(\xi; x_{n-1}), p\right)$ and the value of $\frac{d^{O+1}\phi}{dt^{O+1}}(\eta)$. The latter expression is straightforward to bound because the function ϕ only depends on the value of the step-size h, and so does its $(O+1)$-th derivative. The bound is then obtain using interval analysis tools.

However, the expression $f^{(O)}\left(\xi, x(\xi; x_{n-1}), p\right)$ is not so easy to bound as it requires to evaluate f for a particular value of the IVP solution $x(\xi; x_{n-1})$ at an unknown time $\xi \in]t_n, t_{n+1}[$. The solution used is the same as the one found in [18] and it requires to bound the solution of IVP on the interval $[t_n, t_{n+1}]$. This bound is usually computed using the Banach's fixpoint theorem applied with the Picard-Lindelöf operator, see [18]. This operator is used to compute an enclosure of the solution $[\tilde{x}]$ of IVP over a time interval $[t_n, t_{n+1}]$, that is for all $t \in [t_n, t_{n+1}]$, $x(t; x_{n-1}) \in [\tilde{x}]$. We can hence bound $f^{(O)}$ substituting $x(\xi; x_{n-1})$ by $[\tilde{x}]$.

3 From requirements to CSP

During the process of design a system, it is common to define some requirements on its behavior, in particular for *functional requirements*. One requirement on the behavior of a system can be decomposed as one scenario to follow and one or more criteria to fulfill (see theory of System Engineering [4]). Our approach exploits the fact that a scenario can be simulated while each criterion can be translated into a constraint. Then, a design problem is equivalent to find the design parameters which validate the requirement by simulating the related scenario and by verifying a CSP.

Figure 1 shows an example of the use of constraints on a solution of a validated simulation. The square in black represents the hovercraft position, computed by validated simulation, at some instants. We set constraints which are represented by green boxes (for valid sets) and red boxes (for invalid sets). The constraints can then be of different types, for example: (OK-1) Hovercraft stays inside a valid set ; (OK-2) Hovercraft finishes in a valid set; (NOK-3) Hovercraft avoids an obstacle; and (NOK-4) Hovercraft is not in an invalid state at a given time. Constraints OK-1, NOK-3 and NOK-4 are associated to safety constraints while Constraint OK-2 is associated to a functional requirement.

4 Appropriate design algorithms

The modeling parameters can be split into two parts: the design parameters and the other ones, such that $p = \{p_{\text{design}}, p_{\text{model}}\}$. The model parameters p_{model} are fixed and can not be changed by the design. Nevertheless, they can be uncertain and then bounded in intervals (due to uncertainties on construction or modeling). The design parameters p_{design}, taken inside intervals of possible values (depending on construction constraints, part available on the market, etc.) will be filtered till they validate all the requirements. For that, we use the classical Branch and Prune algorithm which is efficient for CSP solving [13, 3].

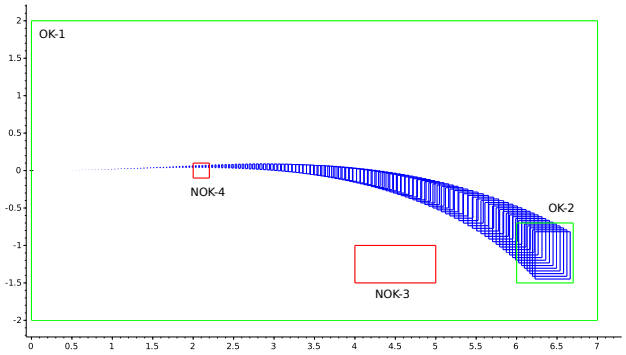


Figure 1: Example of solution by simulation with interval analysis and some allowed constraints in green and prohibited constraints in red.

4.1 Branch and Prune

A Branch and Prune algorithm consists on alternatively branching and pruning to produce two subpavings $\mathcal{L}$ and $\mathcal{S}$, with $\mathcal{L}$ the boxes too small to be bisected and $\mathcal{S}$ the solution boxes. We are then sure that all the solutions are included in $\mathcal{L} \cup \mathcal{S}$ and that every point in $\mathcal{S}$ is solution.

Literally, this algorithm browses a list of boxes and for each one *i)* Prune: the CSP is evaluated (or contracted) on the current box, if it is a solution it is added to $\mathcal{S}$, otherwise *ii)* Branch: if the box is large enough, it is bisected and the two boxes resulting are added to the main list, otherwise the box is added to $\mathcal{L}$.

4.2 Propagation

If a variable domain has been reduced, the reduction is propagated to all constraints on that variable, which allows one to narrow the other variables domains. This process is repeated till a fixed point is reached (or gain too small w.r.t. computation). In our case, it is important to note that if a parameter design p_{design} is reduced, all the simulation need to be computed again.

4.3 Design Algorithm

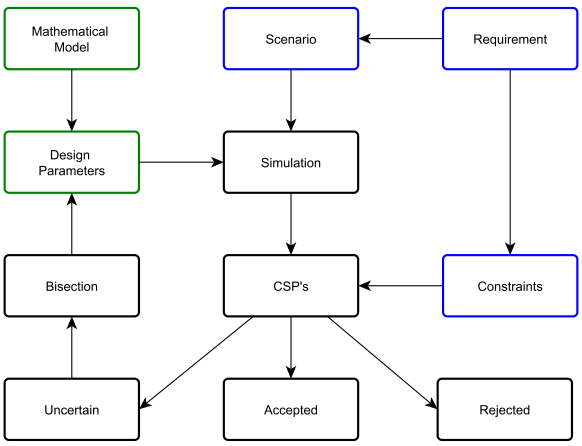


Figure 2: Schematic of the design methodology for one requirement.

A diagram of the proposed design methodology is presented in Figure 2 for only one requirement. A requirement comes with a scenario and some constraints to verify over this scenario. This part represents the engineering process of design. Another part, the mathematical one, provides a model of the considered

system. This model comes with some parameters that we want to compute in order to obtain a design which validates the requirement. This model is used in the simulation process to obtain the behavior of the system over the scenario. Then, CSPs are applied to this behavior to verify the constraints defined by the requirement. If the result of CSPs is positive then the set of design parameters is *accepted*, if the solution does not respect one of the CSPs then the set of design parameters is *rejected*. In other case, we classify it as *uncertain* and bisect the set of design parameters to perform a new analysis on each part. It is a classical Branch and Prune algorithm supplemented with set-membership simulation to deal with ODEs.

Algorithm 1 The design methodology.

Require: $Stack = \emptyset, Stack_{acc} = \emptyset, Stack_{rej} = \emptyset, Stack_{unc} = \emptyset, [par]_0 \subset \mathbb{R}^n$
  Push $[par]_0$ in $Stack$
  while $Stack \neq \emptyset$ **do**
    Pop a $[par]$ from $Stack$
    $\forall S_i \in \{S_1, ...S_m\}$ compute Sol_i solution using Simulation S_i with $[par]$ parameters
    if $\forall j \in \{1, ..., \tilde{m}_i\}, CSP\_i\_j\_accept(Sol_i)$ **then**
      Push $[par]$ in $Stack_{accepted}$
    else if $\exists j \in \{1, ..., \tilde{m}_i\} : CSP\_i\_j\_reject(Sol_i)$ **then**
      Push $[par]$ in $Stack_{rejected}$
    else if $width([par]) > lim$ **then**
      $([par_{left}], [par_{right}]) = Bisect([par])$
      Push $[par_{left}]$ in $Stack$
      Push $[par_{right}]$ in $Stack$
    else
      Push $[par]$ in $Stack_{uncertain}$
    end if
  end while

A general algorithm to handle more than one requirement is presented in Algorithm 1. In this algorithm, we use both CSP for acceptance and CSP for rejection to speed up the algorithm. Note that a CSP for rejection can be seen as the negation $\bar{b}$ of a Boolean formula b, *i.e.*, the values that satisfy b are not satisfy $\bar{b}$ and reciprocally. Hence, if we found values $(v_1, \ldots, v_n)$ satisfying a CSP rejection we know that $(v_1, \ldots, v_n)$ does not satisfy the CSP and so we can look for an other valuation.

5 An hovercraft design

We are currently developing many autonomous small vehicles. One of them is an hovercraft, which has many advantages but also a complex behavior. In order to build an efficient platform, we performed a design step with our method previously presented.

5.1 Problem Definition

The starting point to model this particular vehicle is the study of surface ships [9]. It helps to define a generic model of the dynamics of a rigid-body with 6 degrees of freedom. Applying some physical constraints, such as lying on a plane surface (altitude fixed to zero), and assuming that the body-fixed reference frame is placed at the center of gravity, the mathematical model is given by Equation (4) that is

$$\dot{u} = vr + \frac{1}{m}X \tag{4a}$$

$$\dot{v} = -ur + \frac{1}{m}Y \tag{4b}$$

$$\dot{r} = \frac{1}{I_z}N \tag{4c}$$

$$\dot{x} = \cos(\psi)u - \sin(\psi)v \tag{4d}$$

$$\dot{y} = \sin(\psi)u + \cos(\psi)v \tag{4e}$$

$$\dot{\psi} = r \tag{4f}$$

The function u stands for rectilinear acceleration in x-axis while v stands for rectilinear acceleration in the y-axis. r stands for the angular acceleration while ψ is the angular velocity. The positions of the hovercraft are given by x and y.

The forces and moments considered are related to the motor propulsion, the air resistance, the friction on the ground, damping and rudder. They act on the system by X, Y, N as exposed in equations (5) to (7).

$$X = F_u - \mu_{sec} N f(u) \frac{u}{\sqrt{u^2+v^2}} - \frac{1}{2}\rho C_d S_u \sqrt{u^2+v^2} u - D_{11} u \tag{5}$$

$$Y = \frac{1}{4A} C_L S_v F_u \delta - \mu_{sec} N f(v) \frac{v}{\sqrt{u^2+v^2}} - \frac{1}{2}\rho C_d S_v \sqrt{u^2+v^2} v - D_{22} v \tag{6}$$

$$N = -L \frac{1}{4A} C_L S_g F_u \delta - D_{33} r \tag{7}$$

where f is a regularization function avoiding discontinuity between the static and the dynamic dry frictions.

Table 1 gives the physical meanings of all the parameters used in (4) to (7) with their SI units. Design parameters, *i.e.*, those which will be used in CSP, are in bold font.

Table 1: Model and design (in bold) parameters

Param.	**Description**	**IS Units**
m	**Mass**	kg
$\mathbf{I_z}$	**Inertia moment in relation to the axis** z	kg.m^2
C_d	Air resistance coefficient	none
S_u	Frontal surface area	m^2
S_v	Side surface area	m^2
ρ	Air density	kg.m^{-3}
μ_{sec}	Ground friction coefficient	none
D_{11}	Damping in relation to the axis x	kg.s^{-1}
D_{22}	Damping in relation to the axis y	kg.s^{-1}
D_{33}	Damping in relation to the axis z	kg.s^{-1}
x_G	x-Coordinate of the gravity center	m
y_G	y-Coordinate of the gravity center	m
C_{F_u}	Propulsion coefficient	none
D	**Propulsion propeller diameter**	m
C_L	Rudder lift coefficient	rad^{-1}
$\mathbf{S_g}$	**Rudder area**	m^2
L	**Distance between CG and the rudder aerodynamic center**	m

5.2 Parameters configuration

Table 2 shows the chosen values, points or intervals, of each of the mathematical model parameters. The design parameters m, I_z, D, S_g, L imply that our research space is in $\mathbb{R}^5$. The domains adopted by these parameters determine a space region where one will perform the task to find a point in this region with the values that lead to a validated design configuration, w.r.t. requirements. In this application, some parameters are taken in continuous domains (*i.e.*, the mass m) and some other are discrete modeling.

Discrete values represent a finite number of physical components that can be used to implement the hovercraft such as the propulsion propeller diameter D. Remark that our algorithm straightforwardly handles both kinds of parameter values.

Table 2: Values of the parameters

Model parameters	
C_d	0.6
S_u	0.1276
S_v	0.2146
ρ	1.225
D_{11}	0
D_{22}	0.2
D_{33}	0.1
x_G	0
y_G	0
C_{F_u}	1.339
C_L	1.891
Design parameters	
m	[1.0, 1.3]
I_z	[0.05, 0.08]
D	{ 0.127, 0.1524, 0.1778, 0.2032, 0.2286 }
S_g	{ 0.0042, 0.005, 0.0059, 0.0067, 0.0075, 0.0084, 0.0101, 0.0117, 0.0126, 0.0134, 0.014, 0.0151, 0.0168, 0.0176, 0.0196, 0.0201, 0.0224, 0.0226, 0.0235, 0.0251, 0.0268, 0.0279, 0.0302, 0.0335, 0.0352, 0.0391, 0.0402, 0.0419, 0.0447, 0.0453, 0.0503, 0.0587, 0.0671, 0.0754 }
L	[0.25 , 0.35]

For example, if we pick two different points in the design parameter space, we are able to simulate a scenario which provides two very different behaviors, see Figure 3.

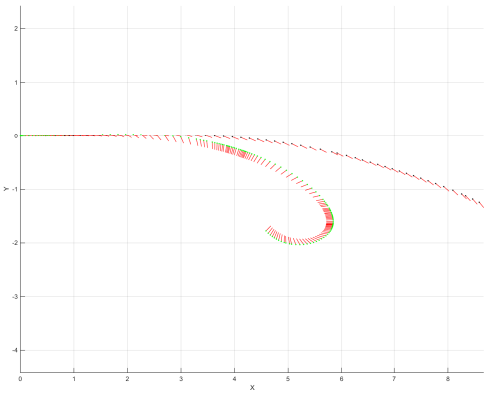


Figure 3: Two simulation solutions with different punctual values of the design parameters.

5.3 Requirements on Behavior

In this section, we show how requirements on the behavior of the hovercraft can be translated into a simulation scenario and a CSP. For the hovercraft case study, we will consider four requirements each of them is described in a separated section.

In the rest of this section, a simulation scenario is defined by *i)* the initial condition associated to (4); *ii)* the control law used to obtain the desired behavior of the hovercraft; and *iii)* a simulation duration. Each CSP will be described by the list of variables, their domains and a set of constraints.

5.3.1 Global constraints

Before describing the main requirements we considered for the hovercraft case study, we note that a dynamical systems is constrained by global constraints depending on its intrinsic characteristics, *e.g.*, its maximal possible speed. In other words, we take into account constraints which represents some limitations on the behavior of the hovercraft for all its possible trajectories. These constraints will naturally be added into all the CSP associated to the requirements we considered and can be also seen as an independent CSP.

We will assume that the maximum speed reached by the hovercraft will be $V_{max} = 10\text{m/s}$ and its maximum angular speed will be $R_{max} = \pi\text{rad/s}$. Moreover, we define some constants useful to describe simulation scenario such as the cruising speed $V_c = 2\text{m/s}$ and the maximum travel distance $D_{\max} = V_{\max} \times t_f$ where t_f is the simulation duration depending on each requirement.

In Table 3 we give the additional constraints which have to be added in each CSP associated to requirements on the behavior of the hovercraft. More precisely, Constraint $C^0_{t,u}$ means that the rectilinear acceleration in the x-axis is positive and bounded, *i.e.*, we cannot go backward. Constraint $C^0_{t,v}$ means that the acceleration in the y-axis of the hovercraft is bounded. Constraint $C^0_{t,r}$ means that the angular acceleration is bounded. Constraints $C^0_{t,x}$ and $C^0_{t,y}$ give a bound on the position w.r.t. the duration of the displacement.

5.3.2 Requirement 1: Cruising Speed

The first requirement on the behavior of the hovercraft is give in Table 4. It represents that the system may have an acceleration allowing to reach the cruising speed V_c before 1 second. Note that constraints $C^1_{t,y}$ and $C^1_{t,\psi}$ mean that the hovercraft follow a line along the x-axis, *i.e.*, their values are closed to zero within a distance of $\varepsilon > 0$.

5.3.3 Requirement 2: Braking Distance

The second requirement is to have the hovercraft stopping within a distance between 3m and 7m, once the engine is turned off after a run in a straight line at cruising speed. Table 5 gives the mathematical formulation in terms of simulation scenario and CSP. Constraint $C^2_{t,u}$ means that the rectilinear speed along x-axis is positive and bounded by 0.1m/s at the end of the simulation. Constraints $C^2_{t,x}$ and $C^2_{t,y}$ mean that the position of the hovercraft are bounded during the simulation.

5.3.4 Requirement 3: Curve

The third requirement specifies that the hovercraft can follow a curve without deriving too much from the desired trajectory. The scenario, given in Table 6, consists to run in a straight line at the cruising speed and turn at a given time. The requirement imposes that the hovercraft keeps a good behavior in a sense of the CSP. Constraints $C^3_{t,x}$ and $C^3_{t,y}$ mean that at the end of the simulation the position of the hovercraft is strictly positive in x and strictly negative in y. In other terms, theses constraints mean that the hovercraft has turn to the right. Note that this requirement is hard to express in terms of radius of curvature that is why we simplified it.

5.3.5 Requirement 4: Perturbation of Symmetry

The fourth requirement consists in verifying the robustness of the design to a possible uncertainty on the mass distribution (expressed by an uncertainty on the gravity center position). For that, the gravity center in y-coordinate is set to $y_G = [-0.2, 0.2]$ which represents 10% of the width of the hovercraft. The

Table 3: CSP of global constraints

Variables:	$\{t, u, v, r, x, y, \psi\}$
Domains:	$\mathbb{R}_+ \times \mathbb{R}^6$
Constraints:	$\{C^0_{t,u}, C^0_{t,v}, C^0_{t,r}, C^0_{t,x}, C^0_{t,y}\}$
$C^0_{t,u}$	$: (t, u(t)) \in [0, t_f] \times [0, V_{max}]$
$C^0_{t,v}$	$: (t, v(t)) \in [0, t_f] \times [-V_{max}, V_{max}]$
$C^0_{t,r}$	$: (t, r(t)) \in [0, t_f] \times [-R_{max}, R_{max}]$
$C^0_{t,x}$	$: (t, x(t)) \in [0, t_f] \times [x_0 - D, x_0 + D]$
$C^0_{t,y}$	$: (t, y(t)) \in [0, t_f] \times [y_0 - D, y_0 + D]$

Table 4: Cruising speed requirement

Requirement 1 – Cruising Speed
`Simulation scenario`
Initial condition $(u_0, v_0, r_0, x_0, y_0, \psi_0) = (0.1, 0, 0, 0, 0, 0)$
Control input $W(t) = W_{V_{max}}, \forall t \in [0, t_f]$ $\delta(t) = 0$
Simulation duration $t_f = 1$ s
`CSP`
Variables $\{t, u, v, r, x, y, \psi\}$
Domains $\mathbb{R}_+ \times \mathbb{R}^6$
Constraints $C^1_{t,y} : (t, y(t)) \in [0, t_f] \times [-\varepsilon, \varepsilon]$ $C^1_{t,\psi} : (t, \psi(t)) \in [0, t_f] \times [-\varepsilon, \varepsilon]$ $C^1_{t,u} : (t = t_f \wedge u(t_f) = V_c)$

Table 5: Braking distance requirement

Requirement 1 – Braking distance
`Simulation scenario`
Initial condition $(u_0, v_0, r_0, x_0, y_0, \psi_0) = (V_c, 0, 0, 0, 0, 0)$
Control input $W(t) = 0, \forall t \in [0, t_f]$ $\delta(t) = 0$
Simulation duration $t_f = 10$ s
`CSP`
Variables $\{t, u, v, r, x, y, \psi\}$
Domains $\mathbb{R}_+ \times \mathbb{R}^6$
Constraints $C^2_{t,u} : (t = t_f) \wedge (u(t) \in [0, 0.1])$ $C^2_{t,x} : (t, x(t)) \in [0, t_f] \times [3, 7]$ $C^2_{t,y} : (t, y(t)) \in [0, t_f] \times [-0.5, 0.5]$

scenario, given in Table 7, imposes to run in a straight line along x-axis at the cruising speed. After 3 seconds, the requirement is that the hovercraft did not drift more than 15cm from x-axis. Constraints $C^4_{t,y}$ and $C^4_{t,\psi}$ mean that for the duration of the simulation the y coordinate and the headings of the hovercraft are bounded.

5.4 Results

Applying this methodology to the hovercraft design, a set of appropriate solutions verifying the exigences was found. A particular design solution was chosen to start the hovercraft construction. This latter is chosen in the middle of the validated region, in order to be as robust as possible to the building/assembly uncertainties.

Chosen solution for building the hovercraft	
m	$[1.06562, 1.075]$
I_z	$[0.0575, 0.065]$
D	$[0.1778, 0.1778]$
S_g	$[0.0352, 0.0352]$
L	$[0.31875, 0.325]$

Table 6: Curve requirement

Requirement 3
`Simulation scenario`
Initial condition $(u_0, v_0, r_0, x_0, y_0, \psi_0) = (V, 0, 0, 0, 0, 0)$
Control input $W(t) = \frac{1}{3} W_{V_{\max}}, \forall t \in [0, t_f]$ $\delta(t) = \frac{\pi}{12} \left(1 + \sin\left(\frac{\pi}{2} t\right)\right)$
Simulation duration $t_f = 3\text{s}$
`CSP`
Variables $\{t, u, v, r, x, y, \psi\}$
Domains $\mathbb{R}_+ \times \mathbb{R}^6$
Constraints $C^3_{t,x} : (t = t_f) \wedge (x(t) \in [1, \infty])$ $C^3_{t,y} : (t = t_f) \wedge (y(t) \in [-\infty, -1])$

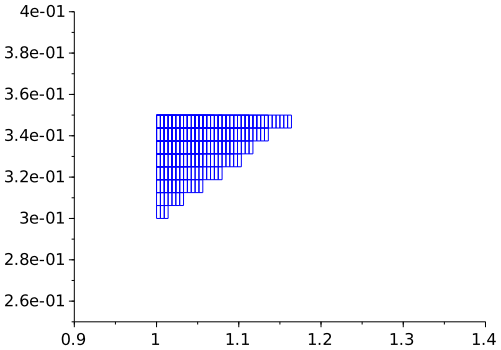
Table 7: Symmetry robustness requirement

Requirement 4
`Simulation scenario`
Initial condition $(u_0, v_0, r_0, x_0, y_0, \psi_0) = (V_c, 0, 0, 0, 0, 0)$
Control input $W(t) = \frac{1}{4} W_{V_{\max}}, \forall t \in [0, t_f]$ $\delta(t) = 0$
Simulation duration $t_f = 3\text{s}$
`CSP`
Variables $\{t, u, v, r, x, y, \psi\}$
Domains $\mathbb{R}_+ \times \mathbb{R}^6$
Constraints $C^4_{t,y} : (t, y(t)) \in [0, t_f] \times [y_0 - 0.15, y_0 + 0.15]$ $C^4_{t,\psi} : (t, y(t)) \in [0, t_f] \times [\psi_0 - \frac{\pi}{16}, \psi_0 + \frac{\pi}{16}]$

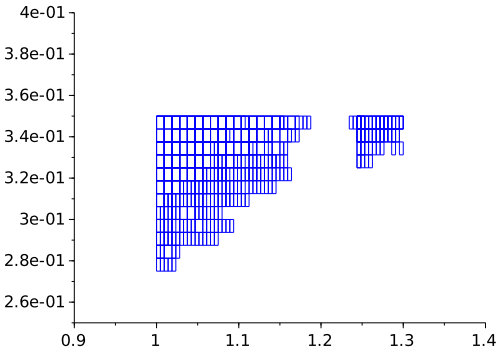
To display the set of all solutions found, a projection from $\mathbb{R}^5$ to $\mathbb{R}^2$ is performed by fixing two discrete parameters D and S_g. We then obtain three figures gathered in Figure 4.

6 Conclusion

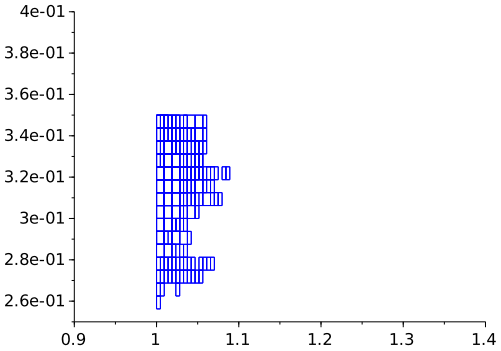
We presented in this paper a tool for the design of complex systems, described by differential equations with bounded uncertainties. In accordance with the requirements, expressed under the constraint satisfaction problem formalism, our method is able to find design parameters. These parameters validate, even in presence of uncertainties, the correspondence between behavior simulation and constraints defined by requirements. This approach has been applied to the design of an hovercraft. The hovercraft is now in construction and a step of validation will then be done. Obviously, the model being a simplification of the real behavior of the hovercraft, we are waiting for a difference between the expected behavior and the reality. Anyway, this difference can be surpassed by a parameter identification process, which may allow to match the model and the real behavior of the hovercraft. This will be the futur work to validate our methodology.



(a)



(b)



(c)

Figure 4: Solution set for $m \times L$, after projection with $I_z = [0.0575, 0.065]$, $D = 0.1778$ and $S_g = 0.0176$ (a); $S_g = 0.0235$ (b); and $S_g = 0.0352$ (c)

References

[1] J. Alexandre dit Sandretto and A. Chapoutot. Validated Explicit and Implicit Runge-Kutta Methods. *Reliable Computing*, 2016. To appear.

[2] S. M. Batill, J. E. Renaud, and X. Gu. Modeling and simulation uncertainty in multidisciplinary design optimization. *AIAA paper*, 4803, 2000.

[3] F. Benhamou, D. McAllester, and P. Van Hentenryck. Clp (intervals) revisited. Technical report, Providence, RI, USA, 1994.

[4] B. S. Blanchard, W. J. Fabrycky, and W. J. Fabrycky. *Systems engineering and analysis*, volume 4. Prentice Hall New Jersey;, 1990.

[5] O. Bouissou, A. Chapoutot, and A. Djoudi. Enclosing temporal evolution of dynamical systems using numerical methods. In *NASA Formal Methods*, number 7871 in LNCS, pages 108–123. Springer, 2013.

[6] J. Cruz and P. Barahona. Constraint reasoning over differential equations. *Applied Numerical Analysis and Computational Mathematics*, 2004.

[7] D. A. DeLaurentis and D. N. Mavris. Uncertainty modeling and management in multidisciplinary analysis and synthesis. In *AIAA Aerospace Sciences Meeting, Paper No. AIAA-2000–422*, 2000.

[8] M. Diez, D. Peri, G. Fasano, and E. F. Campana. Multidisciplinary robust optimization for ship design. In *28th symposium on naval hydrodynamic, Pasadena, Caloifornia, USA*, 2010.

[9] T. I. Fossen. *Guidance and control of ocean vehicles*. John Wiley & Sons, Ltd, 1994.

[10] A. Goldsztejn, O. Mullier, D. Eveillard, and H. Hosobe. Including ordinary differential equations based constraints in the standard CP framework. In *CP*, volume 6308 of *LNCS*, pages 221–235. Springer Berlin Heidelberg, 2010.

[11] E. Hairer, S. P. Norsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer-Verlag, 2nd edition, 2009.

[12] I. Kroo, S. Altus, R. Braun, P. Gage, and I. Sobieski. Multidisciplinary optimization methods for aircraft preliminary design. *AIAA paper*, 4325:1994, 1994.

[13] Y. Lebbah and O. Lhomme. Accelerating filtering techniques for numeric CSPs. *Artificial Intelligence*, 139(1):109 – 132, 2002.

[14] O. Lhomme. Consistency techniques for numeric CSPs. pages 232–238, 1993.

[15] D. N. Mavris and O. Bandte. A probabilistic approach to multivariate constrained robust design simulation. *SAE Technical Paper*, 975508, 1997.

[16] J.-P. Merlet and D. Daney. Appropriate Design of Parallel Manipulators. 2004. INRIA.

[17] R. Moore. *Interval Analysis*. Prentice Hall, 1966.

[18] N. S. Nedialkov, K. Jackson, and G. Corliss. Validated solutions of initial value problems for ordinary differential equations. *Appl. Math. and Comp.*, 105(1):21 – 68, 1999.

[19] N. V. Nguyen, D. Lee, H.-U. Park, and J.-W. Lee. A multidisciplinary robust optimization framework for UAV conceptual design. 118:123–142, 2014.

[20] M. Rueher. Solving continuous constraint systems. In *Invited Talk, Proc. of 8th International Conference on Computer Graphics and Artificial Intelligence*, 2005.

[21] M. D. Stuber and P. I. Barton. Robust simulation and design using parametric interval methods. In *REC*, pages 536–553, 2010.