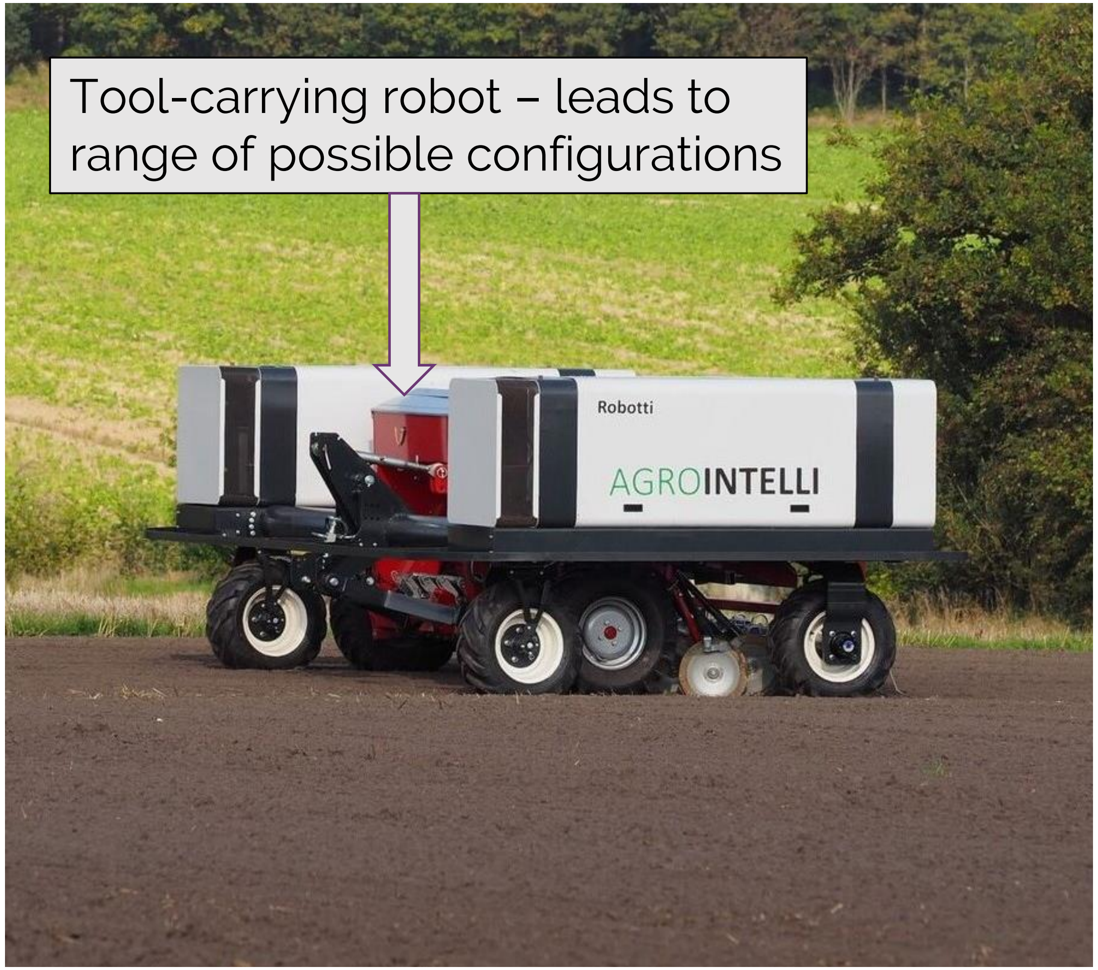
Tuning Robotti: the Machine-assisted Exploration of Parameter Spaces in Multi-Models of a CPS

Sergiy Bogomolov, John Fitzgerald, Frederik Foldager, Carl Gamble, Peter Gorm Larsen, Kenneth Pierce, Paulius Stankaitis, Ben Wooding

Content

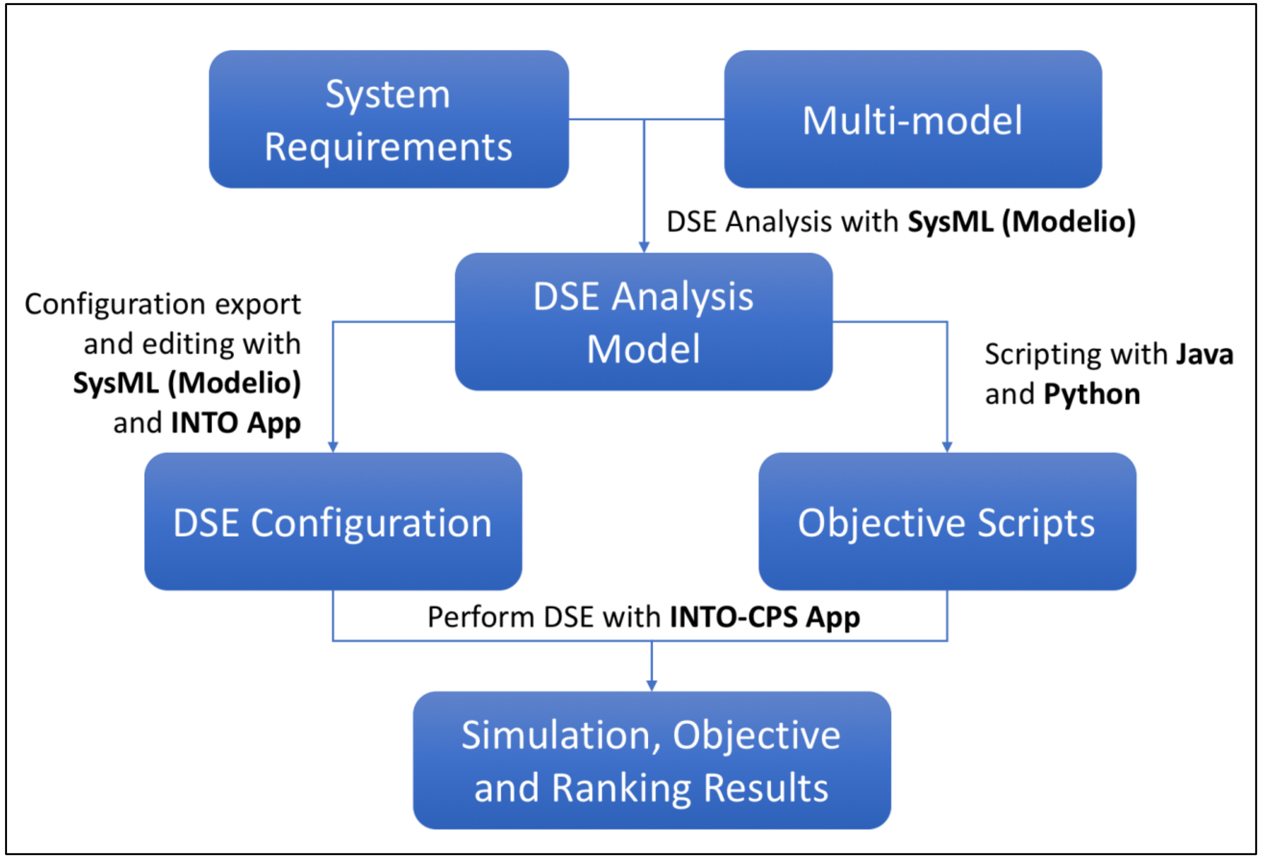
Background

- **Challenge in CPS Design lies in estimating properties of the physical product**
- **Can Design Space Exploration (DSE) help?**
- **We report progress on a pilot study**
  - **Aim: is to determine optimal steering control parameters for a field robot, Robotti**
  - Minimise cross-track error by co-simulation of a simple model representing robot dynamics in a continuous-time (CT) formalism with a model representing the steering controller in a discrete-event (DE) formalism.
- **So far … model of machine dynamics tuned experimentally to ensure that it satisfactorily captures the motion of the robot over a series of scenarios**



DSE in INTO-CPS

- DSE is activity of evaluating multi-models from a collection of alternatives (the *design* space) to reach a basis for subsequent more detailed design.
- Multi-models may differ in terms of *design parameters* or more fundamental properties such as architecture.
- *Ranking* against an *objective* or *cost function.*
- Design spaces may be large!
- Multiple competing objectives



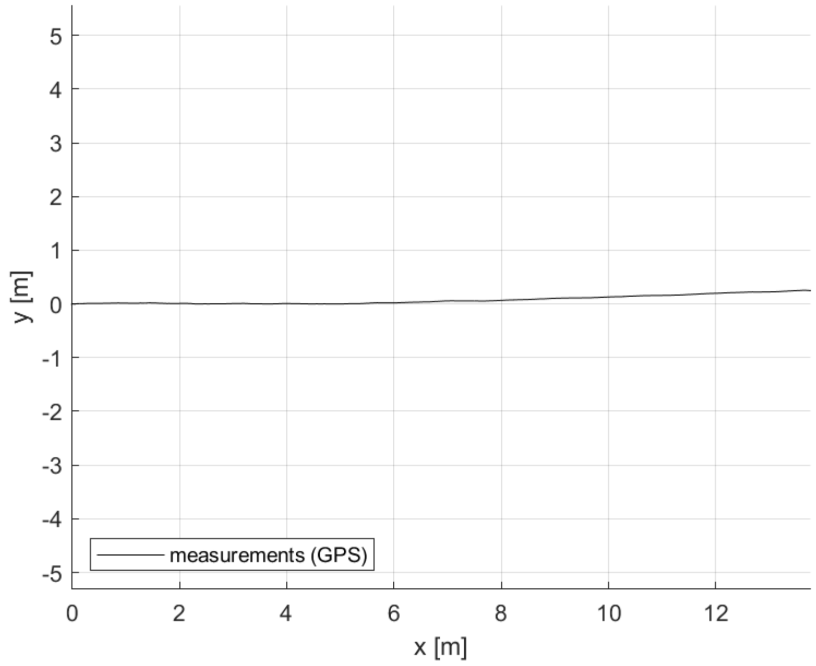
Outline of INTO-CPS approach to DSE

The Robotti Study

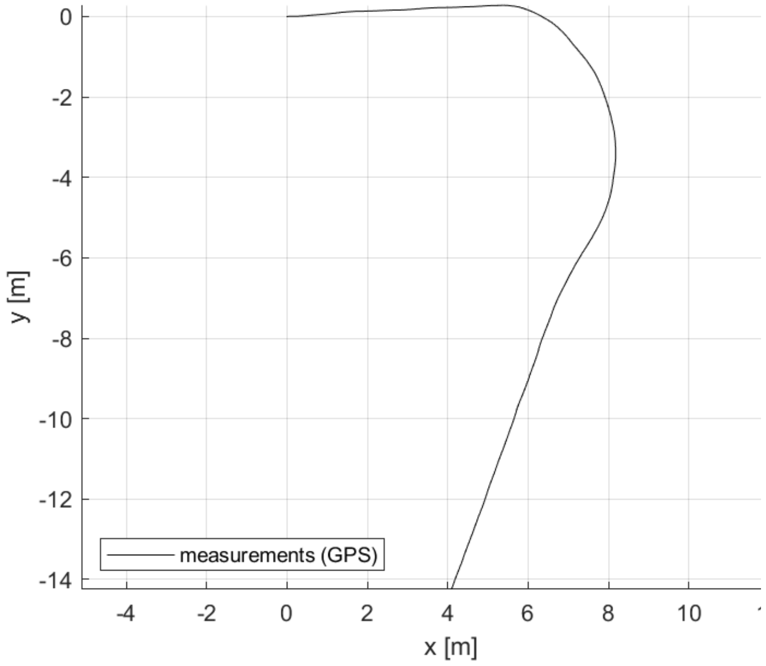
Purpose: assess the viability of DSE as a tool in improving model fidelity, based on real-world data during field trials of the prototype product.

Field Tests and Scenarios

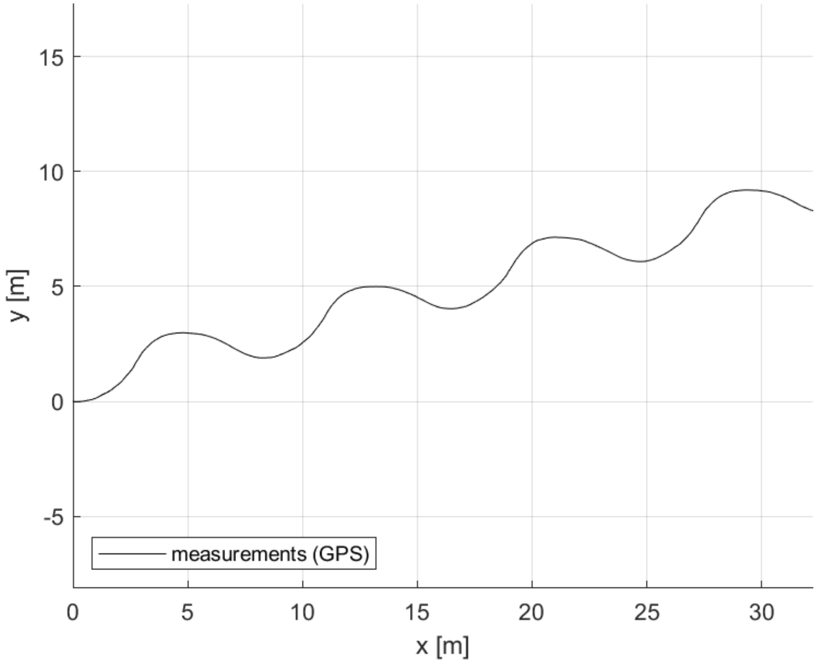
- 27 runs, 4 scenarios
- Each scenario is a pattern of control outputs selected to assess dynamical response to different motion profiles (not to reflect typical field use)
- Control outputs and GPS positions recorded for each run:



Speed step / speed ramp

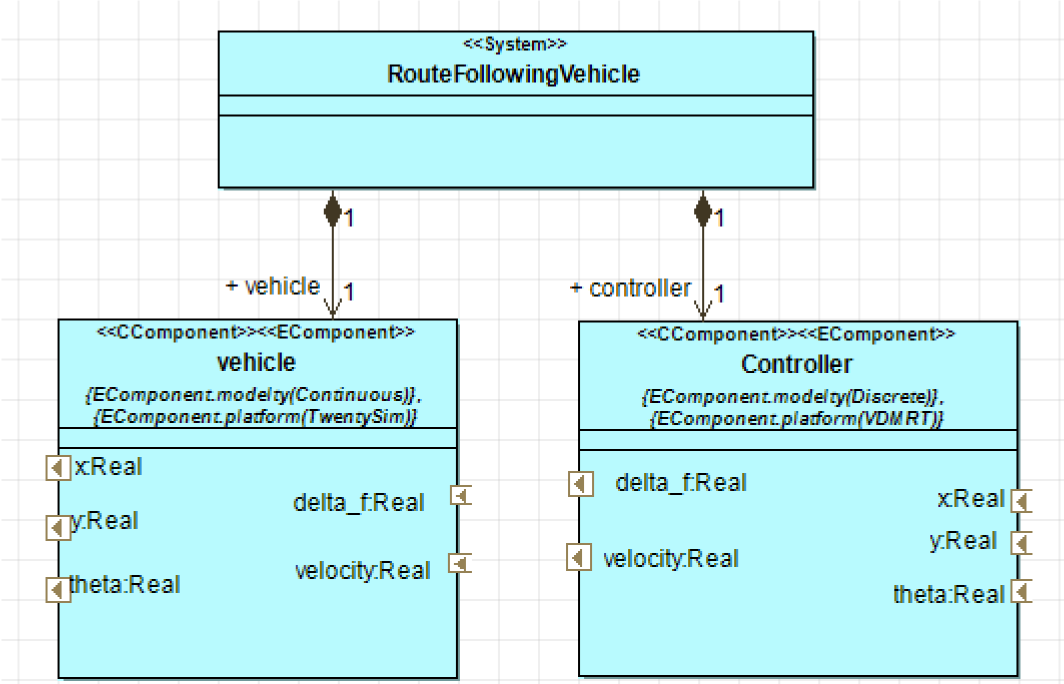


Turn ramp



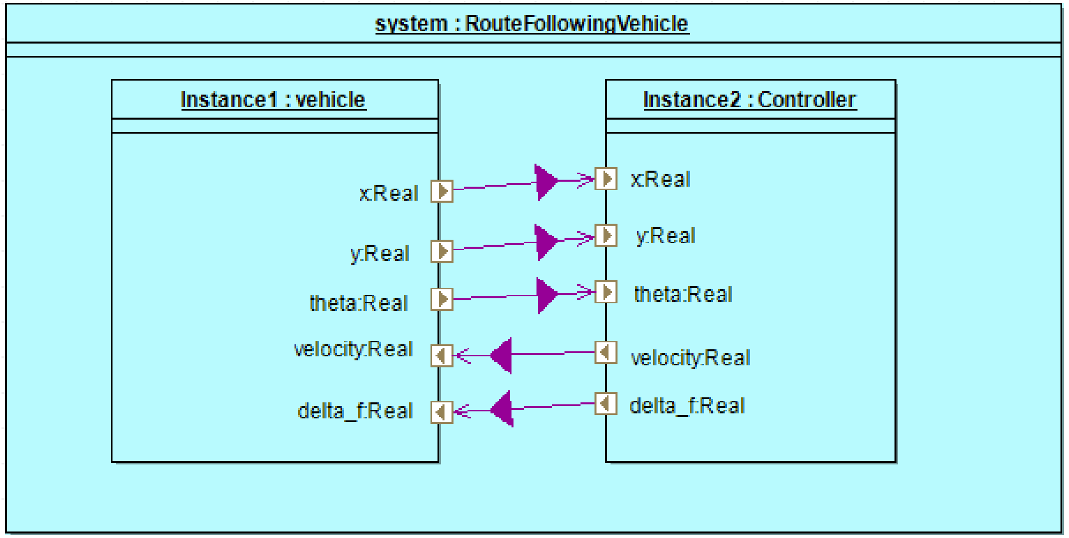
Sin

Baseline Robotti Multi-model



20-sim vehicle model

VDM-RT/Overture model of open-loop controller



x,y are vehicle position
theta is vehicle rotation

(Not used in pilot study)

velocity is vehicle velocity
delta_f is steering angle
(0 = straight ahead)

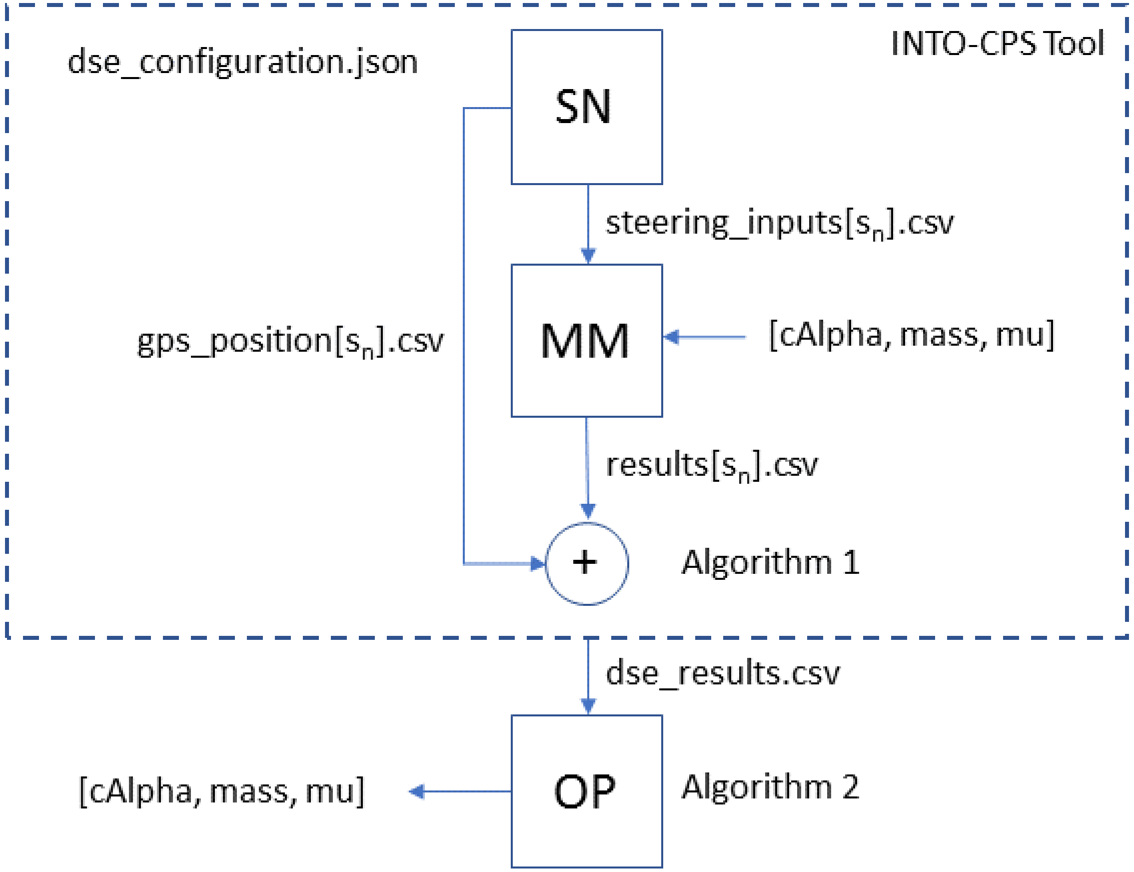
Overview of DSE

- **Configuration file defines and orchestrates DSE**
- **External python scripts code objective functions to quantify and rank designs.**

- **SN is manoeuvring scenario files**
- **MM is the multi-model**
- **Algorithm 1 compares simulated results with actual recorded Robotti positions.**

- **Config file specifies process parameters:**
  - cAlpha is tyre stiffness
  - mass is total vehicle mass
  - mu is surface friction

- **OP is optimisation algorithm returning best fitting parameter values for all scenarios**



Newcastle University

DSE Config

- DSE search algorithm
- Definition of objective function and simulation outputs
- Specifies ranges of design parameters used by exhaustive algorithm
- Identifies the scenarios

```
{
"algorithm": {"type":"exhaustive"},

"objectiveDefinitions": {"externalScripts":{ "robottiCrossTrack":
{"scriptFile":"robottiCrossTrack.py", "scriptParameters":{
"1":"time",
"2":"{Robotti}.RobottiInstance.y3_out",
"3":"{Robotti}.RobottiInstance.y4_out"}
},
"parameterConstraints":[],
"parameters":{
"{Robotti}.RobottiInstance.cAlphaF":[20k,24.5k,29k,33.5k,38k],
"{Robotti}.RobottiInstance.mu":[0.3,0.4,0.5,0.6,0.7],
"{Robotti}.RobottiInstance.m_robot":[1k,1.5k,2k,2.5k,3k]
},
"scenarios":["sin1, sin2, sin3,
              turn_ramp1, turn_ramp2, turn_ramp3,
              speed_ramp1, speed_ramp2,
              speed_step1, speed_step2, speed_step3"]
}
```

Objective function

- Aim for minimal distance between measured and simulated results
- Implemented as a Python script
- Cost computed for all permutations of the parameters and scenarios and returned as dse_results.csv

$$\text{rmse} = \frac{\sum_{i=1}^{n} \sqrt{(x_{i_2} - x_{i_1})^2 - (y_{i_2} - yi_{1})^2}}{n}$$

Optimum parameter search

- Find best fitting parameter value of the model
- Implemented as external Python script

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

Summary of Results

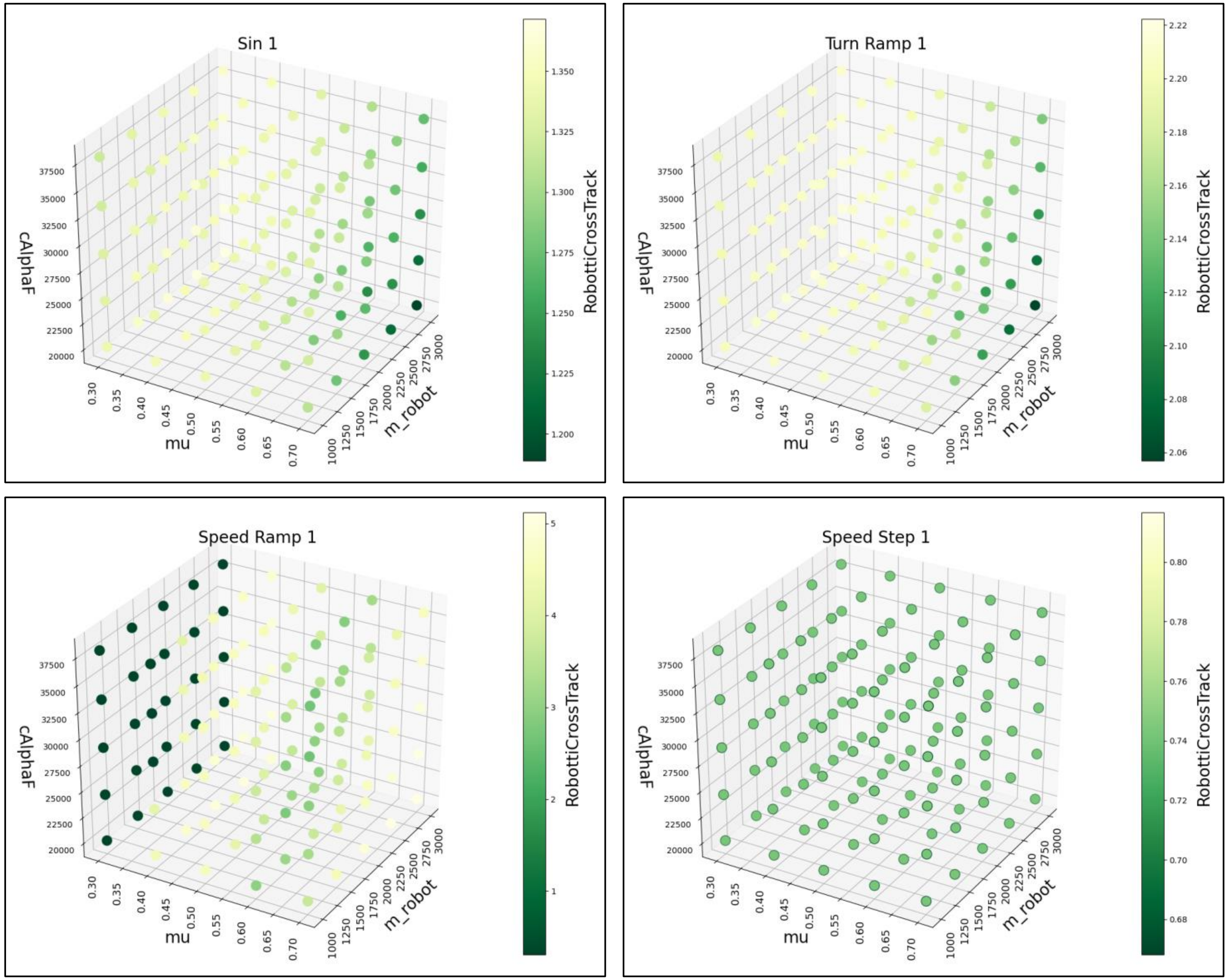
- **12 scenarios, each with 125 parameter value variations**
- **Speed-step scenario error value unaffected by parameter variations and so had no effect on tuning parameters.**

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)

Best fitting parameter values are CAlphaF= 38000, mass= 1000, mu = 0.3 resulting in a total mean cross track error of 17.865. These give above average errors in 5 and below average errors in 4.

Summary of Results

- **Sin1 and turn_ramp1 manoeuvres show similar correlations to chaging parameter values**
- **… but speed_ramp has lowest error at mu=0.3 and only slightly affected by the other parameters.**
- **For most groups of scenarios, the parameters affect error unevenly.**



Observations

- Much of the work was initial data engineering
- Demonstrated DSE-based tuning of the model to real observed data over three design parameters against objective based on men deviation between model and field data.
- Study incomplete: next step is co-simulation with ADRC controller and confirm extent of fidelity to field trial data.
- Requirement for Python 2.7
- Ranking set up for two parameters
  - Awkward to find 1-parameter ranking
  - Missed ability to compare across scenarios
- Considering scenarios as well as parameters
  - Parameters not influencing speed_step,

Thank You!