Tuning Robotti: the Machine-assisted Exploration of Parameter Spaces in Multi-Models of a Cyber-Physical System

Sergiy Bogomolov[1], John Fitzgerald[1], Frederik F. Foldager[2], Carl Gamble[1], Peter Gorm Larsen[3], Kenneth Pierce[1], Paulius Stankaitis[1], and Ben Wooding[1]

[1] School of Computing, Newcastle University, United Kingdom
Sergiy.Bogomolov,John.Fitzgerald,Carl.Gamble@ncl.ac.uk
Ken.Pierce,P.Stankaitis,B.Wooding1@ncl.ac.uk
[2] Agro Intelligence, Agro Food Park 13, 8200 Aarhus N, Denmark,
ffo@agrointelli.com
[3] DIGIT, Department of Engineering, Aarhus University, Aarhus, Denmark
pgl@eng.au.dk

Abstract. We describe a pilot study that aims to evaluate a systematic approach to tuning the design parameters of Cyber-Physical Systems by using machine-assisted Design Space Exploration supported by the INTO-CPS Toolchain. The study focuses on the design of a modified controller for a semi-autonomous agricultural robot. Beginning from a candidate multi-model built from VDM and 20-sim, the study uses data derived from test runs of the robot to tune selected parameters of a Continuous-Time model of the robot's physics in order to improve fidelity. The study raises questions around support for the selection of parameters for tuning, selection of test scenarios, and the ranking of results. We identify further work to complete the design loop by assessing the consequences of introducing a modified controller and deploying it in further field tests.

1 Introduction

Cyber-Physical Systems (CPSs) integrate computational and physical processes with data and network technologies, creating opportunities for new smart products and services and the digitalisation of industrial processes [24]. CPS design is inherently multi-disciplinary: a developer must bring together the products of diverse disciplines, often from diverse suppliers, with the attendant risks of incompatibilities and misunderstandings which may only come to light in prototyping. Model-based methods of *virtual engineering* [3] have been proposed as a means of addressing these challenges. Among the proposed advantages are the ability to use design models as a basis for optimising parameters by performing systematic Design Space Exploration (DSE) with the goal of reducing the number of prototyping cycles required to converge on an optimal design.

A model-based approach to multi-disciplinary CPS design entails the federation of semantically heterogeneous models. We call such federations *multi-models* and the co-ordinated execution of multi-models we term *co-simulation* [16]. Co-simulation allows the exploration of CPS-level behaviours that arise from the interaction between cyber

and physical elements. In practice, this means that open tool chains should allow for the collaborative construction and co-simulation of multi-models. However, there is only limited experience with DSE over such multi-models within CPS product development.

A practical challenge in CPS design is that of estimating the properties of the physical product, and hence ensuring that cyber elements such as control and communication are matched to physical system elements' properties, to deliver the desired CPS-level performance. In this paper, we report progress in a study that aims to pilot the use of DSE to address this challenge. The underlying co-simulation technology is that of the "Integrated Tool Chain for Model-based Design of Cyber-Physical Systems" (INTO-CPS) project [10]. This makes use of the mature Functional Mock-up Interface (FMI) 2.0 standard for co-simulation [6][4]. The core FMI-compliant tool of the INTO-CPS tool chain is a Co-simulation Orchestration Engine supporting both fixed and variable step-size simulations [27].

The ultimate goal of the study is to evaluate the use of DSE over a multi-model to determine optimal control parameters under different operating conditions for an autonomous agricultural field robot, Robotti [13]. The robot is a tool-carrier designed for accommodating a wide range of field operations. By combining the robot with several tools, a large number of load cases emerge. To reduce time to market and the cost of testing diverse configurations in the field, a structured simulation-based approach is needed to address the increasing number of possible configurations in a virtual setting.

The intention is to use DSE to optimise the steering controller, minimising cross-track error by co-simulation of a simple model representing the dynamics of the robot in a continuous-time (CT) formalism and a model representing the steering controller in a discrete-event (DE) formalism [9]. In the work performed to date and reported in this paper, the model of the machine dynamics is tuned experimentally to ensure that it satisfactorily captures the motion of the robot over a series of scenarios (which are intended to exercise the controller rather than reflecting real field-work). Model calibration and fidelity are critical to applicability [15,16].

We briefly discuss related work on DSE in Section 2. The relevant elements of the INTO-CPS tool chain and its underlying co-simulation are summarised in Section 3. The INTO-CPS support for DSE technology is explained in Section 4. The pilot study of the Robotti case is presented in Section 5 and the DSE analysis of it in Section 6. Finally, Section 7 discusses our results so far and future work.

2 Related Work

The techniques of automated DSE originated in the embedded systems domain, focusing on optimum placement of software functionality on constrained hardware. The need for efficient DSE was noted early [7] and applied to integrated circuits (e.g. [2]). The aerospace domain pioneered adoption of modelling and DSE at the larger-scale systems level, e.g. co-design of the Chinook helicopter's integrated circuit and controller [8] and to multi-objective energy-based optimisation [1]. As in agriculture, the component or components being optimised in aerospace are often from a single domain

[4] `http://fmi-standard.org`

(e.g. a turbine [5] or exhaust units [18]), although the objectives being optimised may be from multiple domains. Where DSE is applied with components from different domains, these models still tend to represent only the physical elements of the system (e.g. combined heat and power [25] and hybrid energy systems cells [23]).

A key challenge in DSE is that the size of the design space increases exponentially as the number and range of parameters is increased [17]. Large design spaces can be managed by increasing simulation resources, or by reducing the number of simulations required. Guidelines on experiment design can help engineers identify important factors to focus their searches, for example using the Taguchi Method [14]. This can be supported using techniques such as computational data clustering algorithms [21] and global sensitivity analysis [19].

3 The INTO-CPS Toolchain

INTO-CPS is an open tool chain that facilitates collaborative development and co-simulation for multi-models[5]. Although it supports traceable systems engineering activities from requirements analysis to test, the tool chain's central feature is support for construction and co-simulation of multi-models using FMI.

The tool chain allows requirements to be described using SysML [26] using a profile for CPSs which has been implemented in the Modelio tool [4]. This profile allows the architecture of a CPS to be described, including both computational and physical elements. Based on a profile-conformant model, one can automatically generate FMI descriptions for each constituent model. It is also possible to generate the connections between different Functional Mockup Units (FMUs) for a co-simulation.

The FMI model descriptions can be imported into diverse modelling and simulation tools. Each of these can produce detailed models that, exported as FMUs, can be incorporated into an overall co-simulation as independent simulation units. The element models can either be in the form of DE models or in the form of CT models combined in different ways. Thus, heterogeneous constituent models can then be built around this FMI interface, using the initial model descriptions as a starting point. A Co-simulation Orchestration Engine (COE) allows these constituent models to be evaluated through co-simulation [27]. The COE also allows real software and physical elements to participate in co-simulation alongside models, enabling both Hardware-in-the-Loop (HiL) and Software-in-the-Loop (SiL) simulation.

A web-based INTO-CPS Application has been produced to allow engineers to manage the multi-modelling and co-simulation processes [22]. This can be used to launch the COE, enabling multiple co-simulations to be defined and executed, and the results collated and presented automatically. The tool chain will allow these multiple co-simulations to be defined via DSE.

4 Design Space Exploration in INTO-CPS

DSE is the activity of evaluating multi-models from a collection of alternatives in order to reach a basis for subsequent more detailed design [14]. The set of multi-models under

[5] `www.into-cps.org`

consideration is termed a *design space*. The multi-models within a design space may differ from one another solely in terms of the values of specific *design parameters*, or may be more fundamentally different, for example in terms of structure or component elements. The purpose of DSE is to evaluate these multi-models under a series of co-simulations, in order to arrive at a *ranking* against an *objective* (or *cost function*), i.e., a set of criteria or constraints that are important to the developer, such as cost or performance. A multi-model or group of multi-models selected from a ranking might then form a basis for further design steps. The value of DSE is in prioritising cyber-physical prototypes which may have to be produced and on which expensive tests may have to be performed.

Design spaces for CPSs may be extremely large, and so different DSE techniques aim to explore the space of alternatives rapidly. These include both *open* and *closed*-loop approaches. In this kind of assessment there can be tradeoffs over multiple potentially competing objectives (e.g., speed and energy consumption). To enable the engineer to select optimal configurations it is necessary to have a convincing way to present the results, for example in the form of a *pareto-front*.

Since system-level properties of CPSs are often the result of interactions between computational and physical processes, it is worth noting that tradeoffs can be made across these domains. For example, one might want to examine any tradeoff between different error detection and recovery strategies in supervisory controller code and, say, the performance and cost of sensors. The ability to do this over a multi-model is one of the advantages of taking a multi-disciplinary co-simulation approach.

The INTO-CPS tool chain supports DSE by running co-simulations on selected points in the design space of interest. Both open loop (exhaustive search) and closed loop (generic search or simulated annealing) approaches can be supported in exploring alternative solutions [12].

Given system requirements (potentially including constraints for different aspects of a CPS) and a multi-model, a desired multi-model configuration can be create for the INTO-CPS Application. Using scripts implemented in a combination of Java and Python, the DSE analysis can be performed from the INTO-CPS Application, and the ranking of results can be displayed after individual simulations have been conducted. This workflow was followed in the process illustrated later in Figure 4.

INTO-CPS supports cloud deployment of DSE through the open-source CondorHTC framework, and genetic algorithms for reducing simulations [11]. Existing guidelines on experimental design for DSE over multi-models, for example using Taguchi Methods, can also help engineers using INTO-CPS [14].

5 Robotti Pilot Study: Overview and Experimental Setup

5.1 Goal of the Study

The purpose of this pilot study is to assess the viability of DSE as a tool in improving the fidelity of a multi-model, based on real-world data observed during field trials of a prototype system. The study is based on the multi-model of the Robotti unmanned platform developed by Agro Intelligence (Agrointelli) for field operations in commercial agriculture (see Figure 1).

Fig. 1: The Robotti unmanned platform

The challenge we address is that some physical properties can be difficult to estimate during design. These include those affected by changing conditions, such as surface friction and tyre stiffness. In order to develop a useful (multi-)model, we aim to approximate these values by measuring the overall performance of a prototype design and tuning the physical parameters of a simple model such that the modelled behaviour of the application matches the measurements. A number of field tests are carried out in order to characterise the actual motion of the robot as a function of given input trajectories. We then vary the physical parameters of the multi-model via DSE and identify the inputs that result in the closest correlation between the modelled and measured trajectories in corresponding scenarios. Note that we do not undertake modelling of the terramechanics including soil characteristics [28].

The goal is to be able to make a simple, low degree of freedom model with only a few parameters that, to an acceptable degree, describes the motion of the robot. The fundamentals of the CT model are also described in [20]. This allows us to tune the model without knowing and modelling in detail the entire physical system by simply finding the best possible set of parameters to describe this motion. In the end, if we can achieve a validated multi-model of both the CT model describing the dynamics and the DE model describing the controller, we can use this knowledge to provide an estimate of how to adjust the control parameters for the best possible performance based on a future round of DSE.

5.2 Field Tests and Scenarios

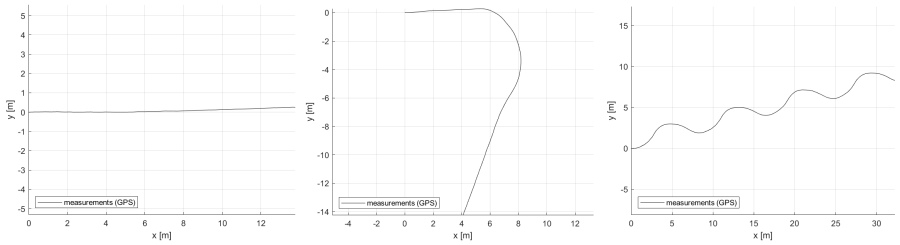
The field tests comprised a total of 27 runs across four scenarios. Each scenario is a pattern of control outputs designed to assess the dynamical response of Robotti to different motion profiles. Within each test run, the controller was set to “open loop” mode, where the control outputs change but there is no active control based on feedback. The control outputs and GPS position were recorded for each run. The position of the Robotti in a selection of test runs is shown in Figure 2. The four scenarios are:

Speed step: The speed of each wheel is increased incrementally in lock step, resulting in the Robotti driving straight as in Figure 2a. This test was repeated eight times.

Speed ramp: The speed of each wheel is increased smoothly. This also results in a straight path as in the speed step tests. This test was repeated three times.

Turn ramp: The speed of the right wheel is increased smoothly, while the left wheel is kept constant. This results in the turning motion seen in Figure 2b. This test was repeated eight times.

Sin: The speed of the left and right wheel is increased and decreased to produce the sinusoidal motion seen in Figure 2c. This test was repeated eight times.



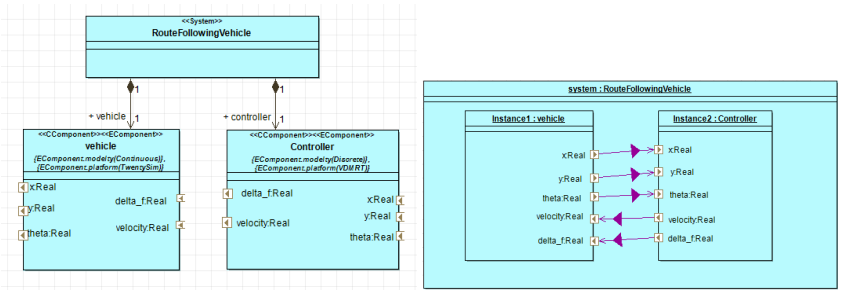
(a) Speed step / speed ramp (b) Turn ramp (c) Sin

Fig. 2: Path taken by the Robotti during field tests across various scenarios.

5.3 Baseline Robotti Multi-model

The baseline multi-model used for the DSE experiments comprises two FMUs: a CT model of the vehicle realised in 20-sim and a DE model of the open-loop controller realised in VDM-RT and Overture. Figure 3 shows the architecture of the multi-model and the connections between FMUs. The vehicle FMU receives two control signals from the controller FMU, *velocity* and *delta_f*. The former controls the overall velocity of the vehicle and the latter is the steering angle where a value 0 corresponds to driving in a straight line, and positive or negative values indicate right or left turns respectively. Higher absolute values of *delta_f* result in a sharper turn. The vehicle FMU reports the current position and rotation of the Robotti through output signals *x*, *y*, and *theta*.

In turn, the controller FMU sets the control signals as outputs and receives the position and rotation signals from the vehicle as inputs. In this pilot study, these inputs are not used. However this interface means that a closed-loop FMU containing a feedback controller can be easily substituted in future with a closed loop controller. The control outputs for the open-loop field tests were recorded in Comma-Separated Values (CSV) format. In order to perform the same experiments as carried out in the field in a co-simulation, it is necessary to "replay" these control outputs at the appropriate time



(a) Architecture Diagram (b) Connection Diagram

Fig. 3: SysML diagrams showing the baseline multi-model of the Robotti.

during co-simulation. The CSV library included with the Overture tool is used to load the data, then each data row is compared with the wall clock of the VDM-RT interpreter and the output set appropriately.

6 Robotti Pilot Study: Design Space Exploration

The objective of the DSE was to calibrate Robotti model parameters using results obtained from a physical robot trial. In this section we describe the DSE methodology applied (Section 6.1), the setup (Section 6.2) and results obtained so far (Section 6.3).

6.1 Robotti Pilot Study: DSE Analysis Overview

The INTO-CPS toolchain provides a systematic and flexible method for DSE. Flexibility is achieved by allowing the user to configure DSE analysis with a configuration file (.json format) which defines and orchestrates the exploration process. The methodology also provides an infrastructure for developing and importing external objective functions (as Python scripts) which are used to quantify and rank designs.

Figure 4 shows the workflow and key components of the Robotti DSE. The region in the dashed box represents an existing INTO-CPS DSE infrastructure which contains (and co-simulates) the Robotti multi-model (MM) and a series of manoeuvre scenarios (SN) with corresponding steering inputs (steering_inputs[s_n].csv) and recorded positions from the field trial (gps_position[s_n].csv) for a scenario s_n. Computing a cost of the specific design is performed by objective function script (Algorithm 1) which compares simulated (results[s_n].csv) and recorded positions of the Robotti. The whole DSE process is configured and orchestrated by configuration file (dse_configuration.json) which specifies process parameters such as search algorithm type, multi-model parameters ([cAlpha, mass, mu] which represent total mass of the vehicle, tyre stiffness and surface friction respectively) and their value ranges as well as objective function script file. Once the DSE experiment has been configured, an internal search algorithm simulates the Robotti multi-model with specific parameters and the objective function algorithm computes a cost value of the design. The result of this DSE stage is a dse_results.csv file which maps different simulated multi-model scenarios to a cost value.

The second stage of the Robotti DSE analysis was concerned with finding Robotti multi-model parameters ([cAlpha, mass, mu]) from a generated dse_results.csv data file that would universally fit all simulated scenarios. Finding the best parameter values for all scenarios can be translated to a minimisation problem where an optimisation algorithm attempts to minimise a sum of cost values for different scenarios (with identical [cAlpha, mass, mu] parameters). In this study the optimisation algorithm (OP) has been implemented as an external Python script, which takes dse_results.csv data file and returns best fitting parameter values for all scenarios.

6.2 Robotti Pilot Study: DSE Configuration and Objective Function

Performing DSE entails defining an objective function (Python script) and configuration file which will guide the exploration and ranking processes. The configuration

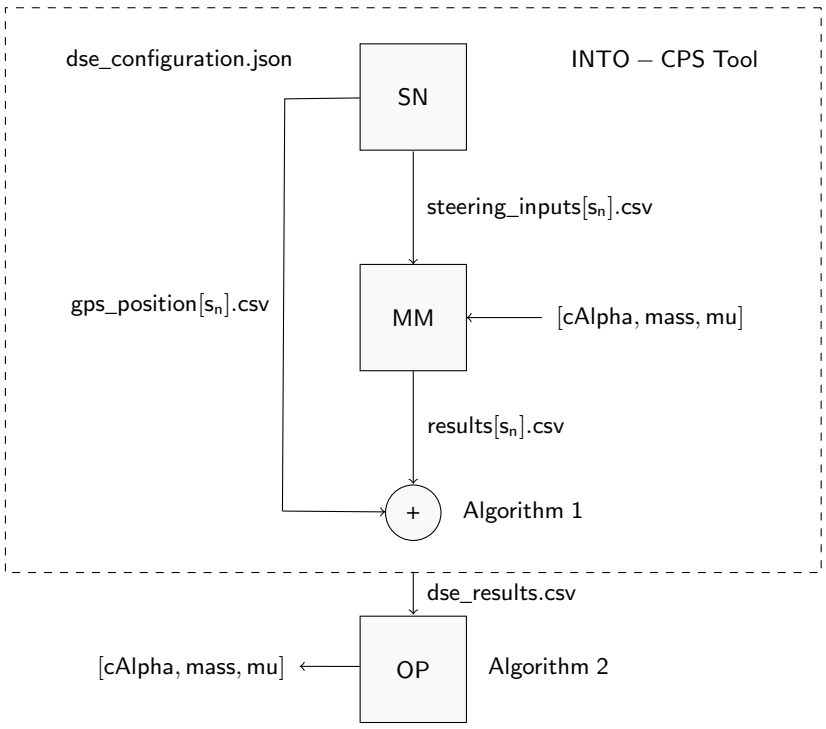


Fig. 4: The DSE analysis workflow overview of the Robotti pilot study

file for our pilot study is shown in Listing 1. It first defines the DSE search algorithm (in our study exhaustive). Next it defines an objective function (external script robottiCrossTrack.py) and simulation outputs used to produce an objective cost value. It then (line 10) specifies the ranges of design parameters which will be used by the exhaustive algorithm to generate different designs. Finally it gives the names of the scenarios to be used.

Listing 1: Robotti pilot study DSE configuration file

```
1  {
2  "algorithm": {"type":"exhaustive"},
3
4  "objectiveDefinitions": {"externalScripts":{ "robottiCrossTrack":
5  {"scriptFile":"robottiCrossTrack.py", "scriptParameters":{
6  "1":"time",
7  "2":"{Robotti}.RobottiInstance.y3_out",
8  "3":"{Robotti}.RobottiInstance.y4_out"}
9  },
10 "parameterConstraints":[],
11 "parameters":{
12 "{Robotti}.RobottiInstance.cAlphaF":[20k,24.5k,29k,33.5k,38k],
13 "{Robotti}.RobottiInstance.mu":[0.3,0.4,0.5,0.6,0.7],
14 "{Robotti}.RobottiInstance.m_robot":[1k,1.5k,2k,2.5k,3k]
15 },
16 "scenarios":["sin1, sin2, sin3, turn_ramp1, turn_ramp2, turn_ramp3,
       speed_ramp1, speed_ramp2, speed_step1, speed_step2, speed_step3"]
17 }
```

Given that the purpose here is calibration, we aim for minimal distance between measured and simulated results. Hence a root mean squared quadratic error is considered an appropriate cost function to evaluate. It was decided to base the tuning of the Robotti model parameters on the mean deviation between recorded and simulated robot positions for a specific manoeuvre. The deviation measure has been implemented with a root-mean-squared-error metric which measures mean a distance between two series of coordinates and can be expressed as Equation 1.

$$\mathsf{rmse} = \frac{\sum_{i=1}^{n} \sqrt{(x_{i2} - x_{i1})^2 + (y_{i2} - y_{i1})^2}}{n} \tag{1}$$

The measure was implemented as a Python script (described in Algorithm 1) which imports gps_pos.csv and simulation result results.csv files. The cross_track_error function then firstly applies a numerator part of the root-mean-squared-error measure for corresponding rows of imported files and once every position entry of the gps_pos.csv has been compared the function computes and returns a mean error value. The measured objective value with the name of objective name are then stored in the objective.json file with an internal tool function writeObjectiveToOutfile.

Algorithm 1 Root mean squared error objective function algorithm script

```
1: [Skeleton Code]
2:
3: import gps_pos.csv as gps
4: import results.csv as results
5:
6: def cross_track_error(gps, results) :
7:     result = [ ]
8:     for p in range[0, len(gps)]
9:         result.append[sqrt((gps[x_p] − results[x_p])^2 + (gps[y_p] − results[y_p])^2)]
10:    mean_error = sum(result)/len(gps)
11:    return mean_error
12:
13: objective_result = cross_track_error(gps, results)
14: writeObjectiveToOutfile(objectives.json, objectiveName, objective_result)
```

Once a cost value has been computed for all permutations of parameters and scenarios, all data is collated and a dse_results.csv file is produced, which in the Robotti DSE study contained the mapping between different simulated multi-model scenarios and a cost value.

6.3 Robotti Pilot Study: Optimum Parameter Search and Results

The main objective of this exercise is finding the best fitting parameter value of the model given the different simulated and measured scenarios. The problem of finding the best fitting parameters can be considered as optimisation (function minimization) problem where a function in Robotti pilot study to be minimised maps parameter values to a mean cross track value error computed by Algorithm 1.

The function we intend to minimise is given in Equation 1 where $\mathcal{E}_s$ is a set of mean track error functions constructed from the simulated scenarios and parameterised by a scenario $s \in \{sin1, sin2, sin3 \cdots\}$ and constrained domain $dom(\mathcal{E})$. The domain of the function $\mathcal{E}_s$ is identical to parameter ranges used by the exhaustive search algorithm (see lines 12-14 in Listing 1)

$$f^{-} = arg\ \min \sum_{s \in S} \mathcal{E}_s(\mathsf{cAlpha}, \mathsf{mass}, \mathsf{mu}) \tag{2}$$

The minimization function has been implemented as an external Python script and is described in Algorithm 2. The main function of the algorithm searchParameters explores all permutations of parameter values and for each permutation calls getResults function which takes specific parameter values and returns a sum of mean squared error values of all scenarios with given parameters. If the returned total_error_value is smaller than the current value of variable minimum then a new minimum value is assigned and optimal parameter values params are updated. The algorithm is exhaustive and will terminate once all parameter value permutations have been explored.

Algorithm 2 Total mean track error value minimisation algorithm

```
1:  def getResults(cAlpha, mass, mu) :
2:      total_error_value = 0.0
3:        for s in [s0, s1, ··· sn]
4:          total_error_value += Es(cAlpha, mass, mu)
5:      return total_error_value
6:
7:  def searchParameters() :
8:      minimum = init
9:      cAlphaF = [20000, 24500, 29000, 33500, 38000]
10:     mass = [1000, 1500, 2000, 2500, 3000]
11:     mu = [0.3, 0.4, 0.5, 0.6, 0.7]
12:
13:     for i0 in range[0, len(cAlphaF)]
14:         for i1 in range[0, len(mass)]
15:             for i2 in range[0, len(mu)]
16:                 if(getResults(cAlphaF[i0], mass[i1], mu[i2]) < minimum) :
17:                     minimum = getResults(cAlphaF[i0], mass[i1], mu[i2])
18:                     parameters = cAlphaF[i0], mass[i1], mu[i2]
19:
20:     return (minimum, parameters)
```

The results of the Robotti DSE exercise are summarised in Table 1 and Figures 5 and 6. In total 12 different Robotti manoeuvering scenarios have been considered where each scenario was simulated with 125 parameter [cAlpha, mass, mu] value variations giving us 1500 data points in dse_results.csv file. The optimisation algorithm found the best fitting parameter values Robotti multi-model are CAlphaF = 38000, mass = 1000, μ = 0.3 resulting in a total mean cross track error of 17.865.

In Table 1 for each scenario we provide the smallest and largest possible error values as well as average error over 125 parameter value permutations. The computed parameters for 4 scenarios out of 12 give a below average mean squared error value and 5

slightly above. The interesting case was a speed_step scenario (accelerating and braking in a straight line) where, as expected, a mean squared error value was unaffected by parameter variations and therefore had no effect on tuning Robotti parameters. From Figure 5 we can also notice how different parameter values were affecting four different groups of scenarios and we can notice that sin1 and turn_ramp1 manoeuvres correlate similarly with changing parameter values, whereas speed_ramp has the lowest mean squared error when mu is 0.3 and only very slightly was affected by varying other parameters. Figure 6 illustrates another correlation, which includes the scenario groups sin, speed_ramp, turn_ramp and sub-scenarios (e.g. sin1, sin2) in the individual sub-plots. With this figure we illustrate that the search space for the globally best parameters, where optimised parameters would be universally best was very small and for most scenario groups parameters affect mean squared error unevenly.

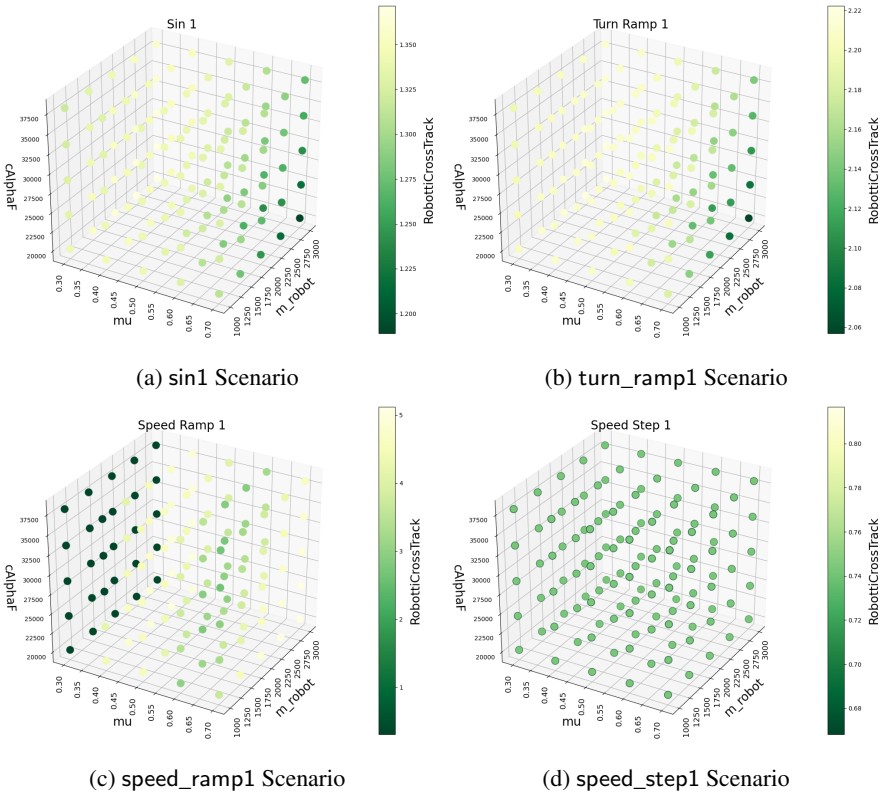
scenario	min_error	max_error	average_error	computed
sin1	1.188	1.371	1.323	1.318 (-5E-3)
sin2	1.680	1.960	1.884	1.891 (+7e-3)
sin3	2.139	9.218	4.561	5.486 (+9.25e-1)
speed_step1	0.742	0.742	0.742	0.742 (0)
speed_step2	0.578	0.578	0.578	0.578 (0)
speed_step3	0.482	0.482	0.482	0.482 (0)
turn_ramp	2.056	2.222	2.188	2.194 (+6e-3)
turn_ramp2	1.408	1.560	1.525	1.542 (+1.7e-2)
turn_ramp3	2.233	2.497	2.431	2.456 (+2.5e-2)
speed_ramp1	0.307	5.120	3.244	0.312 (-2.932)
speed_ramp2	0.367	3.965	3.328	0.430 (-2.898)
speed_ramp3	0.431	1.460	1.214	0.431 (-1.1709)

Table 1: The summary of optimisation results and best fitting parameters: $\text{mean\_cross\_track\_error}_{\text{total}} = 17.865$ with CAlphaF = 38000, mass = 1000, $\mu = 0.3$

7 Discussion

We have so far undertaken the data engineering needed to perform co-simulation on the Robotti CT model, and have demonstrated DSE-based tuning of this model over three design parameters against an objective function based on mean deviation between the model and the field data. The study, though so far incomplete, has proved technically viable for this scale of multi-model and volume of data. We here consider two main discussion areas based on the experience gained with DSE, INTO-CPS and the Robotti.

Firstly, as with all tools, there is a need for constant updating to stay relevant and useful in a changing technological landscape. While using INTO-CPS, we found difficulties in two main areas. First, INTO-CPS requires Python 2.7 which for most computers required backdating from the more recent Python 3, which is not backwards compatible. A real work-around was necessary to run INTO-CPS tools. Second, in ranking the data, the Pareto Method is built into the toolchain, it ranks simulations against two parameters to provide a Pareto front of optimal values. For any simulations that want to rank against a single parameter the functionality was not simple to find. We also



(a) sin1 Scenario

(b) turn_ramp1 Scenario

(c) speed_ramp1 Scenario

(d) speed_step1 Scenario

Fig. 5: Mean cross track error variation of scenarios against different parameter values

wanted to compare different scenarios in the same set of ranking which is also not a feature readily available in INTO-CPS. In the end we resorted to using Pareto with a second parameter, max cross track error, but as we were not interested in this parameter we ended up writing our own scripts for ranking and finding the optimal parameters across the scenarios.

Secondly, although parameter selection is important when considering the insights that DSE can offer, seeking to optimise performance through DSE, it is equally important to consider the scenarios. For example, in the sin scenarios, all three parameters we looked at affect the cross track error. However, none of them had an impact in the speed_step scenarios and whichever parameter combination was selected the same cross track error was measured. It was theorised that the parameters we discussed were only relevant when the robot would be turning, hence the largest impact appears for the sin wave which has many turns and no impact was felt on the speed_step which has no turning. To improve our results, it is necessary to consider the parameters which constantly affect the robot, not just when turning, so that the most accurate behaviour can be predicted.

In terms of further work, the next step in the pilot study is to complete the co-simulation of the ADRC controller with the plant model that has been tuned so far,

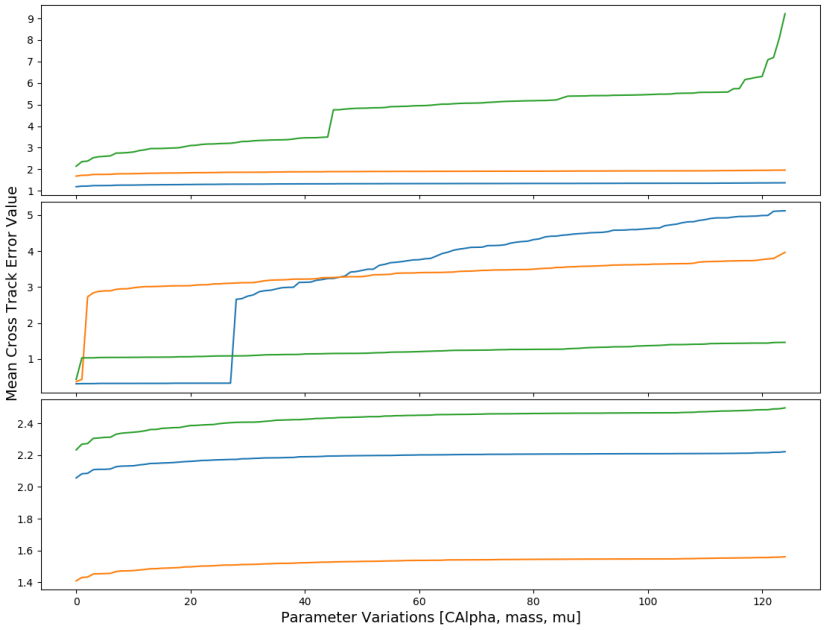


Fig. 6: Mean cross track error for different scenario groups (from top sinN, speed_rampN, turn_rampN) and scenarios (blue (N=1), orange (N=2), green (N=3))

and confirm the extent to which this is faithful to data gathered from field trials. An ADRC controller in itself has about 6 parameters, so exhaustive coverage may not be available within available resources, suggesting the use of genetic algorithms to select co-simulation runs. Aside from the tooling issues above, a priority for future work might be strengthening support for managing scenarios, and more user-friendly methods for generating custom rankings.

Acknowledgements. We are grateful to AgroIntelli for supplying Robotti data, and to Sadegh Soudjani at Newcastle. We acknowledge the European Union's support for the INTO-CPS and HUBCAP projects (Grant Agreements 644047 and 872698). This research was supported in part by the Air Force Office of Scientific Research under award number FA2386-17-1-4065. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the United States Air Force.

References

1. Ahmadi, P., Rosen, M.A., Dincer, I.: Multi-objective exergy-based optimization of a polygeneration energy system using an evolutionary algorithm. Energy 46(1), 21 – 31 (2012), http://www.sciencedirect.com/science/article/pii/S0360544212000990, energy and Exergy Modelling of Advance Energy Systems
2. Apvrille, L., Muhammad, W., Ameur-Boulifa, R., Coudert, S., Pacalet, R.: A uml-based environment for system design space exploration. In: 2006 13th IEEE International Conference on Electronics, Circuits and Systems. pp. 1272–1275 (Dec 2006)
3. Van der Auweraer, H., Anthonis, J., De Bruyne, S., Leuridan, J.: Virtual engineering at work: the challenges for designing mechatronic products. Engineering with Computers 29(3), 389–408 (Jul 2013)
4. Bagnato, A., Brosse, E., Quadri, I., Sadovykh, A.: The INTO-CPS Cyber-Physical System Profile. In: DeCPS Workshop on Challenges and New Approaches for Dependable and Cyber-Physical System Engineering Focus on Transportation of the Future. Vienna, Austria (June 2017)
5. Barzegar Avval, H., Ahmadi, P., Ghaffarizadeh, A., Saidi, M.H.: Thermo-economic-environmental multi-objective optimization of a gas turbine power plant with preheater using evolutionary algorithm. International Journal of Energy Research 35(5), 389–403 (2011), https://onlinelibrary.wiley.com/doi/abs/10.1002/er.1696
6. Blochwitz, T., Otter, M., Arnold, M., Bausch, C., Clauß, C., Elmqvist, H., Junghanns, A., Mauss, J., Monteiro, M., Neidhold, T., Neumerkel, D., Olsson, H., Peetz, J., Wolf, S., Gmbh, G.I.T.I., Oberpfaffenhofen, D.L.R.: The Functional Mockup Interface for Tool independent Exchange of Simulation Models. In: 8th International Modelica Conference. pp. 105–114. Munich, Germany (September 2011)
7. Chou, P., Ortega, R.B., Borriello, G.: Interface co-synthesis techniques for embedded systems. In: Proceedings of the 1995 IEEE/ACM International Conference on Computer-aided Design. pp. 280–287. ICCAD '95, IEEE Computer Society, Washington, DC, USA (1995), http://dl.acm.org/citation.cfm?id=224841.225052
8. Chou, P.H., Ortega, R.B., Borriello, G.: The chinook hardware/software co-synthesis system. In: Proceedings of the 8th International Symposium on System Synthesis. pp. 22–27. ISSS '95, ACM, New York, NY, USA (1995), http://doi.acm.org/10.1145/224486.224491
9. Christiansen, M.P., Larsen, P.G., Jørgensen, R.N.: Robotic Design Choice Overview using Co-simulation and Design Space Exploration. Robotics pp. 398–421 (October 2015)
10. Fitzgerald, J., Gamble, C., Larsen, P.G., Pierce, K., Woodcock, J.: Cyber-Physical Systems design: Formal Foundations, Methods and Integrated Tool Chains. In: FormaliSE: FME Workshop on Formal Methods in Software Engineering. ICSE 2015, Florence, Italy (May 2015)
11. Fitzgerald, J., Gamble, C., Payne, R., Lam, B.: Exploring the Cyber-Physical Design Space. In: Proc. INCOSE Intl. Symp. on Systems Engineering. vol. 27, pp. 371–385. Adelaide, Australia (2017), http://dx.doi.org/10.1002/j.2334-5837.2017.00366.x
12. Fitzgerald, J., Gamble, C., Payne, R., Larsen, P.G., Basagiannis, S., Mady, A.E.D.: Collaborative model-based systems engineering for cyber-physical systems, with a building automation case study. INCOSE International Symposium 26(1), 817–832 (2016)
13. Foldager, F., Balling, O., Gamble, C., Larsen, P.G., Boel, M., Green, O.: Design Space Exploration in the Development of Agricultural Robots. In: AgEng conference. pp. 314–321. Wageningen, The Netherlands (July 2018)

14. Gamble, C., Pierce, K.: Design space exploration for embedded systems using co-simulation. In: Fitzgerald, J., Larsen, P.G., Verhoef, M. (eds.) Collaborative Design for Embedded Systems, pp. 199–222. Springer Berlin Heidelberg (2014)
15. Gomes, C., Thule, C., Broman, D., Larsen, P.G., Vangheluwe, H.: Co-simulation: State of the art. Tech. rep. (feb 2017), http://arxiv.org/abs/1702.00686
16. Gomes, C., Thule, C., Broman, D., Larsen, P.G., Vangheluwe, H.: Co-simulation: a Survey. ACM Comput. Surv. 51(3), 49:1–49:33 (May 2018)
17. Kang, E., Jackson, E., Schulte, W.: An approach for effective design space exploration. In: Calinescu, R., Jackson, E. (eds.) Foundations of Computer Software. Modeling, Development, and Verification of Adaptive Systems. pp. 33–54. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
18. Karaali, R., Ärlhan Tekin ÃŰztÃijrk: Thermoeconomic optimization of gas turbine cogeneration plants. Energy 80, 474 – 485 (2015), http://www.sciencedirect.com/science/article/pii/S0360544214013619
19. Kucherenko, S., Feil, B., Shah, N., Mauntz, W.: The identification of model effective dimensions using global sensitivity analysis. Reliability Engineering & System Safety 96(4), 440 – 449 (2011), http://www.sciencedirect.com/science/article/pii/S0951832010002437
20. Legaard, C.M., Gomes, C., Larsen, P.G., Foldager, F.F.: Rapid Prototyping of Self-Adaptive-Systems using Python Functional Mockup Units. In: 2020 Summer Simulation Conference. SummerSim '20, ACM New York, NY, USA, Virtual event (2020)
21. Li, Q., Mahmoud, E., Michael, R.: A combinatorial approach to tradespace exploration of complex systems: A cubesat case study. INCOSE International Symposium 27(1), 763–779 (2017), https://onlinelibrary.wiley.com/doi/abs/10.1002/j.2334-5837.2017.00392.x
22. Macedo, H.D., Rasmussen, M.B., Thule, C., Larsen, P.G.: Migrating the INTO-CPS Application to the Cloud. In: Sekerinski, E., Moreira, N., Oliveira, J.N., Ratiu, D., Guidotti, R., Farrell, M., Luckcuck, M., Marmsoler, D., Campos, J., Astarte, T., Gonnord, L., Cerone, A., Couto, L., Dongol, B., Kutrib, M., Monteiro, P., Delmas, D. (eds.) Formal Methods. FM 2019 International Workshops. pp. 254–271. Springer-Verlag, LNCS 12233 (2020)
23. Mamaghani, A.H., Najafi, B., Shirazi, A., Rinaldi, F.: Exergetic, economic, and environmental evaluations and multi-objective optimization of a combined molten carbonate fuelÂăcell-gas turbine system. Applied Thermal Engineering 77, 1 – 11 (2015), http://www.sciencedirect.com/science/article/pii/S135943111401134X
24. Reimann, M., Rückriegel, C., et al.: Road2CPS: Priorities and Recommendations for Research and Innovation in Cyber-Physical Systems. Steinbeis-Edition (2017)
25. Sanaye, S., Katebi, A.: 4e analysis and multi objective optimization of a micro gas turbine and solid oxide fuel cell hybrid combined heat and power system. Journal of Power Sources 247, 294 – 306 (2014), http://www.sciencedirect.com/science/article/pii/S0378775313014158
26. OMG Systems Modeling Language (OMG SysML$^{\mathrm{TM}}$). Tech. Rep. Version 1.5, Object Management Group (May 2017), http://www.omg.org/spec/SysML/1.5/
27. Thule, C., Lausdahl, K., Gomes, C., Meisl, G., Larsen, P.G.: Maestro: The INTO-CPS co-simulation framework. Simulation Modelling Practice and Theory 92, 45 – 61 (2019), http://www.sciencedirect.com/science/article/pii/S1569190X1830193X
28. Wong, J.Y., Reece, A.: Prediction of rigid wheel performance based on the analysis of soil-wheel stresses part i. performance of driven rigid wheels. Journal of Terramechanics 4(1), 81 – 98 (1967), http://www.sciencedirect.com/science/article/pii/002248986790105X