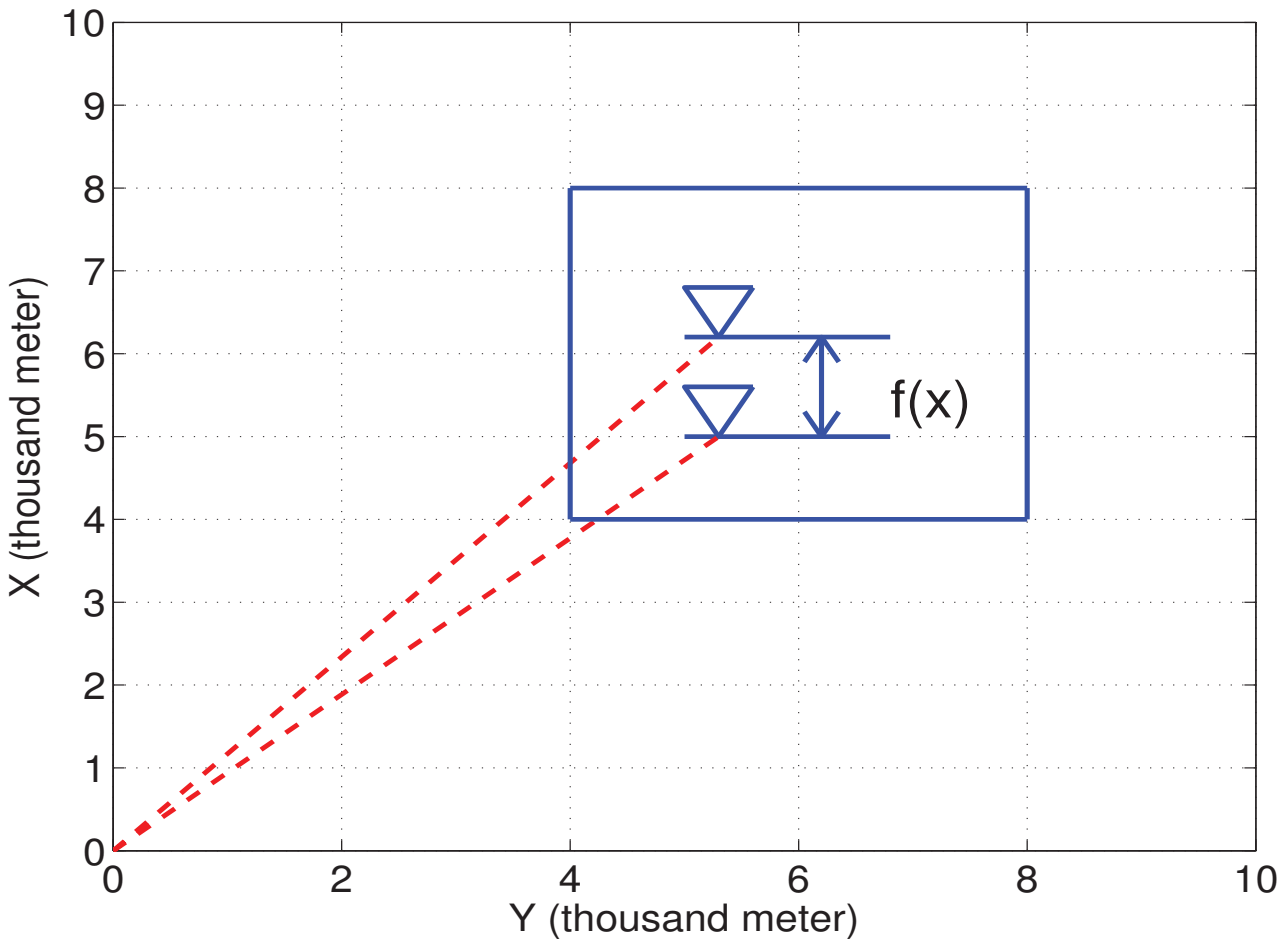
FOI

An optimization-based verification method for comparing a real-time missile model to a reference model

DANIEL SKOOGH, DAG WALLSTRÖM, NICLAS STENSBÄCK,
FREDRIK BEREFELT, PETER M ELIASSON, DANIEL TOURDE



FOI-R--4143--SE
ISSN 1650-1942 November 2015

Daniel Skoogh, Dag Wallström,
Niclas Stensbäck, Fredrik Berefelt,
Peter M Eliasson, Daniel Tourde

An optimization-based verification method for comparing a real-time missile model to a reference model

Titel	Optimeringsbaserad validering av modeller
Title	An optimization-based verification method for comparing a real-time missile model to a reference model
Report no	FOI-R--4143--SE
Month	November
Year	2015
Pages	46
ISSN	ISSN-1650-1942
Customer	Försvarsmakten
Project no	E36506
Approved by	Lars Höstbeck Information and Aeronautical Systems
Division	Information- and Aeronautical Systems
FOI Research area	Weapons, protection and security
Armed Forces R&T area	Weapons and protection

FOI Swedish Defence Research Agency

Abstract

This report describes an optimization-based verification method for comparing a real-time missile model to a reference model. By using a measure of the difference of the output data of the two models, the validation task can be formulated as an optimization problem. Several different measures of trajectories are discussed. The population based stochastic global optimization method Differential Evolution is used in the numerical tests. Further, it is illustrated by examples that large difference in trajectories can be found, by the proposed method.

Keywords

optimization, verification, validation, flight clearance, model, missile, simulation, measure, differential evolution

Sammanfattning

Denna rapport beskriver en metod för optimeringsbaserad validering för jämförelse av en realtidsmodell av robot mot en referensmodell. Genom att använda ett mått på skillnaden på utdata av de två modellerna, valideringen kan formuleras som ett optimeringsproblem. Flera olika mått på skillnaden av trajektorier diskuteras. Den populationsbaserade stokastiska globala optimeringsmetoden Differential Evolution används i de numeriska testerna. Det illustreras genom exempel att stora skillnader i trajectorier kan hittas genom den föreslagna metoden.

Nyckelord

optimering,verifikation, validering, modell, robot, simulering, mått, differential evolution

Contents

1 Introduction

The aim of this report is to describe the basic ideas of a verification method for missile models and also discuss how to view the modelling process associated with the verification process. The numerical tests in this report have been carried out with in house developed MATLAB-based [1] optimization software in combination with frameworks for model implementations. The software is just briefly described, for further information see sections 4.1 and 4.2.

Since the aim is on "basic ideas" we have chosen to describe the models in a context of Ordinary Differential Equations (ODE), which is sufficient for the discussion in this report. The ideas given in this report can easily be applied to models described by more complex mathematics e.g. Differential Algebraic Equations (DAE) or Partial Differential Equations (PDE).

1.1 Background

The aim with the method described in this report is to improve the current validation process by using an efficient and highly confident method that is easy to use in automation of the validation process.

This report describes an optimization-based method for verification of a real-time missile model to a reference model by comparing output data of the corresponding system models. A reference model is a detailed model of the physical missile and usually models all subsystems in detail. It is used as a simulation-based reference of the physical missile. (Often the missile manufacturer supplies the missile buyer with a reference model.) A system model is a model that can be decomposed into several sub-models, e.g. missile, target, and environment models, included are also the initial conditions. Since this method only uses the output of the system models, i.e. uses the models as black boxes, any type of missile models can be compared, e.g. two different reference models. In order for the comparison to be meaningful, the two models must represent the same real world missile with the same purpose, e.g. the same target behaviour, and environment model.

By using a measure of the difference of the output data of two models, the validation task can be formulated as an optimization problem. If the maximum difference is sufficiently small, the real-time missile model is validated against the reference model. By using optimization rather than more traditional gridding techniques the validation process can be done much more efficiently and with higher confidence. The problem with gridding techniques is that the number of grid points grows exponentially with the number of parameters, and that important points can be missed in between the grid points.

The methodology described here is not limited to missile models, and can be applied to models of other aerial vehicles such as unmanned aerial vehicles and aircraft, and thus also other types of dynamical systems, e.g. cars.

Optimization-based verification or validation has been used in other applications, one example is validation of flight control laws of aeroplanes, see next section.

1.2 Optimization and simulation-based flight clearance

The methods discussed later in this report are built on and further developed from similar methods used in flight clearance to verify control laws for manned aircraft, which are briefly described here.

1.2.1 Overview of optimization and simulation-based flight clearance

Optimization has been used in connection to validating flight control laws, both for military and civil aircraft, see e.g. [2, 3]. FOI has been involved in the GARTEUR Flight Mechanics Action Group 11 FM(AG11) [2], and the European Union sponsored project Clearance of Flight Control Laws Using Optimization (COFCLUO) [3]. At FOI, mainly validation based on optimization and simulation of nonlinear aircraft models has been studied. There exists various other optimization-based techniques for validation, e.g. μ-analysis and Lyapunov-based techniques, see [3]. What follows is an overview of validation based on optimization and simulation of nonlinear aircraft models.

Several works of optimisation-based clearance are using simulation of the nonlinear model together with a fixed pilot signal over a parametric uncertainty set. The evaluated criterion is the normal load factor, n_z or angle of attack, α exceedance criterion. In [4, 5, 6, 7] the pilot signal is a fixed ramp or step function, and in [8] the pilot signal is the "klonk" signal, a fixed sequence designed to cause the worst possible behaviour.

None of the above methods covers the case of a general time dependent pilot input signal.

A few works have been done with a nonlinear simulation model together with parameterised pilot input signals [9, 10, 11, 12, 13]. It should be noted that the methods discussed in [9, 10] only use a finite discrete parameter set. In the works [11, 12, 13], the pilot signals are discretized by a finite number of parameters. However, differently from [9, 10], the parameters are real valued and can take all values in a finite interval.

A work dealing with clearance of flight control laws using multi-objective optimization is given in [14].

In the above cited works, global, local and hybrid optimization methods were used. Among the most robust and successful methods used are the global population-based stochastic optimization methods Differential Evolution and Evolutionary Strategies. Local methods such as quasi-Newton, pattern search and simplex methods have also been employed. The resulting objective function arising from flight clearance is often a nonlinear function with multiple local optima. Thus global methods are needed in order to find the worst case. However, there exists no optimization method that with absolute certainty can find the global optimum for a general nonlinear function in finite time. So in practice, all use of global methods should be done with caution and care. Local methods are usually much more efficient than global methods, but they converge towards a local optimum. They can still play an important role in that they can be used in, initial investigation or used together with global methods in hybrid methods in order to speed up the computations.

For more information about global optimization methods see e.g. [15, 16, 17, 18, 19] and for local [20, 21, 22, 23].

In this work we have chosen Differential Evolution [24] as the optimization method. The reason for this

- It is a robust method.
- Has the ability to find global optimum (approximately).
- Rather good convergence properties.
- It is easy to set up and use.
- Straightforward to use in parallel computation (future work).

In the next two subsections, we will discuss validation of flight control laws based on optimization and simulation of a nonlinear simulation model.

1.2.2 Clearance criteria

In order to capture the worst possible aircraft behaviour, simplified measures need to be defined, e.g. taking the maximum of a relevant flight variable, e.g. the angle of attack over a given time period $[0, T]$

$$c(\boldsymbol{p}) := \max_{t \in [0,T]} \alpha(t, \boldsymbol{p}) \tag{1.1}$$

where $\boldsymbol{p}$ is a parameter-vector. The parameters may consist of pilot signal parameters, flight condition parameters and uncertainty parameters. By maximizing the criterion function $c(\boldsymbol{p})$ over a given parameter set $\mathcal{P}$, the worst-case behaviour is determined. The worst case behaviour is acceptable if it is below a certain value $c_{\max}$

$$c(\boldsymbol{p}_{\max}) \leq c_{\max} \tag{1.2}$$

The clearance criteria is fulfilled if the parameter vector $\boldsymbol{p}_{\max}$ corresponding to the worst case fulfils the above inequality.

In [25] a more complex criterion is described, several flight variables are captured in the same criteria, angle of attack α, side slip angle β, and normal load factor n_z.

1.2.3 Optimization problem

We will here discuss how to formulate the task of finding the parameter vector $\boldsymbol{p}_{\max}$ in the previous section as an optimization problem.

Let the objective function be equal to the criterion function $c(\boldsymbol{p})$ in the previous section. The worst-case behaviour is determined through solving the optimization problem

$$\max_{\boldsymbol{p} \in \mathcal{P}} c(\boldsymbol{p}) \tag{1.3}$$

The objective function $c(\boldsymbol{p})$ is defined through the solution $\boldsymbol{x}(\cdot, \boldsymbol{p})$ of a parameter-dependent ordinary differential equation with initial conditions

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{p}), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0$$

and a measure [1] $h(\boldsymbol{x}(\cdot, \boldsymbol{p}))$ on the solution $\boldsymbol{x}(\cdot, \boldsymbol{p})$

$$c(\boldsymbol{p}) = h(\boldsymbol{x}(\cdot, \boldsymbol{p})) \tag{1.4}$$

Notations

A scalar is defined by a lowercase letter v. A finite dimensional row or column vector is defined as a boldface lowercase letter $\boldsymbol{x}$. Element number i of the vector $\boldsymbol{x}$ is referred to as x_i. E.g the column vector $\boldsymbol{y}$ can be defined by

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_i \\ \vdots \\ y_m \end{bmatrix} = [y_1, \cdots, y_i, \cdots, y_m]^T \tag{1.5}$$

A matrix is defined by uppercase bold letter $\boldsymbol{X}$. Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m \in \mathcal{R}^{1 \times n}$ be row vectors of the same dimension, then the matrix $\boldsymbol{X}$ can be decomposed

1. h is not a measure in the standard mathematical sense

into

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_i \\ \vdots \\ \boldsymbol{x}_m \end{bmatrix} \in \mathcal{R}^{m \times n} \tag{1.6}$$

2 Model presumptions

The purpose of this chapter is to define a system model and its sub-models, and to establish a notation and a language that is used later on. [1]

Since the aim is on "basic ideas" we have chosen to describe the models in a context of ordinary differential equations, which is sufficient for the discussion.

2.1 Parameter-dependent system model

In this report we consider parameter-dependent system models of the type

$$\begin{aligned} &\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{p}), \;\; \boldsymbol{x}(t=0) = \boldsymbol{g}(\boldsymbol{p}) \\ &\boldsymbol{f}, \boldsymbol{x}, \boldsymbol{g} \in \mathcal{R}^{n\times 1}, \;\; \boldsymbol{p} \in \mathcal{R}^{m\times 1}, \;\; t \in \mathcal{R}, \;\; t \geq 0 \end{aligned} \tag{2.1}$$

in order to study parameter-dependent missile-target duels. The aim is to verify a real-time missile model against a reference model, by comparing the output of the corresponding system models. In more details how this is done is described in chapter 3.

The system model (2.1) is described by parameter-dependent first order Ordinary Differential Equations (ODEs) with initial conditions at time $t = 0$. Such problems can also be referred to as parameter-dependent Initial Value Problems (IVP). A more general description of the state space equations of missile-target duels than can be done with parameter-dependent ODEs ($\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{p})$), can e.g. be done with parameter-dependent Differential Algebraic Equations (DAEs) of the form

$$\boldsymbol{f}(t, \boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{p}) = 0 \tag{2.2}$$

In this report, we will restrict the discussion of system models that can be described by IVPs. Many of the results described in this report can be generalised to other types of system models, e.g. system models described by DAEs.

The system model can be decomposed into several sub-models, e.g missile model, target model and environment model.

$$\begin{aligned} \dot{\boldsymbol{x}}_{\text{missile}} &= \boldsymbol{f}_{\text{missile}}(t, \boldsymbol{x}, \boldsymbol{p}) \\ \dot{\boldsymbol{x}}_{\text{target}} &= \boldsymbol{f}_{\text{target}}(t, \boldsymbol{x}, \boldsymbol{p}) \\ \dot{\boldsymbol{x}}_{\text{environment}} &= \boldsymbol{f}_{\text{environment}}(t, \boldsymbol{x}, \boldsymbol{p}) \end{aligned} \tag{2.3}$$

where the initial conditions are given by

$$\begin{aligned} \boldsymbol{x}_{\text{missile}}(t=0) &= \boldsymbol{g}_{\text{missile}}(\boldsymbol{p}) \\ \boldsymbol{x}_{\text{target}}(t=0) &= \boldsymbol{g}_{\text{target}}(\boldsymbol{p}) \\ \boldsymbol{x}_{\text{environment}}(t=0) &= \boldsymbol{g}_{\text{environment}}(\boldsymbol{p}) \end{aligned} \tag{2.4}$$

and the state vector $\boldsymbol{x}$ can be divided into

$$\boldsymbol{x}^T = [\boldsymbol{x}^T_{\text{missile}} \boldsymbol{x}^T_{\text{target}} \boldsymbol{x}^T_{\text{environment}}] \tag{2.5}$$

We will here define the *missile model* as the differential equations describing its behaviour $\dot{\boldsymbol{x}}_{\text{missile}} = \boldsymbol{f}_{\text{missile}}(t, \boldsymbol{x}, \boldsymbol{p})$ where the vector $\boldsymbol{x}$ is the whole state vector (2.5). (The target model and environment model are defined in a similar

1. Other work might define things differently.

way.) To summarise, for our purposes a common case is that a system model can be decomposed into missile, target and environment models and the initial conditions. Note that, in our definitions, initial conditions are not included in the sub-models (missile, target and environment), but the initial conditions are included in the system model.

Although local analysis can be carried out around a given state space vector $\boldsymbol{x}$ and parameter vector $\boldsymbol{p}$, the missile model is not simulated alone. Any simulation involving the missile model needs to be done within a system model. This is a consequence of our definitions.

Even the local analysis might need assumption about the environment and target due to the whole state vector $\boldsymbol{x}$ being input data to the state evolution function of the missile $\boldsymbol{f}_{\text{missile}}(t, \boldsymbol{x}, \boldsymbol{p})$.

2.2 Parameters

2.2.1 System model parameters

Parameters of the system model can be divided into the main groups: missile model parameters, target model parameters, environment parameters and scenario parameters.

Missile model parameters $\boldsymbol{p}_{\text{missile}}$ include e.g. missile engine parameters, missile target seeker parameters, and missile control law parameters.

Target model parameters $\boldsymbol{p}_{\text{target}}$ include e.g. target behaviour parameters, target engine parameters, and target control law parameters.

Environment parameters $\boldsymbol{p}_{\text{env}}$ include e.g. parameters to the environment model.

Scenario parameters $\boldsymbol{p}_{\text{sce}}$ include initial conditions for the missile, target and environment.

As a consequence of our definitions of missile, target and environment models, initial conditions are not included in those model definitions. Thus, here all initial conditions are considered as scenario parameters. Furthermore target behaviour parameters are considered as part of the target model parameters.

Different authors may group and define parameters differently. Note, how parameters are grouped and defined in this subsection is not that important in order to understand the concepts in this report. However, it is helpful to define different groups of parameters, and thus stick to the definitions.

2.2.2 Evaluation and optimization parameters

In this subsection parameters will be grouped in other ways for other purposes than in the previous subsection. Both views exist at the same time.

Evaluation parameters $\boldsymbol{p}_{\text{eval}}$ are parameters set within a framework (here a MATLAB based framework: Frameopt 1.0, see section 4.1) and external to the system model. Evaluation parameters may belong to any group of the system model parameters, i.e. missile model, target model, environment model and scenario parameters. The point is to put those parameters that one may want to change into the group evaluation parameters. On the practical side, the evaluation parameters are accessible to the framework through an interface. In many cases, it is not practical to put all system model parameters into the group of evaluation parameters.

Optimization parameters $\boldsymbol{p}_{\text{opt}}$ are parameters of the objective function and are a subset of the evaluation parameters.

Fixed evaluation parameters $\boldsymbol{p}_{\text{fixed}}$ are evaluation parameters that do not belong to group of optimization parameters.

Evaluation parameters can be decomposed into optimization parameters

and fixed evaluation parameters.

$$\boldsymbol{p}_{\text{eval}} = \begin{bmatrix} \boldsymbol{p}_{\text{opt}} \\ \boldsymbol{p}_{\text{fixed}} \end{bmatrix} \tag{2.6}$$

Internal system model parameters $\boldsymbol{q}$ are parameters of the system model that are not chosen to be included in the evaluation parameters.

All parameters of the system model either belong to the evaluation parameters $\boldsymbol{p}_{\text{eval}}$ or the internal system model parameters $\boldsymbol{q}$.

The system model can now be written as

$$\begin{aligned} &\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{p}_{\text{eval}}, \boldsymbol{q}), \;\; \boldsymbol{x}(t=0) = \boldsymbol{g}(\boldsymbol{p}_{\text{eval}}, \boldsymbol{q}) \\ &\boldsymbol{f}, \boldsymbol{x}, \boldsymbol{g} \in \mathcal{R}^{n\times 1}, \;\; \boldsymbol{p}_{\text{eval}} \in \mathcal{R}^{m\times 1}, \;\; \boldsymbol{q} \in \mathcal{R}^{l\times 1}, \;\; t \in \mathcal{R}, \;\; t \geq 0 \end{aligned} \tag{2.7}$$

2.3 Optimization Problem

We will here describe how to transform the system model (2.7) into an equivalent system model more suitable for optimization, and also how to state the corresponding optimization problem. Note, the real system is the same (it is the view that is changed).

The first step consists of tuning the internal system model parameters to adequate values. They will remain constant throughout the optimization, and possibly remain constant throughout several optimization runs.

The next step is to pick out the optimization parameters $\boldsymbol{p}_{\text{opt}}$ among the evaluation parameters $\boldsymbol{p}_{\text{eval}}$. On a practical side, this is quite easily managed with the framework Frameopt 1.0, see section 4.1. The evaluation parameters $\boldsymbol{p}_{\text{eval}}$ can, as previously stated in subsection 2.2.2, be divided into optimization parameters $\boldsymbol{p}_{\text{opt}}$ and fixed evaluation parameters $\boldsymbol{p}_{\text{fixed}}$. As the name indicates, fixed evaluation parameters are fixed during an optimization run.

Before the optimization run, the fixed evaluation parameters $\boldsymbol{p}_{\text{fixed}}$ are tuned to adequate values. This is similar to tuning the internal system model parameters $\boldsymbol{q}$. But, note on the practical side, this is easily managed by the framework Frameopt 1.0. The operator does not need to know anything about the implementation of the system model.

Now the system model (2.7) is redefined into

$$\begin{aligned} &\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{p}_{\text{opt}}), \;\; \boldsymbol{x}(t=0) = \boldsymbol{g}(\boldsymbol{p}_{\text{opt}}) \\ &\boldsymbol{f}, \boldsymbol{x}, \in \mathcal{R}^{n\times 1}, \;\; \boldsymbol{p}_{\text{opt}} \in \mathcal{R}^{m\times 1}, \;\; t \in \mathcal{R}, \;\; t \geq 0 \end{aligned} \tag{2.8}$$

where all other parameters but $\boldsymbol{p}_{\text{opt}}$ are suppressed, but they still exist.

The next step is to formulate an objective function $c(\boldsymbol{p}_{\text{opt}})$. This is done through defining a measure $h(\boldsymbol{x}(\cdot, \boldsymbol{p}_{\text{opt}}))$ on the solution $\boldsymbol{x}(\cdot, \boldsymbol{p}_{\text{opt}})$ to the IVP (2.8). Thus the objective function is

$$c(\boldsymbol{p}_{\text{opt}}) = h(\boldsymbol{x}(\cdot, \boldsymbol{p}_{\text{opt}})) \tag{2.9}$$

For example, the measure $h(\boldsymbol{x}(\cdot, \boldsymbol{p}_{\text{opt}}))$ could be the maximum angle of attack $\alpha_{\max}$, and in this case the objective function [2] is

$$c(\boldsymbol{p}_{\text{opt}}) := \max_{t\in[0,T]} \alpha(t, \boldsymbol{p}_{\text{opt}}) \tag{2.10}$$

Finally, the optimization problem can be stated as

$$\max_{\boldsymbol{p}_{\text{opt}}\in\mathcal{P}} c(\boldsymbol{p}_{\text{opt}}) \tag{2.11}$$

2. In this case, a criterion function is used as an objective function

A more detailed version of this section with intermediate steps is presented in appendix A.1.

3 Method

The purpose of this chapter is to describe our new method for optimization-based verification of a real-time missile model to a reference model by comparing output data of the corresponding system models.

3.1 Introduction

Different models of the same physical missile are developed for different purposes. A *reference model* is usually a detailed model of a missile and its subsystems such as control system, actuators and target seeker. The simulation time of a scenario using a reference model is often longer than a real-time. A *real-time model* is developed to execute in real-time, e.g. to be used in connection of pilot training using simulators. In order for a real-time model to fulfil the real-time demand it is usually considerably less complex than a reference model.

The purpose here is to develop a method that compares two different missile models of the same physical missile by comparing the output of the simulations of the corresponding system models. With a system model we mean a composite model consisting of missile model, target model, environment model and initial conditions, see section 2.1.

The two models may

- Be implemented in different frameworks like Merlin and ACSL.
- Use different modelling concept like 3-DOF [1] and 5-DOF.
- Use different degree of accuracy for actuator modelling.

If comparing the output of two system models should be meaningful then it is important to

- Have the same scenario.
- Have the same target behaviour.
- Have the same environment model.

It is important to think through so that the conditions of the two system models are as similar as possible, e.g. one should "compare apples to apples".

The driving idea is in a situation when there exists a reference missile model that is already validated to sufficient degree and there also exists a real-time missile model that is not validated. If the maximum difference of the output of the two system models with respect to a given difference measure and a parameter set is sufficiently small, then the real-time missile model is verified against the reference missile model with respect to the given difference measure, parameter set and the implicit conditions of the system models, e.g. the used environment model. Note that it is actually the output of the system models that are compared.

The method itself can only answer how much the maximum difference between the two system models is with respect to a given measure. If it is appropriate or not to use the method in connection to verification depends on other sources of information of the two system models. For example, one of the models can be a validated reference model from a missile company. Of course, this does not rule out the possibility of errors in the reference model. The method developed here should be used with caution.

The measure of the difference of the output of the two models can be based on

1. DOF = Degree Of Freedom

– A distance measure between the two trajectories.
– A distance measure between the time derivative of two trajectories.
– The distance between the impact points of the two models.

or a combination of the above.

3.2 Problem description

We will here describe the steps needed to be done just prior to formulate an optimization problem.

Consider the two system models

$$\begin{aligned} \dot{\boldsymbol{x}}_1 &= \boldsymbol{f}_1(t, \boldsymbol{x}_1, \boldsymbol{p}_{\text{eval}}, \boldsymbol{q}_1), \quad \boldsymbol{x}_1(t=0) = \boldsymbol{g}_1(\boldsymbol{p}_{\text{eval}}, \boldsymbol{q}_1) \quad \text{Model 1} \\ \dot{\boldsymbol{x}}_2 &= \boldsymbol{f}_2(t, \boldsymbol{x}_2, \boldsymbol{p}_{\text{eval}}, \boldsymbol{q}_2), \quad \boldsymbol{x}_2(t=0) = \boldsymbol{g}_2(\boldsymbol{p}_{\text{eval}}, \boldsymbol{q}_2) \quad \text{Model 2} \end{aligned} \tag{3.1}$$

where

$$\begin{aligned} &\boldsymbol{f}_1, \boldsymbol{x}_1, \boldsymbol{g}_1 \in \mathcal{R}^{n_1 \times 1}, \quad \boldsymbol{f}_2, \boldsymbol{x}_2, \boldsymbol{g}_2 \in \mathcal{R}^{n_2 \times 1}, \quad \boldsymbol{p}_{\text{eval}} \in \mathcal{R}^{m \times 1} \\ &\boldsymbol{q}_1 \in \mathcal{R}^{l_1 \times 1}, \quad \boldsymbol{q}_2 \in \mathcal{R}^{l_2 \times 1}, \quad t \in \mathcal{R}, \quad t \geq 0 \end{aligned} \tag{3.2}$$

The parameter-vector $\boldsymbol{p}_{\text{eval}}$ consists of evaluation parameters, they are the same for the two models. The parameter vectors $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ consist of internal model parameters of the two system models.

The first step consists of tuning the internal parameters of the two models $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$, such that the essential output of the two models are as equal as possible, e.g. the trajectories of the two models should be as similar as possible.

The next step is to pick out the optimization parameters among the evaluation parameters. The two system models are now redefined into

$$\begin{aligned} \dot{\boldsymbol{x}}_1 &= \boldsymbol{f}_1(t, \boldsymbol{x}_1, \boldsymbol{p}_{\text{opt}}), \quad \boldsymbol{x}_1(t=0) = \boldsymbol{g}_1(\boldsymbol{p}_{\text{opt}}) \quad \text{Model 1} \\ \dot{\boldsymbol{x}}_2 &= \boldsymbol{f}_2(t, \boldsymbol{x}_2, \boldsymbol{p}_{\text{opt}}), \quad \boldsymbol{x}_2(t=0) = \boldsymbol{g}_2(\boldsymbol{p}_{\text{opt}}) \quad \text{Model 2} \end{aligned} \tag{3.3}$$

where

$$\boldsymbol{f}_1, \boldsymbol{x}_1, \boldsymbol{g}_1 \in \mathcal{R}^{n_1 \times 1}, \quad \boldsymbol{f}_2, \boldsymbol{x}_2, \boldsymbol{g}_2 \in \mathcal{R}^{n_2 \times 1}, \quad \boldsymbol{p}_{\text{opt}} \in \mathcal{R}^{m \times 1}, \quad t \in \mathcal{R}, \quad t \geq 0 \tag{3.4}$$

Note that it is only the optimization parameters $\boldsymbol{p}_{\text{opt}}$ that are visible in equation (3.3), all other parameters are suppressed, but they still exist.

A more detailed version of this section with intermediate steps is put in appendix A.2.

3.3 Criteria

In order to capture how much two models differ, simplified difference measures need to be developed on the output of the models during simulations, e.g. the difference of the impact points (where the missile hits the target),

$$c(\boldsymbol{p}) = \| \boldsymbol{x}_{1,impact} - \boldsymbol{x}_{2,impact} \| \tag{3.5}$$

where $\boldsymbol{p}$ is a parameter vector and $\boldsymbol{x}_{1,impact}$, $\boldsymbol{x}_{2,impact}$ are the impact points for model 1 and 2 respectively. The difference measure (3.5) will be referred to as a criterion function.

By maximizing the criterion function $c(\boldsymbol{p})$ over a given parameter set $\mathcal{P}$ the largest difference between the two models is determined with respect to used difference measure.

The largest difference of the two models is acceptable if it is below a certain value $c_{\max}$

$$c(\boldsymbol{p}_{\max}) \leq c_{\max} \tag{3.6}$$

In general the criterion function depends on the whole solutions $\boldsymbol{x}_1(\cdot)$ and $\boldsymbol{x}_2(\cdot)$ of the IVPs (3.3) on two finite time intervals $[0, T_1]$ and $[0, T_2]$ respectively. The criterion function can thus be written as

$$c(\boldsymbol{p}) = h(\boldsymbol{x}_1(\cdot, \boldsymbol{p}), \boldsymbol{x}_2(\cdot, \boldsymbol{p})) \tag{3.7}$$

where $h(\boldsymbol{x}_1(\cdot, \boldsymbol{p}_{\text{opt}}), \boldsymbol{x}_2(\cdot, \boldsymbol{p}_{\text{opt}}))$ is a measure between the solutions $\boldsymbol{x}_1(\cdot, \boldsymbol{p}_{\text{opt}})$ and $\boldsymbol{x}_2(\cdot, \boldsymbol{p}_{\text{opt}})$.

3.4 Optimization problem

As the objective function $c(\boldsymbol{p}_{\text{opt}})$ in the optimization problem (3.8), we use a criterion function, e.g. the distance between the impact points (3.5).

The largest difference between model 1 and model 2 with respect to the objective function $c(\boldsymbol{p}_{\text{opt}})$ is determined through solving the optimization problem

$$\max_{\boldsymbol{p}_{\text{opt}} \in \mathcal{P}} c(\boldsymbol{p}_{\text{opt}}) \tag{3.8}$$

The objective function $c(\boldsymbol{p}_{\text{opt}})$ is defined through the solutions $\boldsymbol{x}_1(\cdot, \boldsymbol{p}_{\text{opt}})$ and $\boldsymbol{x}_2(\cdot, \boldsymbol{p}_{\text{opt}})$ of two parameter-dependent ODE, see (3.3), and a difference measure $h(\boldsymbol{x}_1(\cdot, \boldsymbol{p}_{\text{opt}}), \boldsymbol{x}_2(\cdot, \boldsymbol{p}_{\text{opt}}))$ between the solutions $\boldsymbol{x}_1(\cdot, \boldsymbol{p}_{\text{opt}})$ and $\boldsymbol{x}_2(\cdot, \boldsymbol{p}_{\text{opt}})$

$$c(\boldsymbol{p}_{\text{opt}}) = h(\boldsymbol{x}_1(\cdot, \boldsymbol{p}_{\text{opt}}), \boldsymbol{x}_2(\cdot, \boldsymbol{p}_{\text{opt}})) \tag{3.9}$$

3.5 Composite criteria function

The criterion function can be based on several criteria functions, in this case we will call it a composite criterion function. For example the criterion function can be a weighted sum of criteria functions

$$c(\boldsymbol{p}) = \sum w_i c_i(\boldsymbol{p}), \; w_i > 0 \tag{3.10}$$

This is the only type of composite criteria function that we will discuss in this report. The composite criteria function can be used as an objective function in an optimization problem (3.8), but note also that it has a special meaning in connection with multi-objective optimization that we will discuss in the next section.

3.6 Multi-objective optimization

This section is closely based on section 4.10 optimisation, in [26].

Consider the multi-objective optimization problem

$$\max_{\boldsymbol{p}_{\text{opt}} \in \mathcal{P}} \boldsymbol{c}(\boldsymbol{p}_{\text{opt}}) \tag{3.11}$$

where

$$\boldsymbol{c} \in \mathcal{R}^{n \times 1}, \;\; \boldsymbol{p}_{\text{opt}} \in \mathcal{R}^{m \times 1} \tag{3.12}$$

Only in special cases is it possible to find a parameter vector $\boldsymbol{p}_{\text{opt}}$ such that all objective functions $c_i(\boldsymbol{p}_{\text{opt}})$ are simultaneously minimised. Instead one wish to compute a subset $\mathcal{P}'$ of $\mathcal{P}$ such that no member in $\mathcal{P}'$ is dominated by any member of $\mathcal{P}$. Such set $\mathcal{P}'$ is called a Pareto optimal set.

A parameter vector $\boldsymbol{p}_1$ is said to *dominate* another parameter vector $\boldsymbol{p}_2$ if it is no worse in all objective $c_i(\boldsymbol{p})$ and is strictly better for at least one objective.

By computing a Pareto optimal set one has a set of solutions that are equally good with respect to the objectives and one can choose one particular solution based on higher order information.

In practice, for real world problem it is hard or impossible to compute the whole Pareto-optimal set. However, it is possible to approximate it by using evolutionary algorithms, see [27]. Evolutionary algorithms iterate with a population of points instead of a single point as in traditional gradient based optimization. So it is possible to obtain a set of points in the Pareto optimal set, and in this way approximate it.

It is possible to compute one member of the Pareto optimal set by maximizing the function (3.10) ($w_i > 0$) with respect to the parameter set $\mathcal{P}$, i.e. converting the multi-objective optimization problem into a single-objective optimization problem. In general, it is a difficult problem to choose the weights w_i, see [27]. For multi-objective optimization in connection to flight clearance, see [14].

single-objective optimization is considerably computationally more efficient than multi-objective optimization. However, multi-objective optimization gives more information, and thus is better suited for trade-off analysis than single-objective optimization.

Note that no algorithm capable of multi-objective optimization is implemented in the software toolbox Globopt 3.0, see section 4.2.

3.7 Distance measure between trajectories

Much of the material presented in this section is a further development from the ideas presented in [28].

In this section we will develop distance measures between two trajectories $\boldsymbol{x}_1(\cdot)$ and $\boldsymbol{x}_1(\cdot)$, e.g. the two trajectories may correspond to the missile trajectories of system models 1 and 2 (3.3). We will assume that a trajectory $\boldsymbol{x}(\cdot)$ exists on a finite time interval $t \in [0, T]$. Further, in order to simplify the notation, the parameter dependency of trajectories will be suppressed, e.g. we will use the notation $\boldsymbol{x}(\cdot)$ of a trajectory instead of $\boldsymbol{x}(\cdot, \boldsymbol{p})$. Here, a distance measure is a difference measure applied to space trajectories.

We will here define a *distance measure* $d(\boldsymbol{x}_1(\cdot), \boldsymbol{x}_2(\cdot))$ between two trajectories $\boldsymbol{x}_1(\cdot)$ and $\boldsymbol{x}_2(\cdot)$ as follows

$$
\begin{aligned}
d(\boldsymbol{x}_1(\cdot), \boldsymbol{x}_2(\cdot)) &\in \mathcal{R} && (3.13)\\
d(\boldsymbol{x}_1(\cdot), \boldsymbol{x}_2(\cdot)) &\geq 0 && (3.14)\\
d(\boldsymbol{x}_1(\cdot), \boldsymbol{x}_2(\cdot)) &= 0 \text{ iff } \boldsymbol{x}_1(\cdot) = \boldsymbol{x}_2(\cdot) && (3.15)\\
d(\boldsymbol{x}_1(\cdot), \boldsymbol{x}_2(\cdot)) &= d(\boldsymbol{x}_2(\cdot), \boldsymbol{x}_1(\cdot)) && (3.16)
\end{aligned}
$$

The first requirement states that the distance measure should be real valued, the second that it should be greater than or equal to zero, the third that it should be equal to zero if and only if the the two trajectories are equal, and the fourth that the measure is symmetric.

Here, we define two trajectories to be equal if

$$\boldsymbol{x}_1(t) = \boldsymbol{x}_2(t), \quad \forall t \in [0, T], \quad T = T_1 = T_2 \tag{3.17}$$

Note, the above requirements are necessary but not sufficient in order to construct a good measure. There are also other considerations for distance measures , for example

- Computationally efficient.
- Continuous.
- Differentiable (not necessary for, e.g. differential evolution).
- The measure should give a larger value when the difference between the trajectories is larger.

Note that the requirements we have on the distance measure here are due to practical and computational issues, i.e. they do not follow standard mathematical requirements about distance measures. For example, the Euclidean distance measure between two vectors

$$d_e(\boldsymbol{x}_1, \boldsymbol{x}_2) = \| \boldsymbol{x}_1 - \boldsymbol{x}_2 \|, \quad \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{R}^{n\times 1} \tag{3.18}$$

in addition to the stated requirements (3.13) throughout (3.16), it fulfils the inequality

$$d_e(\boldsymbol{x}_1, \boldsymbol{x}_3) \leq d_e(\boldsymbol{x}_1, \boldsymbol{x}_2) + d_e(\boldsymbol{x}_2, \boldsymbol{x}_3) \tag{3.19}$$

We do not require our distance measure to fulfil the corresponding inequality for trajectories.

We will now go on and construct some measures.

The length of a trajectory

$$\boldsymbol{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} \tag{3.20}$$

is given by

$$l(\boldsymbol{x}) = \int_0^T \| \boldsymbol{x}^{'}(\tau) \| \, d\tau \tag{3.21}$$

where

$$\| \boldsymbol{x}^{'}(\tau) \| = \sqrt{x_1^{'}(\tau)^2 + x_2^{'}(\tau)^2 + x_3^{'}(\tau)^2} \tag{3.22}$$

The trajectory $\bar{x}(\cdot)$ is defined to be the average trajectory of trajectory one and two as follows

$$\bar{\boldsymbol{x}}(t) = \frac{1}{2}(\boldsymbol{x}_1(t) + \boldsymbol{x}_2(t)), \quad t \in [0, \min(T_1, T_2)] \tag{3.23}$$

By integrating the distance between the points $\boldsymbol{x}_1(t)$ and $\boldsymbol{x}_2(t)$ on the trajectories one and two along the average trajectory, we get

$$d_1(\boldsymbol{x}_1(\cdot), \boldsymbol{x}_2(\cdot)) = \int_0^{\min(T_1,T_2)} \| \boldsymbol{x}_1(\tau) - \boldsymbol{x}_2(\tau) \| \| \bar{\boldsymbol{x}}^{'}(\tau) \| \, d\tau \tag{3.24}$$

which is a measure of the accumulated distance between trajectories one and two. Divide (3.24) by the length of the average trajectory and we get a distance measure of the average distance between trajectories one and two

$$d_2(\boldsymbol{x}_1(\cdot), \boldsymbol{x}_2(\cdot)) = \frac{\int_0^{\min(T_1,T_2)} \| \boldsymbol{x}_1(\tau) - \boldsymbol{x}_2(\tau) \| \| \bar{\boldsymbol{x}}^{'}(\tau) \| \, d\tau}{\int_0^{\min(T_1,T_2)} \| \bar{\boldsymbol{x}}^{'}(\tau) \| \, d\tau} \tag{3.25}$$

One disadvantage of the above distance measures is that it does not take into account the trajectory that exists in the time interval $[\min(T_1, T_2), \max(T_2, T_2)]$. One remedy is built on measure the shortest distance from a point on trajectory one to trajectory two

$$\min_t \| \boldsymbol{x}_1(\tau) - \boldsymbol{x}_2(t) \| \tag{3.26}$$

and a similar measure from trajectory two to one. The final distance measure built on the above ideas is as follows

$$\begin{aligned} & d_3(\boldsymbol{x}_1(\cdot), \boldsymbol{x}_2(\cdot)) \\ =& \frac{1}{2}\Big(\int_0^{T_1} \min_t \| \boldsymbol{x}_1(\tau) - \boldsymbol{x}_2(t) \| \| \boldsymbol{x}_1^{'}(\tau) \| \, d\tau + \int_0^{T_2} \min_t \| \boldsymbol{x}_2(\tau) - \boldsymbol{x}_1(t) \| \| \boldsymbol{x}_2^{'}(\tau) \| \, d\tau\Big) \end{aligned} \tag{3.27}$$

normalised by the length of trajectory one and two, and we obtain the normalised distance measure

$$d_4(\boldsymbol{x}_1(\cdot),\boldsymbol{x}_2(\cdot)) = \frac{\int_0^{T_1} \min_t \parallel \boldsymbol{x}_1(\tau)-\boldsymbol{x}_2(t) \parallel \parallel \boldsymbol{x}_1'(\tau) \parallel d\tau + \int_0^{T_2} \min_t \parallel \boldsymbol{x}_2(\tau)-\boldsymbol{x}_1(t) \parallel \parallel \boldsymbol{x}_2'(\tau) \parallel d\tau}{\int_0^{T_1} \parallel \boldsymbol{x}_1'(\tau) \parallel d\tau + \int_0^{T_2} \parallel \boldsymbol{x}'(\tau) \parallel d\tau} \tag{3.28}$$

One disadvantage of the above two distance measures is that they do not fulfil the requirement (3.15), since if two trajectories are identical with respect to space coordinates, but not identical in the sense that the points $\boldsymbol{x}_1(t)$ and $\boldsymbol{x}_2(t)$ are identical for a finite time interval $t \in [0,T]$, where $T = T_1 = T_2$, then the above measure will still give a zero value.

As stated earlier in the report, one possible distance measure would be to measure the difference in the impact points

$$d_5(\boldsymbol{x}_1(\cdot),\boldsymbol{x}_2(\cdot)) = \parallel \boldsymbol{x}_{1,\text{impact}} - \boldsymbol{x}_{2,\text{impact}} \parallel \tag{3.29}$$

The disadvantage with the above measure is that it does not take into account the shape of the trajectories or the time of impact, just the end points of the trajectories. So the above distance measure does not fulfil the requirements (3.15).

A normalized version of the measure (3.29) can be obtained by dividing it by the average of the lengths to impact points one and two from origo

$$d_6(\boldsymbol{x}_1(\cdot),\boldsymbol{x}_2(\cdot)) = \frac{\parallel \boldsymbol{x}_{1,\text{impact}} - \boldsymbol{x}_{2,\text{impact}} \parallel}{\frac{1}{2}(\parallel \boldsymbol{x}_{1,\text{impact}} \parallel + \parallel \boldsymbol{x}_{2,\text{impact}} \parallel)} \tag{3.30}$$

Even though the measures 4-6 are not formally distance measures according to the requirements (3.13)-(3.16), they can be useful in a combined measure. Define the combined distance measure of the the non-normalised distance measures one, three and five

$$d_{c,1}(\boldsymbol{x}_1(\cdot),\boldsymbol{x}_2(\cdot)) = \alpha_1 d_1(\boldsymbol{x}_1(\cdot),\boldsymbol{x}_2(\cdot)) + \alpha_3 d_3(\boldsymbol{x}_1(\cdot),\boldsymbol{x}_2(\cdot)) + \alpha_5 d_5(\boldsymbol{x}_1(\cdot),\boldsymbol{x}_2(\cdot)) \tag{3.31}$$

where

$$\alpha_i > 0 \tag{3.32}$$

Note that the above combined measure fulfil the requirements of a distance measure (3.13)-(3.16).

In a similar way one can define the combined distance measure of the normalised distance measures two, four and six

$$d_{c,2}(\boldsymbol{x}_1(\cdot),\boldsymbol{x}_2(\cdot)) = \alpha_2 d_2(\boldsymbol{x}_1(\cdot),\boldsymbol{x}_2(\cdot)) + \alpha_4 d_4(\boldsymbol{x}_1(\cdot),\boldsymbol{x}_2(\cdot)) + \alpha_6 d_6(\boldsymbol{x}_1(\cdot),\boldsymbol{x}_2(\cdot)) \tag{3.33}$$

where

$$\alpha_i > 0 \tag{3.34}$$

Let $\boldsymbol{x}_1(\cdot,\boldsymbol{p})$ $\boldsymbol{x}_2(\cdot,\boldsymbol{p})$ be missile trajectories corresponding to system model one and two (parameters not suppressed). The criterion function $c(\boldsymbol{p})$ corresponding to the distance measure $d(\boldsymbol{x}_1(\cdot,\boldsymbol{p}),\boldsymbol{x}_2(\cdot,\boldsymbol{p}))$ is simply defined as follows

$$c(\boldsymbol{p}) = d(\boldsymbol{x}_1(\cdot,\boldsymbol{p}),\boldsymbol{x}_2(\cdot,\boldsymbol{p})) \tag{3.35}$$

Note that the distance measure $d(\boldsymbol{x}_1(\cdot,\boldsymbol{p}),\boldsymbol{x}_2(\cdot,\boldsymbol{p}))$ has the same structure as the difference measure $h(\boldsymbol{x}_1(\cdot,\boldsymbol{p}),\boldsymbol{x}_2(\cdot,\boldsymbol{p}))$ in (3.9), we use the letter d here to emphasise that it is a distance measure.

4 Frameworks

The purpose of this chapter is to give an overview of the used framework Frameopt 1.0, the model architecture Merlin, the toolbox Globopt 3.0 and also give an overview of the optimization method of choice, differential evolution.

4.1 The framework Frameopt 1.0, a MATLAB based framework for verification of missile models

The framework Frameopt 1.0 is a MATLAB based framework for verification of a real-time missile model against a reference model. It is the output of the corresponding system models that are compared by using a measure. The framework is a collection of MATLAB scripts and functions as well as the subdirectories BackupData and doc. Note that the models and the optimization code do not belong to the framework, but are linked in. An illustration of how the framework Frameopt 1.0 is related to the optimization toolbox Globopt 3.0 and the models in ACSL [1] and Merlin is given in figure 4.1.

In our design we have chosen to split the interfaces to the models in two parts, where one part belongs to frameopt1.0, and the other to the model directory.

Overall, including framework, models and optimization code, the main directories are as follows

frameopt1.0	A framework for optimization-based verification, including MATLAB scripts and functions.
globopt3.0	A toolbox for global optimization, including the DE solver diffev.m.
missile_matlab1.0	Includes two surface-to-air missile models, based on [29].
merlin	Includes two air-to-air missile models.

In the directory frameopt1.0/doc, guidelines are given on how to include a new missile model and a new measure into the framework etc. Further in frameopt1.0/BackupData, data from every generation are saved, which can be used to restart the computations in case of a computer crash.

In the following, a short overview on how Frameopt 1.0 works is given. It is assumed that the two surface-to-air missile models in the directory missile_matlab1.0 are used, for more detail, see frameopt1.0/doc.

The main script to execute the framework is called `scriptOpt1.m`, where the problem is set up. The script and framework are written to be easily modified. The following are set in the main script and are passed to the objective function

- Interface to model 1, the interface returns a subset of the solution $\boldsymbol{x}_1$
- Interface to model 2, the interface returns a subset of the solution $\boldsymbol{x}_2$
- A measurement function $d(\boldsymbol{x}_1(\cdot, \boldsymbol{p}), \boldsymbol{x}_2(\cdot, \boldsymbol{p}))$ taking the the two solutions $\boldsymbol{x}_1(\cdot, \boldsymbol{p})$ and $\boldsymbol{x}_2(\cdot, \boldsymbol{p})$ as input-data.
- Define different groups of parameters, optimization parameters $\boldsymbol{p}_{\text{opt}}$ and fixed evaluation parameters $\boldsymbol{p}_{\text{fixed}}$.
- Parameters to the differential evolution solver `diffev.m` such as population size and number of iterations.

The framework is described in more detail in the directory `frameopt1.0/doc`.

Many of the ideas used in the framework Frameopt 1.0, are taken from previous projects using optimization, mainly from the projects COFCLUO [30] and

1. Advanced Continuous Simulation Language (ACSL)

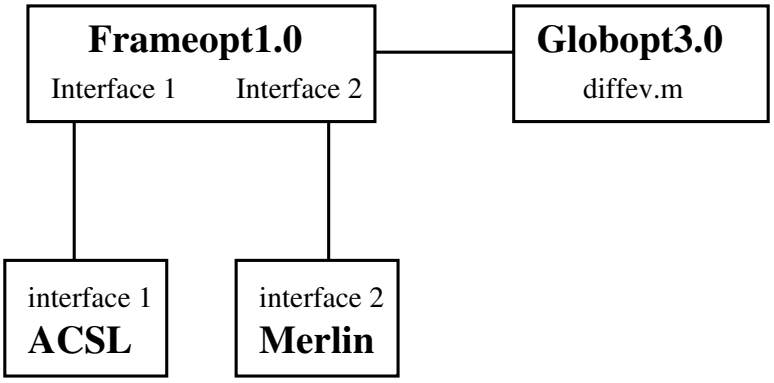


Figure 4.1 – The relation between the framework Frameopt 1.0, the optimization toolbox Globopt 3.0, a model in ACSL and a model in Merlin.

ATTESI [31]. These ideas are further developed in this work. The framework Frameopt 1.0 can easily be modified for other purposes, e.g. aerodynamical shape optimization.

One very important aspect of the DE solver from the Globopt3.0 toolbox is that the objective function evaluations are vectorized, meaning that all objective function evaluations corresponding to one generation can be done in parallel. This has been utilised in previous projects, e.g. ATTESI. Note that Mathworks own toolbox for global optimization is not vectorized.

In Frameopt 1.0, all simulations in one generation corresponding to one interface can be done in parallel, which has not yet been utilised. This is further discussed in frameopt1.0/doc.

4.2 The Globopt 3.0 toolbox

Many real-world optimization problems are non-linear, non-differentiable, and non-convex. Often, the objective function itself has no analytically tractable closed form and can only be calculated using simulation. To solve such problems, one has to rely on black-box optimization methods which make no assumptions of the internal problem structure.

A toolbox for global optimization in a bounded real-valued domain, Globopt 3.0, has been implemented in MATLAB [32]. The aim is to provide an easy to use package of some interesting and well proven global optimization algorithms. Apart from the specialisation to real-valued search spaces, the implementations are not specific to any problem domain. None of the included methods requires calculation of a gradient, only evaluation (sampling) of the objective function.

Support is provided for passing extra input and output variables to the objective function. Variables that do not belong to the search space but are needed for objective function evaluation (e.g. for running a simulation) or useful for later analysis of aspects that lay outside the optimization problem itself. It is also possible to run user defined functions during each iteration, functions that may be useful for calculating, plotting or saving various intermediate result or information.

Three population-based stochastic search algorithms are included: Differential Evolution, Genetic Algorithms and Evolution Strategies.

Differential Evolution is covered in depth in the next section.

The implementation of GA [15, 33] has many internal parameters, as well as several selectable mutation, crossover and selection operators. It is also possible

to "blend" different operators in a single run (to use them in parallel). This generality makes the present GA implementation rather like a "GA-toolbox" in itself, and therefore somewhat difficult to use. On the other hand, there are plenty of possibilities to tune the performance to specific problems.

Evolution Strategies (ES) [19, 15], shares many features with GA, but explicitly employs self-adaption of its control parameters (the operators in GA typically depend on fixed parameters). By including some parameters in an extended search space, subjecting them to mutation and crossover, ES may cope more efficiently with difficult objective functions ("rugged fitness landscapes").

A deterministic global method, *Dividing Rectangles* (DIRECT) [34, 35], is also included. It systematically samples the search space in a clever way, including both global and local explorations simultaneously. It has only one tuning parameter. Unfortunately, the computational complexity scales unfavourably with the number of variables and it is useful only for low-dimensional problems. However, it may be used for a coarse preliminary investigation of the search space.

Finally, also included is a local direct search method, *Pattern Search* (PS) [36, 23] , useful for a final refinement of the result from a global optimizer.

4.3 Differential Evolution

4.3.1 Introduction

The present implementation utilises an algorithm for global optimization in a bounded real-valued domain, called *Differential Evolution* (DE), to search for certain parameter values that maximize a given objective function.

Differential Evolution belongs to a group of optimization methods called *Evolutionary algorithms* (EA), [37, 38]. The most well known member of the EA family is perhaps *Genetic Algorithms* (GA). Evolutionary algorithms is a class of computational techniques that share a common framework and terminology borrowed from biology. These methods are *population based*, i.e. they operate on a whole collection of points in search space. The collection is updated, partially by *random* procedures, into new generations of points that (hopefully) will converge towards an optimal point. The update mechanisms do not involve calculation of derivatives or any other a priori information about the objective function. Unlike a multi-start method however, the points are not evolving in isolation but are interacting with one another during evolution. By doing this in a clever way, one has a greater chance of finding a global optimum than pure Monte Carlo (random walk).

A population of feasible solutions evolves into a series of generations by the action of *crossover*, *mutation* and *selection* operators. On one level, successful individuals produce offspring (crossover) that sometimes undergo small random changes in their traits (mutations). On another level, there is a natural selection going on, "survival of the fittest". The resulting evolution represents both local and global principles for improving already found candidate solutions (*exploitation*) and covering the search space well (*exploring*).

The operators typically depend on some parameters, by which you may, for instance, change the relative balance between the local and global character of the search. Different evolutionary algorithms differ in the implementation of these operators. There are no general rules for which algorithm and parameter values will give the best performance for a given problem.

4.3.2 DE algorithm

Differential evolution (DE) is a relatively recent optimization method that has a particularly simple implementation, with just a small number of control parameters. It is fairly easy to use and quite robust (not overly sensitive to parameter values). DE is often a good starting point for many different kinds of optimization problems. A clear presentation and motivation for DE is given in the original paper [24]. A fuller account, with a lot of implementational details, is given by [17].

The *population* is a sequence $\mathcal{P} = (\boldsymbol{x}_i)_{i=1,\ldots,n}$ of n vectors from a given bounded region of $\mathbb{R}^m$ called the *search space* (simple variable bounds, i.e. "box constraints"). The m coordinates of the vectors are the *decision variables*. After initialization, $\mathcal{P}$ is iteratively evolved into a series of new *generations* by applying simple variation and selection operators. The terms of $\mathcal{P}$ are referred to as "members", or "individuals".

The sequence of the best individual from each generation will hopefully converge to a global optimizer ("best" in terms of objective function value, in this context also called *fitness*). The initial population is usually seeded randomly within the bound constraints, but may also be a "recycled" population from a previous run.

The most important parameter is the size of the population, n. It influences not only the total computation time, but also the quality of convergence. A large population maintains a greater diversity, explores larger areas of search space and may even lead to faster convergence. A too small population may give premature convergence. In [24], a population size of $5m$ to $10m$ is suggested as reasonable for DE. It is important not to be too greedy about n.

The population is updated by running through three steps, *mutation*, *crossover* and *survivor selection*, for each member $\boldsymbol{x}_i$ of $\mathcal{P}$. Thus, for each index $i = 1, \ldots, n$, pick the member $\boldsymbol{x}_i$ and do the following:

Mutation

Randomly choose three different individuals $(\boldsymbol{x}_{i_1}\, \boldsymbol{x}_{i_2}\, \boldsymbol{x}_{i_3})$ from $\mathcal{P}$ other than the current individual $\boldsymbol{x}_i$. Form a temporary vector v as

$$\boldsymbol{v} = \boldsymbol{x}_{i_1} + F \cdot (\boldsymbol{x}_{i_2} - \boldsymbol{x}_{i_3}),$$

where F is a fixed scale factor (an option parameter) $\in (0, 2)$.

Crossover

From the current vector $\boldsymbol{x}_i$ and the temporary $\boldsymbol{v}$, a *trail vector* $\boldsymbol{u}$ is formed component-wise, by selecting with probability $\boldsymbol{p}_C$ the coordinate from $\boldsymbol{v}$, otherwise from $\boldsymbol{x}_i$ (this is repeated independently for each coordinate of $\boldsymbol{u}$). Here, p_C is another option parameter $\in (0, 1]$.

Survivor selection

The trail vector $\boldsymbol{u}$ is then compared with the current vector $\boldsymbol{x}_i$, and if $\boldsymbol{u}$ has a better fitness value, it takes the place of $\boldsymbol{x}_i$ in the next generation, otherwise $\boldsymbol{x}_i$ is retained.

Let us just point out some differences between DE and a genetic algorithm, [37, 33]. In DE, there is no "parent selection" based on fitness values in the mutation/crossover steps. There is no need for collecting and sorting a whole sequence of offspring, instead the survivor selection takes place by a simple direct tournament between the current vector and the corresponding trail vector.

A mutation operator in GA usually perturbs the coordinates of a single individual according to some fixed probability distribution. In DE, on the other hand, mutation is determined by the population itself through the linear combination of the randomly chosen vectors ($\boldsymbol{x}_{i_1}\, \boldsymbol{x}_{i_2}\, \boldsymbol{x}_{i_3}$). This gives a "self-adapting" character which seems to be a very beneficial feature of DE.

4.4 Merlin

Merlin [39, 40] is an architecture for development of real-time simulation models, based on C++ computer language and Extensible Markup Language (XML), designed to run on all major operating systems. It is currently used in several simulators and applications such as at FLSC [2] and MSS [3]/MT [4] where Merlin is used to simulate missiles. It may be interesting to note that Merlin is not an application by itself. Rather, the Merlin architecture allows for various applications to integrate many different models that follow the interfaces defined in Merlin.

Models in Merlin are built by composition, using various components in a "model object composition" file which defines the parameters that constitutes a complete missile model. [5]

In this work, a simple test-bed application has been used that sets up a missile model according to the Merlin standard and manages a simple target model. The application is started via systems calls from the MATLAB framework. Data is exported to output files from the simulation by recording the state of the missile and the target during the stepping of the model time. These data files are easily parsed in the Merlin plugin to the MATLAB framework.

2. Swedish Air Force Combat Simulation Centre
3. Mission Support System
4. Mission Trainer
5. stored in XML

5 Results

There is no guarantee that differential evolution converges towards a global optimum, nevertheless there is a need of convergence measures that can be used as guide when to stop the optimisation.

Based on computing the mean value of the objective function values and the best objective function value of the last population, we will define relative convergence for a maximisation problem as follows

$$\text{relative convergence} = \frac{\max_{\boldsymbol{x}\in\text{last pop.}} f(\boldsymbol{x}) - \text{mean}_{\boldsymbol{x}\in\text{last pop.}}\ f(\boldsymbol{x})}{\max_{\boldsymbol{x}\in\text{last pop.}}\ f(\boldsymbol{x})} \tag{5.1}$$

5.1 Case study 1: surface-to-air missile

In this case study, two surface-to-air missile models are compared. The missile models are implemented in MATLAB, and are based on the surface-to-air missile model described in [29]. The two missile models, model 1 and model 2, differ in C_{D_0} and C_{D_α} by a multiplicative factor, 1.1, in the following way

$$\begin{aligned} C_{D_0}^{\text{missile 2}} &= 1.1 \times C_{D_0}^{\text{missile 1}} \\ C_{D_\alpha}^{\text{missile 2}} &= 1.1 \times C_{D_\alpha}^{\text{missile 1}} \end{aligned} \tag{5.2}$$

As discussed in the report in previous chapters, it is the system models that are compared, by comparing the output. In this case the trajectories are compared by several different measures in separate optimization runs.

The computation of the end point of the missile trajectory is done as follows. First the minimum passing distance of the missile target duel is computed. The end point of the line that defines the minimum passing distance on the missile trajectory is defined as the end point of the missile trajectory.

Throughout the tests, we assume a flat earth, uses a North East Down (NED) reference frame. Further we let the surface-to-air missile start in the origin ($x = 0, y = 0, z = 0$), and the target flies on a straight line from north to south on a constant altitude $z_t(0)$. As the optimization parameters we uses the starting point for the target system $x_t(0), y_t(0), z_t(0)$, the optimization parameter space consists of the cube defined by $x_t(0), y_t(0)[\in 4.0 \times 10^3,\ 8.0 \times 10^3]$, $z_t(0) \in [-8.0 \times 10^3,\ -4.0 \times 10^3]$.

First, test each of the six measures $d_1, \ldots, d_6$ defined in chapter 3 in six different optimization runs, the objective function $c_i(\boldsymbol{p})$ corresponding to measure i is defined as

$$c_i(\boldsymbol{p}) = d_i(\boldsymbol{x}_1(\cdot, \boldsymbol{p}), \boldsymbol{x}_2(\cdot, \boldsymbol{p})) \tag{5.3}$$

where $\boldsymbol{x}_1(\cdot, \boldsymbol{p})$ and $\boldsymbol{x}_2(\cdot, \boldsymbol{p})$ are the solutions to the two ODEs given in (3.3) describing the two system models. Further, as described just above the optimization parameters $\boldsymbol{p}_{\text{opt}}$ are given by

$$\boldsymbol{p}_{\text{opt}} = [x_t(0), y_t(0), z_t(0)]^T \tag{5.4}$$

that is the starting point for the target system.

The measures d_7 and d_8 are composite measures, and more precisely the measure d_7 is defined as

$$d_7(\boldsymbol{x}_1(\cdot, \boldsymbol{p}), \boldsymbol{x}_2(\cdot, \boldsymbol{p})) = \alpha_1 d_1(\boldsymbol{x}_1(\cdot, \boldsymbol{p}), \boldsymbol{x}_2(\cdot, \boldsymbol{p})) + \alpha_5 d_5(\boldsymbol{x}_1(\cdot, \boldsymbol{p}), \boldsymbol{x}_2(\cdot, \boldsymbol{p})) \tag{5.5}$$

and the measure d_8 is defined as

$$d_8(\boldsymbol{x}_1(\cdot,\boldsymbol{p}),\boldsymbol{x}_2(\cdot,\boldsymbol{p})) = \beta_2 d_2(\boldsymbol{x}_1(\cdot,\boldsymbol{p}),\boldsymbol{x}_2(\cdot,\boldsymbol{p})) + \beta_6 d_6(\boldsymbol{x}_1(\cdot,\boldsymbol{p}),\boldsymbol{x}_2(\cdot,\boldsymbol{p})) \quad (5.6)$$

In general it is not a trivial task to choose the coefficients α_1, α_5 and β_2 and β_6. Let $\boldsymbol{p}^\star_{i,\text{opt}}$ be the optimizer to the optimization problem with $c_i(\boldsymbol{p}_{\text{opt}})$ as the objective function. We have chosen the coefficients α_1, α_5 and β_2 and β_6. to be

$$\alpha_1 = \frac{1}{c_1(\boldsymbol{p}^\star_{1,\text{opt}})}, \quad \alpha_5 = \frac{2}{c_5(\boldsymbol{p}^\star_{5,\text{opt}})} \quad (5.7)$$

$$\beta_2 = \frac{1}{c_2(\boldsymbol{p}^\star_{2,\text{opt}})}, \quad \beta_6 = \frac{2}{c_6(\boldsymbol{p}^\star_{6,\text{opt}})} \quad (5.8)$$

Note that if the optimizers of the objective functions one and five are equal $\boldsymbol{p}^\star_{1,\text{opt}} = \boldsymbol{p}^\star_{5,\text{opt}}$ and the optimizers of two and six are equal $\boldsymbol{p}^\star_{2,\text{opt}} = \boldsymbol{p}^\star_{6,\text{opt}}$ then the distance functions d_7 and d_8 at the corresponding points are given by

$$d_7(\boldsymbol{x}_1(\cdot,\boldsymbol{p}^\star_{1,\text{opt}}),\boldsymbol{x}_2(\cdot,\boldsymbol{p}^\star_{1,\text{opt}})) = 3 \;\; d_8(\boldsymbol{x}_1(\cdot,\boldsymbol{p}^\star_{2,\text{opt}}),\boldsymbol{x}_2(\cdot,\boldsymbol{p}^\star_{2,\text{opt}})) = 3 \quad (5.9)$$

In order to get the best decision basis for the choice of the parameters α_1, α_5 and β_2,β_6 one needs to solve the corresponding multiobjective optimization problems, which is just what one tries to avoid by the current strategy, i.e. a "catch 22". However, the goal with weighting basic measures are to get a new composite measure that catches the most important aspects of measuring the difference between two trajectories. For example, measure d_5 measures just the difference between the end points, and not the shape of the trajectories. While measure d_1 measures the difference between two trajectories, but does not catch the difference between the end points very well. The combined measure d_7 is in many ways a better measure since it avoids the weakness of measures d_1 and d_5. In order to make it work, it is important to make reasonable good choices of the parameters α_1 and α_5, a crude approximation of the optimum will serve just fine. The factor 2 in α_5 and the factor 1 in α_1 means that one puts more emphasis towards the measure d_5 than the measure d_1 in the combined measure d_7.

For each of the optimization runs we use differential evolution from the Globopt 3.0 package with a population of size 30.

We will start to describe the optimization test for measure d_5 and corresponding objective function c_5. The measure d_5 measures the distance between the impact points, this is illustrated in figure 5.1. In figure 5.2, objective function values are plotted versus objective function evaluations, in each iteration the objective function is evaluated a total of 30 times. In figure 5.3, for each generation, the maximum objective function value (dot) so far and the mean value (circle) are plotted versus the number of iterations (generations). In figure 5.4, the relative convergence is plotted versus the iterations. The relative convergence reaches below the value 10^{-2} for the first time at iteration 39.

The optimization run took a total of 80 iterations and 2400 objective function evaluations. In the final population the highest objective function value is $f(\boldsymbol{p}_{\text{opt}}) = 9.8054 \times 10^1$ and the corresponding parameter vector is $x_t(0) = 7.9988e \times 10^3$, $y_t(0) = 7.9990 \times 10^3$, $z_t(0) = -7.9865 \times 10^3$. Note that the population converges towards the corner point $x_t(0) = 8.0 \times 10^3$, $y_t(0) = 8.0 \times 10^3$, $z_t(0) = -8.0 \times 10^3$, as the optimizer, as expected.

In table 5.1 a summary of the optimization runs are given. For objective functions c_1, c_2, c_3, c_5, c_7 and c_8 the populations converge towards the corner point $8.0\times10^3, 8.0\times10^3, -8.0\times10^3$ as the optimizer. For the objective function

	$x_t(0)$	$y_t(0)$	$z_t(0)$	$f(\boldsymbol{p}_{\text{opt}})$	it r.c.	total it.
d_1	7.9993×10^3	7.9998×10^3	-7.9874×10^3	1.8680×10^6	45	80
d_2	7.9995×10^3	8.0000×10^3	-7.9877×10^3	1.5689×10^2	32	80
d_3	7.9978×10^3	7.9998×10^3	-7.9870×10^3	4.2431×10^5	42	160
d_4	4.0064×10^3	7.9997×10^3	-7.9744×10^3	3.5535×10^1	49	160
d_5	7.9988×10^3	7.9990×10^3	-7.9865×10^3	9.8054×10^1	39	80
d_6	4.0035×10^3	7.9937×10^3	-4.0181×10^3	9.9165×10^{-3}	35	80
d_7	7.9929×10^3	7.9985×10^3	-7.9848×10^3	2.9993×10^0	29	80
d_8	4.0026×10^3	7.9999×10^3	-4.0206×10^3	2.7224×10^0	31	80

Table 5.1 – Results from A. Sandbloms model. In the table, it r.c., stands for the first iteration number when the relative convergence is below 10^{-2}.

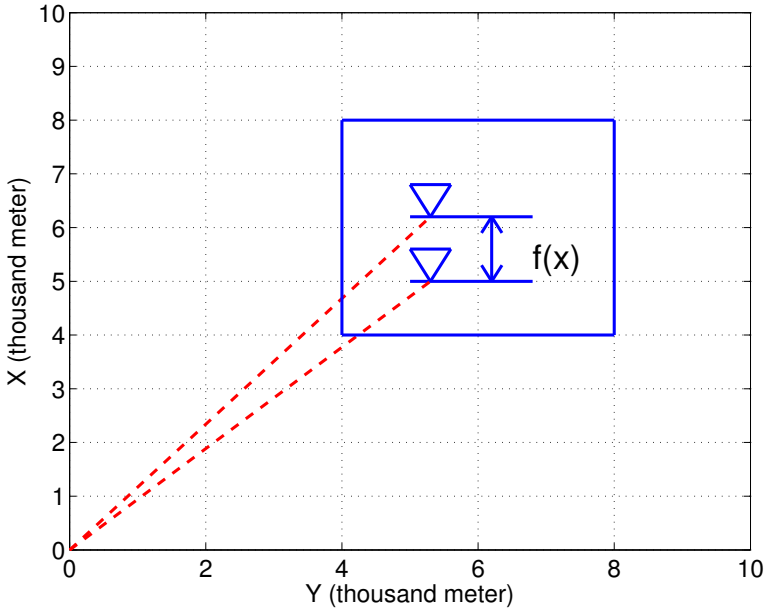


Figure 5.1 – Difference of the two missile target duels. The red dotted lines correspond to the trajectories with origo as starting point. The triangles correspond to the target aircraft. The function value $f(x)$ corresponds to the difference of impact points for the two missile target duels.

c_4, the population converges towards the corner point $4.0 \times 10^3, 8.0 \times 10^3, -8 \times 10^3$, and for the objective function c_6 towards $4.0 \times 10^3, 8.0 \times 10^3, -4.0 \times 10^3$.

For the composed measures, d_7 (5.5) the corresponding population converges towards the corner point $8.0 \times 10^3, 8.0 \times 10^3, -8.0 \times 10^3$ as expected since the populations corresponding to d_1 and d_5 converge towards $8.0 \times 10^3, 8.0 \times 10^3, -8.0 \times 10^3$. The population corresponding to the measure d_8 (5.6) converges towards $4.0 \times 10^3, 8.0 \times 10^3, -4 \times 10^3$, note that the population corresponding to the measure d_2 converges towards $8.0 \times 10^3, 8.0 \times 10^3, -8.0 \times 10^3$ and d_6 towards $4.0 \times 10^3, 8.0 \times 10^3, -4.0 \times 10^3$.

5.2 Case study 2: air-to-air missile

In this case study, two air-to-air missile models of the same physical missile are compared. The two missile models differ in that model 1 uses a non ideal target seeker and model 2 uses an ideal target seeker.

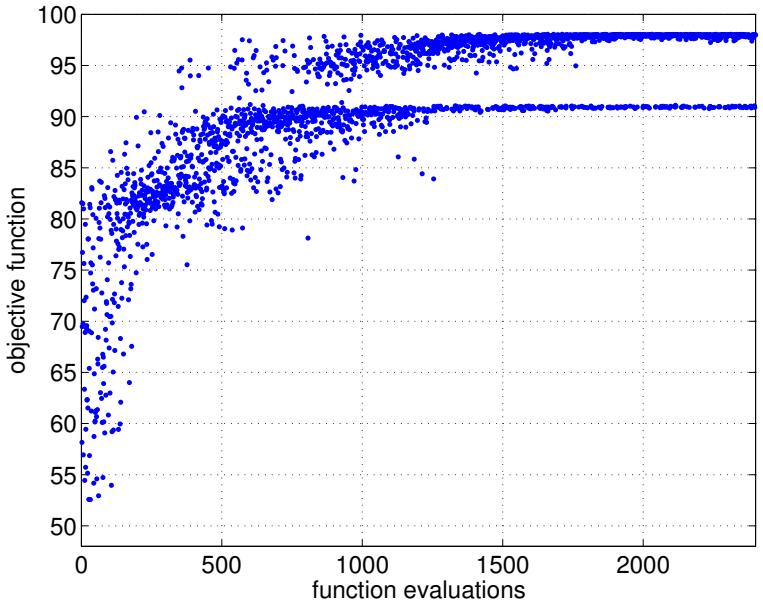


Figure 5.2 – Convergence with respect to objective function evaluations for case study 1 and measure d_5. Objective function values are plotted versus objective function evaluations. Each (dot) corresponds to an objective function evaluation.

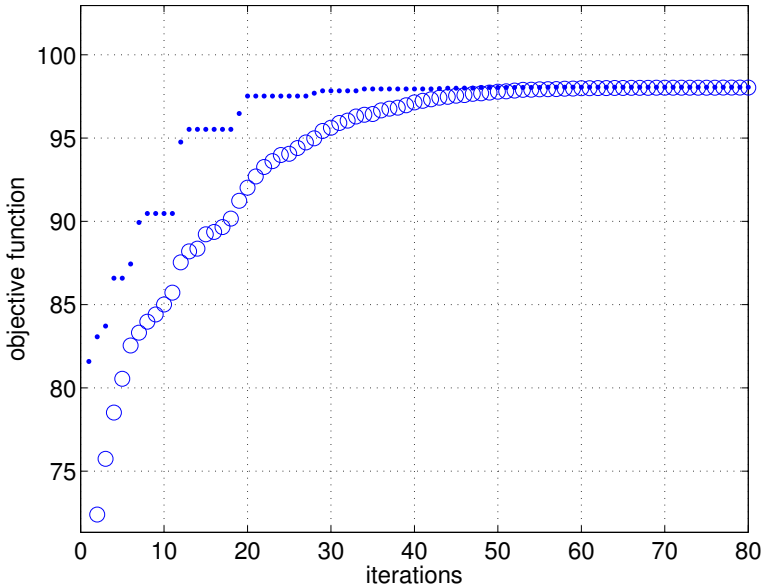


Figure 5.3 – Convergence with respect to max and mean values by generation for case study 1 and measure d_5. The maximum objective function value of each generation (dot) and mean value (circle) are plotted versus the number of iterations (generations).

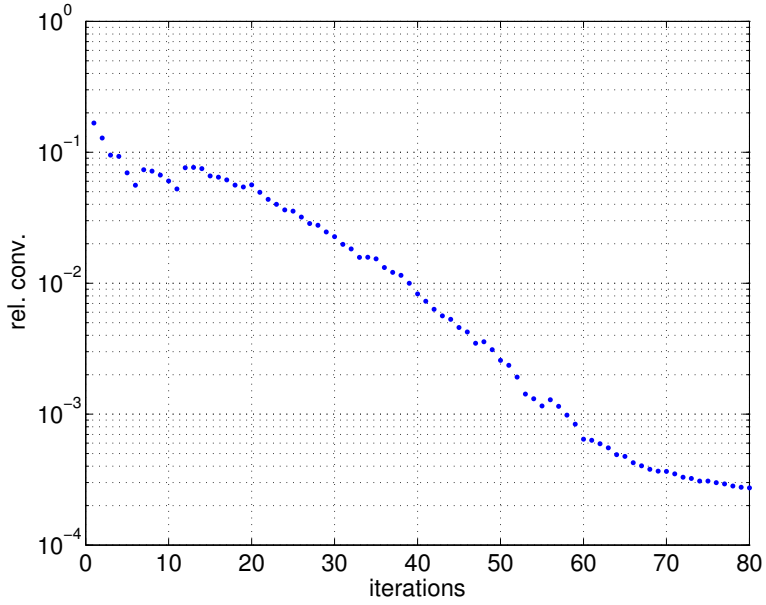


Figure 5.4 – The relative convergence for case study 1 and measure d_5

Both missile models are implemented in the framework Merlin. The computation of the end point of the missile trajectory is done as follows. First the minimum passing distance of the missile target duel is computed by using interpolation or extrapolation of the trajectory obtained from Merlin. The end point of the line that defines the minimum passing distance on the missile trajectory is defined as the end point of the missile trajectory.

Throughout the tests, we assume a flat earth, uses a North East Down (NED) coordinate system. Further we let the air-to-air missile starts in the origin ($x = 0, y = 0, z = 0$), and the target flies on a straight line from north to south on a constant altitude $z_t(0) = 10000m$. As the optimization parameters we use the starting point for the target system $x_t(0), y_t(0)$, the optimization parameter space consists of the rectangle defined by $x_t(0) \in [2.0\times10^3,\ 5.0\times10^3]$ and $y_t(0) \in [-2.5 \times 10^3,\ 2.5 \times 10^3]$.

In the optimization test we will use the measure d_5, which measures the distance between the impact points. The population based stochastic global optimization method differential evolution from the Globopt 3.0 package is used. The population size is 40 and the total number of generations is 800.

In figure 5.5 objective function values are plotted versus objective function evaluations. In figure 5.6, for each generation, the maximum objective function value (dot) so far and the mean value (circle) are plotted versus the number of iterations (generations). The optimization run took a total of 800 iterations and 32000 objective function evaluations. In the final population the highest objective function value is $f(\boldsymbol{p}^\star_{\text{opt}}) = 4.0489 \times 10^3$ and the corresponding parameter vector is $\boldsymbol{p}^\star_{\text{opt}} = [x_t(0) = 3.2820 \times 10^3,\ \ y_t(0) = 1.9691 \times 10^3]^T$.

Around objective function evaluation number 12598 a step in objective function value occurs, up to objective function evaluation number 12598 the highest objective function value is 1.6317 and occur at no 11654. The objective function value at no 12599 is 4.0467×10^3. Convergence plots corresponding to figures 5.5 and 5.6, but with only objective function values before the large "step" at no 12598 are given in 5.7 and 5.8

The trajectories of model 1 and model 2 are given in figures 5.9 and 5.10.

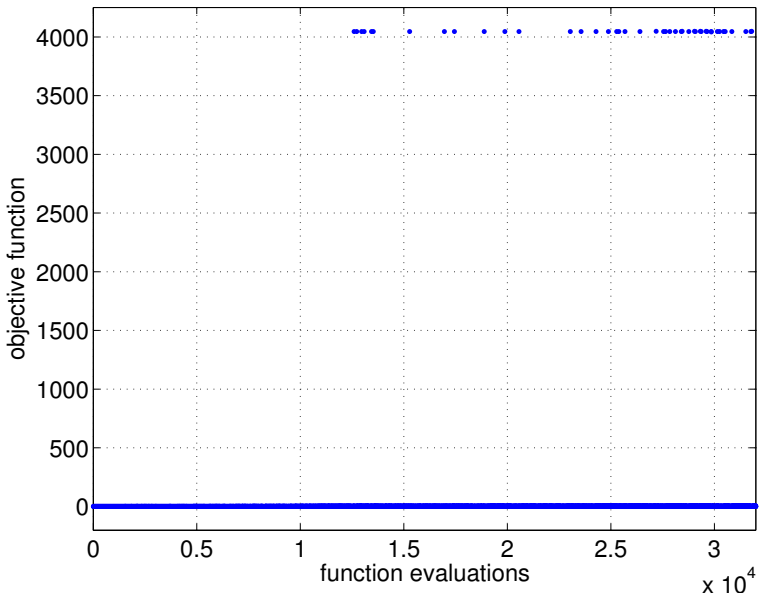


Figure 5.5 – Convergence with respect to objective function evaluations for case study 2 and measure d_5. Objective function values are plotted versus objective function evaluations. Each (dot) corresponds to an objective function evaluation.

For both models, the dashed-line is the target going on a straight line from north to south starting at the optimizer $\boldsymbol{p}^{\star}_{\text{opt}} = [x_t(0) = 3.2820 \times 10^3,\ \ y_t(0) = 1.9691 \times 10^3]^T$ (note $z_t(0) = -1.0 \times 10^4$, not an optimization parameter). For model 1, the missile hits the target at position $x = 1.7034 \times 10^3$, $y = 1.9635 \times 10^3$, $z = -9.9979 \times 10^3$ with a minimum passing distance to target of 15.241m. For model 2, the missile first "miss" the target near missile position of $x = 1.7031 \times 10^3$ $y = 1.9650 \times 10^3$ $z = -9.9979 \times 10^3$ with a minimum passing distance of 15.086m. Next the missile turns and hits the target at position $x = -2.3454 \times 10^3$ $y = 1.9558 \times 10^3$ $z = -9.9915 \times 10^3$ with a minimum passing distance of 24.8342m. From the figures one can see that the maximum difference between the impact points for model 1 and model 2 is quite large, it is 4.0489×10^3, the optimal value.

One possible explanation of the above behaviour is that the termination condition of the simulation of missile 2 does not work properly. The simulation should probably stop near the missile position $x = 1.7031 \times 10^3$ $y = 1.9650 \times 10^3$ $z = -9.9979 \times 10^3$. So if this is the case, the cause can be found in the Merlin framework and not with the missile model itself.

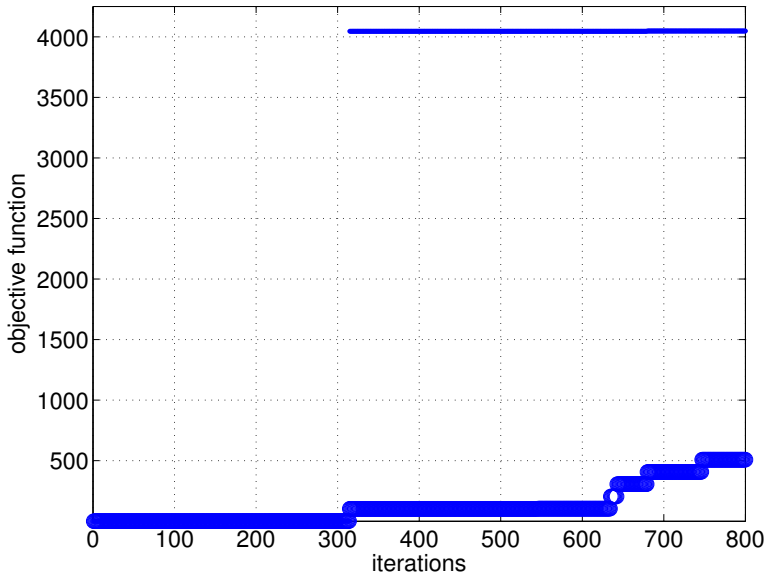


Figure 5.6 – Convergence with respect to max and mean values by generation for case study 2 and measure d_5. The maximum objective function value of each generation (dot) and mean values (circle) are plotted versus the number of iterations (generations).

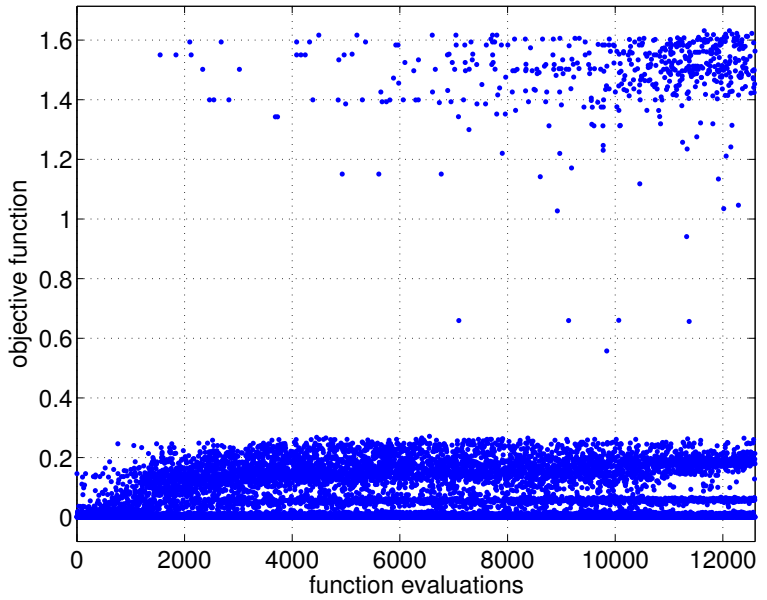


Figure 5.7 – Convergence with respect to objective function evaluations for case study 2 and measure d_5. Objective function values are plotted versus objective function evaluations. Each (dot) corresponds to an objective function evaluation.

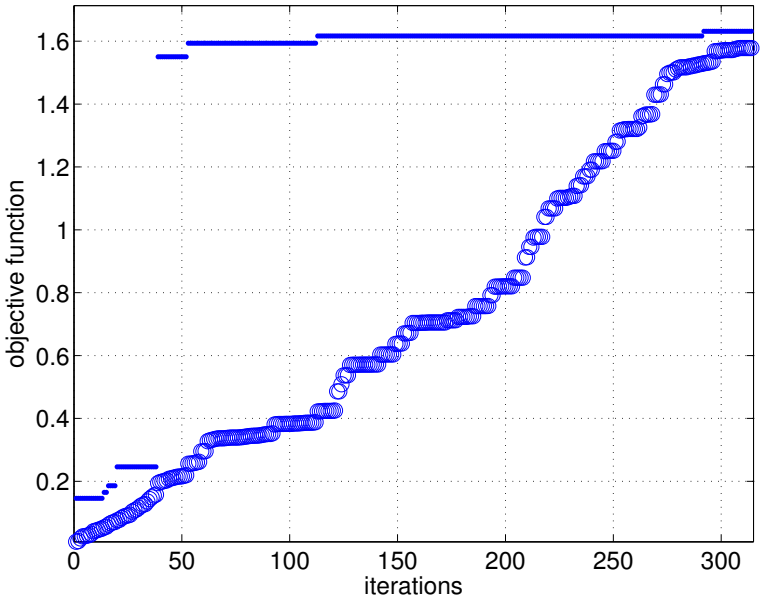


Figure 5.8 – Convergence with respect to max and mean values by generation for case study 2 and measure d_5. The maximum objective function value of each generation (dot) and mean value (circle) are plotted versus the number of iterations (generations).

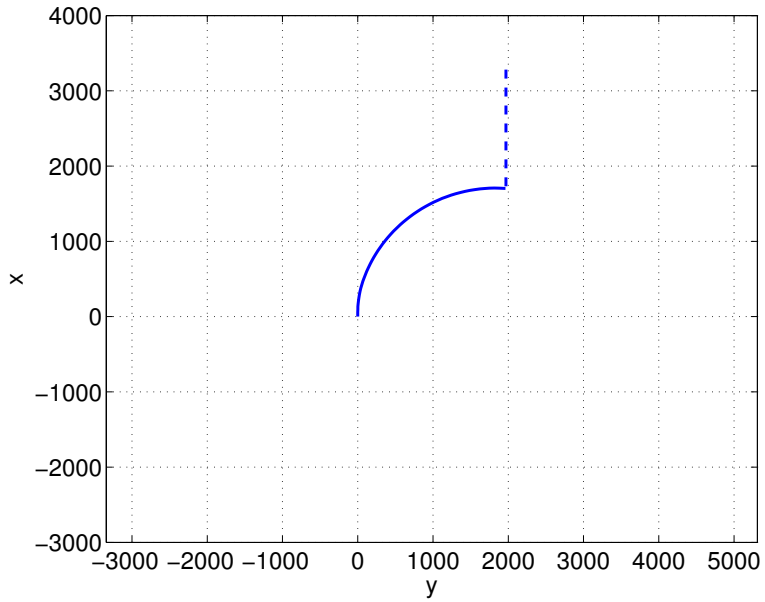


Figure 5.9 – Trajectories for model 1. The blue solid line corresponds to the missile trajectory with origo as starting point. The blue dotted line corresponds to the target trajectory.

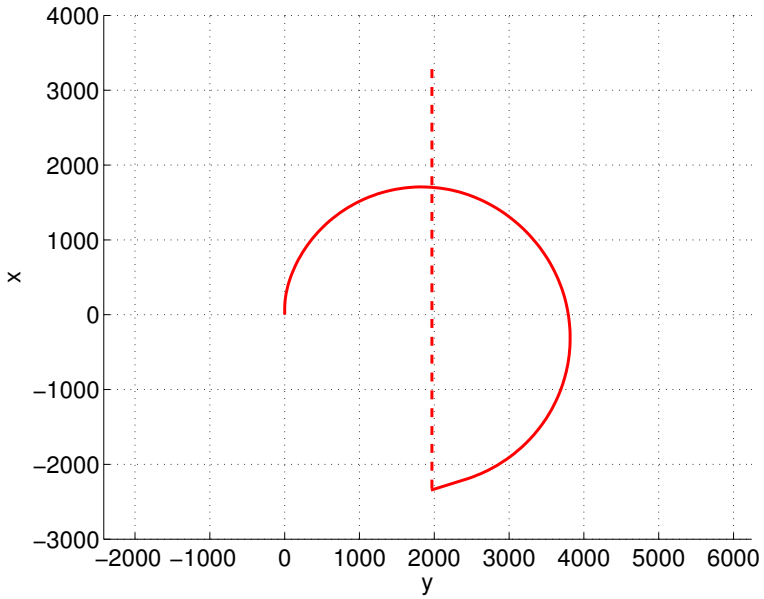


Figure 5.10 – Trajectories for model 2. The red solid line corresponds to the missile trajectory with origo as starting point. The red dotted line corresponds to the target trajectory.

6 Conclusion and future Work

6.1 Conclusion

It has been shown by the computer tests that it is possible to find large differences in model trajectories, if they exist [1], by the discussed optimization based verification method. These large differences occur very rarely in the tested models and would probably be very hard to find by other methods, e.g. Monte Carlo simulations. However, more computer tests, on several models are needed in order to analyse the proposed method strengths and weaknesses.

6.2 Future work

Possible future work includes the following

- More computer tests of the verification method presented in this report, see 6.1.
- Use parallel computation in order to speed up the computations, see 4.1.
- Further development of measures. The discussion of measures between trajectories given in this report is just a staring point.
- Develop new features in the Globopt 3.0 toolbox e.g.
  - Be able to handle real valued and integer valued parameter domain. This will be very helpful in order to analyse problems that have different configurations.
  - Implement a population based multiobjective optimization algorithm. As discussed in the report, it is natural to view the verification method as a multiobjective optimization problem when several measures are used.
  - Implement a more general parameter domain than simple bound constraints.

1. Note, there is no guarantee that the worst case can be found.

A Derivations

A.1 Optimization problem, detailed version

This is a detailed version of section 2.3

We will here describe in detail the steps needed in order to transform the system model (2.7) into an equivalent system model more suitable for optimization, and also how to state the corresponding optimization problem. Note, the real system is the same (it is the view that is changed).

The first step consists of tuning the internal system model parameters to adequate values. They will remain constant throughout the opimization, and possibly remain constant throughout several optimization runs.

We will redefine the state evolution function $\boldsymbol{f}$ and the function $\boldsymbol{g}$

$$\boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{p}_{\text{eval}}) := \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{p}_{\text{eval}}, \boldsymbol{q}), \quad \boldsymbol{g}(\boldsymbol{p}_{\text{eval}}) := \boldsymbol{g}(\boldsymbol{p}_{\text{eval}}, \boldsymbol{q}) \tag{A.1}$$

for a given fixed parameter vector $\boldsymbol{q}$. The system model (2.7) can now be written as

$$\begin{aligned} &\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{p}_{\text{eval}}), \quad \boldsymbol{x}(t=0) = \boldsymbol{g}(\boldsymbol{p}_{\text{eval}}) \\ &\boldsymbol{f}, \boldsymbol{x}, \boldsymbol{g} \in \mathcal{R}^{n\times 1}, \quad \boldsymbol{p}_{\text{eval}} \in \mathcal{R}^{m\times 1}, \quad t \in \mathcal{R}, \quad t \geq 0 \end{aligned} \tag{A.2}$$

The next step is to pick out the optimization parameters $\boldsymbol{p}_{\text{opt}}$ among the evaluation parameters $\boldsymbol{p}_{\text{eval}}$. On a practical side, this is quite easily managed with the framework Frameopt 1.0, see section 4.1. The evaluation parameters $\boldsymbol{p}_{\text{eval}}$ can, as previously stated in subsection 2.2.2, be divided into optimization parameters $\boldsymbol{p}_{\text{opt}}$ and fixed evaluation parameters $\boldsymbol{p}_{\text{fixed}}$. As the name indicates, fixed evaluation parameters are fixed during an optimization run.

Before the optimization run, the fixed evaluation parameters $\boldsymbol{p}_{\text{fixed}}$ are tuned to adequate values. This is similar to tuning the internal system model parameters $\boldsymbol{q}$. But, note on the practical side, this is easily managed by the framework Frameopt 1.0. The operator does not need to know anything about the implementation of the system model.

Now the system model (A.2) is redefined into

$$\begin{aligned} &\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{p}_{\text{opt}}), \quad \boldsymbol{x}(t=0) = \boldsymbol{g}(\boldsymbol{p}_{\text{opt}}) \\ &\boldsymbol{f}, \boldsymbol{x}, \in \mathcal{R}^{n\times 1}, \quad \boldsymbol{p}_{\text{opt}} \in \mathcal{R}^{m\times 1}, \quad t \in \mathcal{R}, \quad t \geq 0 \end{aligned} \tag{A.3}$$

where the system evaluation function $\boldsymbol{f}$ has been redefined as

$$\boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{p}_{\text{opt}}) := \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{p}_{\text{eval}}) \tag{A.4}$$

for a given value of the fixed evaluation parameters $\boldsymbol{p}_{\text{fixed}}$. The function $\boldsymbol{g}$ has been redefined in a similar way.

The next step is to formulate an objective function $c(\boldsymbol{p}_{\text{opt}})$. This is done through defining a measure $h(\boldsymbol{x}(\cdot, \boldsymbol{p}_{\text{opt}}))$ on the solution $\boldsymbol{x}(\cdot, \boldsymbol{p}_{\text{opt}})$ to the IVP (2.8). The objective function is thus

$$c(\boldsymbol{p}_{\text{opt}}) = h(\boldsymbol{x}(\cdot, \boldsymbol{p}_{\text{opt}})) \tag{A.5}$$

For example, the measure $h(\boldsymbol{x}(\cdot, \boldsymbol{p}_{\text{opt}}))$ could be the maximum angle of attack, and in this case the objective function [1] is

$$c(\boldsymbol{p}_{\text{opt}}) := \max_{t\in[0,T]} \alpha(t, \boldsymbol{p}_{\text{opt}}) \tag{A.6}$$

1. In this case, a criterion function is used as an objective function

The final step before optimization is to select the parameter set $\mathcal{P}_{\text{opt}}$ over which the objective function is optimized. For all tests in this report, we will use simple bound constraints, that is each parameter $\boldsymbol{p}_{\text{opt}}(i)$ is bounded above and below

$$\boldsymbol{p}_{\text{opt}}^{\min}(i) \leq \boldsymbol{p}_{\text{opt}}(i) \leq \boldsymbol{p}_{\text{opt}}^{\max}(i) \tag{A.7}$$

Finally, the optimization problem can be stated as

$$\max_{\boldsymbol{p}_{\text{opt}} \in \mathcal{P}} c(\boldsymbol{p}_{\text{opt}}) \tag{A.8}$$

A.2 Problem description

This is a detailed version of section 3.2

We will here describe in detail the steps needed to be done just prior to formulate an optimization problem.

Consider the two system models

$$\begin{aligned} \dot{\boldsymbol{x}}_1 &= \boldsymbol{f}_1(t, \boldsymbol{x}_1, \boldsymbol{p}_{\text{eval}}, \boldsymbol{q}_1), \quad \boldsymbol{x}_1(t=0) = \boldsymbol{g}_1(\boldsymbol{p}_{\text{eval}}, \boldsymbol{q}_1) \quad \text{Model 1} \\ \dot{\boldsymbol{x}}_2 &= \boldsymbol{f}_2(t, \boldsymbol{x}_2, \boldsymbol{p}_{\text{eval}}, \boldsymbol{q}_2), \quad \boldsymbol{x}_2(t=0) = \boldsymbol{g}_2(\boldsymbol{p}_{\text{eval}}, \boldsymbol{q}_2) \quad \text{Model 2} \end{aligned} \tag{A.9}$$

where

$$\begin{aligned} &\boldsymbol{f}_1, \boldsymbol{x}_1, \boldsymbol{g}_1 \in \mathcal{R}^{n_1 \times 1}, \quad \boldsymbol{f}_2, \boldsymbol{x}_2, \boldsymbol{g}_2 \in \mathcal{R}^{n_2 \times 1}, \quad \boldsymbol{p}_{\text{eval}} \in \mathcal{R}^{m \times 1} \\ &\boldsymbol{q}_1 \in \mathcal{R}^{l_1 \times 1}, \quad \boldsymbol{q}_2 \in \mathcal{R}^{l_2 \times 1}, \quad t \in \mathcal{R}, \quad t \geq 0 \end{aligned} \tag{A.10}$$

The parameter-vector $\boldsymbol{p}_{\text{eval}}$ consists of evaluation parameters, they are the same for the two models. The parameter vectors $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ consist of internal model parameters of the system models one and two respectively.

The first step consists of tuning the internal parameters of the two models $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$, such that the essential output of the two models are as equal as possible, e.g. the trajectories of the two models should be as similar as possible.

Next we redefine the state evolution functions

$$\boldsymbol{f}_1(t, \boldsymbol{x}_1, \boldsymbol{p}_{\text{eval}}) := \boldsymbol{f}_1(t, \boldsymbol{x}_1, \boldsymbol{p}_{\text{eval}}, \boldsymbol{q}_1), \quad \boldsymbol{f}_2(t, \boldsymbol{x}_2, \boldsymbol{p}_{\text{eval}}) := \boldsymbol{f}_2(t, \boldsymbol{x}_1, \boldsymbol{p}_{\text{eval}}, \boldsymbol{q}_1) \tag{A.11}$$

for given fixed parameter vectors $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ respectively. In a similar way the functions $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$ are redefined. The two system models (3.1) can now be transformed into

$$\begin{aligned} \dot{\boldsymbol{x}}_1 &= \boldsymbol{f}_1(t, \boldsymbol{x}_1, \boldsymbol{p}_{\text{eval}}), \quad \boldsymbol{x}_1(t=0) = \boldsymbol{g}_1(\boldsymbol{p}_{\text{eval}}) \quad \text{Model 1} \\ \dot{\boldsymbol{x}}_2 &= \boldsymbol{f}_2(t, \boldsymbol{x}_2, \boldsymbol{p}_{\text{eval}}), \quad \boldsymbol{x}_2(t=0) = \boldsymbol{g}_2(\boldsymbol{p}_{\text{eval}}) \quad \text{Model 2} \end{aligned} \tag{A.12}$$

where

$$\boldsymbol{f}_1, \boldsymbol{x}_1, \boldsymbol{g}_1 \in \mathcal{R}^{n_1 \times 1}, \quad \boldsymbol{f}_2, \boldsymbol{x}_2, \boldsymbol{g}_2 \in \mathcal{R}^{n_2 \times 1}, \quad \boldsymbol{p} \in \mathcal{R}^{m \times 1}, \quad t \in \mathcal{R}, \quad t \geq 0 \tag{A.13}$$

The next step is to pick out the optimization parameters among the evaluation parameters, this is done just as in section A.1. The two system models are now redefined into

$$\begin{aligned} \dot{\boldsymbol{x}}_1 &= \boldsymbol{f}_1(t, \boldsymbol{x}_1, \boldsymbol{p}_{\text{opt}}), \quad \boldsymbol{x}_1(t=0) = \boldsymbol{g}_1(\boldsymbol{p}_{\text{opt}}) \quad \text{Model 1} \\ \dot{\boldsymbol{x}}_2 &= \boldsymbol{f}_2(t, \boldsymbol{x}_2, \boldsymbol{p}_{\text{opt}}), \quad \boldsymbol{x}_2(t=0) = \boldsymbol{g}_2(\boldsymbol{p}_{\text{opt}}) \quad \text{Model 2} \end{aligned} \tag{A.14}$$

where

$$\boldsymbol{f}_1, \boldsymbol{x}_1, \boldsymbol{g}_1 \in \mathcal{R}^{n_1 \times 1}, \quad \boldsymbol{f}_2, \boldsymbol{x}_2, \boldsymbol{g}_2 \in \mathcal{R}^{n_2 \times 1}, \quad \boldsymbol{p}_{\text{opt}} \in \mathcal{R}^{m \times 1}, \quad t \in \mathcal{R}, \quad t \geq 0 \tag{A.15}$$

Note that it is only the optimization parameters $\boldsymbol{p}_{\text{opt}}$ that are visible in equation (A.14), all other parameters are suppressed, but they still exist. The system evaluation functions $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ have been redefined according to

$$\begin{aligned} \boldsymbol{f}_1(t, \boldsymbol{x}_1, \boldsymbol{p}_{\text{opt}}) &:= \boldsymbol{f}_1(t, \boldsymbol{x}_1, \boldsymbol{p}_{\text{eval}}) \\ \boldsymbol{f}_2(t, \boldsymbol{x}_2, \boldsymbol{p}_{\text{opt}}) &:= \boldsymbol{f}_2(t, \boldsymbol{x}_2, \boldsymbol{p}_{\text{eval}}) \end{aligned} \tag{A.16}$$

Bibliography

[1] The Mathworks, Inc., Natick, Massachusetts. *MATLAB /Simulink*. `http://www.mathworks.com`.

[2] C. Fielding, A. Varga, S. Bennani, and M. Selier, editors. *Advanced techniques for clearance of flight control laws*. Number 283 in Lecture notes in control and information sciences. Springer Verlag, 2002.

[3] A. Varga, A. Hansson, and G. Puyou, editors. *Optimization Based Clearance of Flight Control Laws. A civil aircraft application*. Number 416 in Lecture notes in control and information sciences. Springer Verlag, 2012.

[4] Lars Forssell and Andreas Sandblom. Optimisation-based clearance: The nonlinear anlysis. In *Advanced Techniques for Clearance of Flight Control Laws*, pages 415–430. Springer, 2002.

[5] Lars Forssell. Flight clearance analysis using global nonlinear optimisation-based search algorithms. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Austin, Texas, 11-14 August 2003. AIAA paper AIAA-2003-5481.

[6] P.P. Menon, J. Kim, D.G. Bates, and I. Postlethwaite. Clearance of nonlinear flight control laws using hybrid evolutionary optimisation. *IEEE Transactions on Evolutionary Computation*, 10(6):689–699, 2006.

[7] P.P. Menon, D.G. Bates, I. Postletwaite, A. Marcos, V. Fernandez, and S. Bennani. Worst-case analysis of flight control laws for re-entry vehicles. In *Proceedings of the IFAC Symposium on Automatic Control in Aerospace*, Toulouse, July 2007.

[8] L Forssell and A. Hyden. Optimisation based worst case search of ADMIRE. Technical report, GARTEUR, TP-119-26, 2002.

[9] G. W. RyanIII. A genetic search technique for identification of aircraft departures. In *Proceedings of the AIAA Flight Mechanics Conference*, USA, August 1995. AIAA paper AIAA-95-3453.

[10] P.P. Menon, A.A. Pashilkar, and K. Sudhakar. Identification of departure susceptibility for design of carefree maneuverable control scheme. In *Proceedings of the MSO-DMES Conference*, volume 1, Goa, India, 2003.

[11] D. Skoogh, P. Eliasson, F. Berefelt, R. Amiree, and D. Tourde. Clearance of flight control laws for time varying pilot input signals. In *6th IFAC Symposium on Robust Control Design*, June 2009.

[12] Daniel Skoogh and Fredrik Berefelt. Nonlinear programming methods for worst-case pilot input determination. In Andreas Varga, Anders Hansson, and Guilhem Puyou, editors, *Optimization Based Clearance of Flight Control Laws, A Civil Aircraft Application*. Springer, 2011.

[13] Daniel Skoogh and Fredrik Berefelt. Application of nonlinear programming methods for determination of worst-case pilot inputs. In Andreas Varga, Anders Hansson, and Guilhem Puyou, editors, *Optimization Based Clearance of Flight Control Laws, A Civil Aircraft Application*. Springer, 2011.

[14] Rafael Fernandes de oliveira and Guilhem Puyou. On the use of optimization for flight control laws clearance: a practical approach. In *Preprints of the 18th IFAC World Congress*, Milano, September 2011.

[15] A.E. Eiben and Smith J.E. *Introduction to Evolutionary Computing.* Springer, Berlin, 2003.

[16] A.M.S. Zalzala and P.J. Fleming, editors. *Genetic algorithms in engineering systems*, volume 55 of *Control Engineering Series.* The Institution of Electrical Engineers, 1997.

[17] K.V. Price, R.M. Storn, and J.A. Lampinen. *Differential Evolution.* Springer, Berlin, 2005.

[18] James C. Spall. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control.* Wiley, N.J., 2003.

[19] H.-G. Beyer and H. P. Schwefel. *Evolution strategies, Natural Computing 1.* Kluwer Academic Publishers, 2002.

[20] P.E. Gill and M.H. Wright. *Numerical Nonlinear Optimization.* Lecture Notes, 2002.

[21] J. Nocedal and S.J. Wright. *Numerical Optimization.* Springer, 1999.

[22] D.G. Luenberger. *Linear and Nonlinear Programming, second edition.* Addison Wesley, 1984.

[23] V. Torczon. On the convergence of pattern search algorithms. *SIAM J. Optim.*, 7(1):1–25, 1997.

[24] R.M. Storn and K.V. Price. Differential Evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359, 1997.

[25] A. Varga. Clearance criteria for civil aircraft. Technical Report D1.2.1, COFCLUO, EU, July 2007.

[26] B. Berefelt, P.M. Eliasson, M. Hagström, A. Lennartsson, A.S. Lesiario, J. Robinson, and D. Skoogh. Optimisation tool. Technical Report D-1:3:1-3, CESAR EU, 2010.

[27] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms.* Wiley, 2001.

[28] P. Eliasson and D. Skoogh. Methods for parameter sensitivity assessment in aircraft-missile models. Technical Report FOI-R–1005–SE, FOI, 2003.

[29] Andreas Sandblom. A simplified 3-dimensional surface-to-air missile model. Technical Report FOA-R–99-01283-314–SE, FOA, 1999.

[30] Daniel Skoogh and Fredrik Berefelt. Final report wp 2.5. Technical Report D2.5.5, COFCLUO EU, 2010.

[31] P Scholz, S.S. Mahmood, M. Casper, S. Wallin, D. Skoogh, and S Adden. Design of active flow control at a drooped spoiler configuration. In *31st AIAA Applied Aerodynamics Conference*, San Diego, FL, USA, 24-27 June 2013. AIAA paper AIAA-2013-2518.

[32] F. Berefelt. *Globopt 3.0 - A global optimization toolbox*. Forthcoming FOI report.

[33] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, Berlin, 1994.

[34] D.R Jones, C.D. Perttunen, and B.E. Stuckman. Lipschitzian optimization without the lipschitz constant. *Journal of Optimization Theory and Application*, 79(1), October 1993.

[35] D.E. Finkel and C.T. Kelley. Convergence analysis of the direct algorithm. Technical Report CRSC-TR04-28, N.C. State University Center for Research in Scientific Computation, July 2004.

[36] R. Hooke and T. A. Jeeves. Direct search solution of numerical and statistical problems. *J. Assoc. Comput. Mach*, 8:212–229, 1961.

[37] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley, New York, NY, 2001.

[38] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Springer, Berlin, 2003.

[39] Emil Salling, Peter Strömbäck, Jan Lindh, Casper Hildings, Niklas Wallin, and Mats Fredriksson. MERLIN software architecture document. Technical Report FOI-R--2486--SE, Totalförsvarets forskningsinstitut, FOI, Stockholm, May 2012.

[40] Emil Salling and Peter Strömbäck. MERLIN Defence Analyzer v1.2 User Manual. Technical Report FOI-R--3625--SE, Totalförsvarets forskningsinstitut, FOI, Stockholm, May 2012.

FOI, Swedish Defence Research Agency, is a mainly assignment-funded agency under the Ministry of Defence. The core activities are research, method and technology development, as well as studies conducted in the interests of Swedish defence and the safety and security of society. The organisation employs approximately 1000 personnel of whom about 800 are scientists. This makes FOI Sweden's largest research institute. FOI gives its customers access to leading-edge expertise in a large number of fields such as security policy studies, defence and security related analyses, the assessment of various types of threat, systems for control and management of crises, protection against and management of hazardous substances, IT security and the potential offered by new sensors.

FOI
Defence Research Agency
SE-164 90 Stockholm

Phone: +46 8 555 030 00
Fax: +46 8 555 031 00

www.foi.se