Uncertainty quantification and runtime monitoring using environment-aware digital twins

Jim Woodcock[1,2], Cláudio Gomes[2],
Hugo Daniel Macedo[2], and Peter Gorm Larsen[2]

[1] Department of Computer Science, University of York, UK jim.woodcock@york.ac.uk
[2] DIGIT, Department of Electrical and Computer Engineering, Aarhus University, Denmark,
{claudio.gomes,hdm,pgl}@ece.au.dk

Abstract. A digital twin for a Cyber-Physical System includes a simulation model that predicts how a physical system should behave. We show how to quantify and characterise violation events for a given safety property for the physical system. The analysis uses the digital twin to inform a runtime monitor that checks whether the noise and violations observed fall within expected statistical distributions. The results allow engineers to determine the best system configuration through what-if analysis. We illustrate our approach with a case study of an agricultural vehicle.

1 Introduction

We engineer a Cyber-Physical System (CPS) using separate models for its different parts [13,16,32,27,9]. If we use different formalisms, then this is a multi-model and we must use the corresponding support tools in coordination. In particular, we must connect the different simulation tools in a co-simulation. A co-simulation is a generalised form of simulation where different simulation tools are coupled together (see [14,19] for an introduction to the topic).

Typically, we use multi-models only in the engineering phase of the CPS in question (e.g., as in the V-process [31]). Recently, we have been reusing multi-models as digital twins after deploying the CPS (see [10]). We stream sensor and actuation signals from the CPS (the physical twin) to the multi-model co-simulation (the digital twin). We then predict how the physical twin should behave. However, the predictions can never be 100% accurate. This is because the multi-models cannot capture the full detail of physical reality: the sensor data is noisy and sampled at a finite frequency; the numerical solution of the multi-models is an approximation; and the environment for the physical twin may be different from the one used for prototyping.

An important practical question is: how large can these discrepancies be before we judge that the CPS is no longer behaving as we intended? We cannot hope to answer this question in general, as it depends on the CPS, the level of

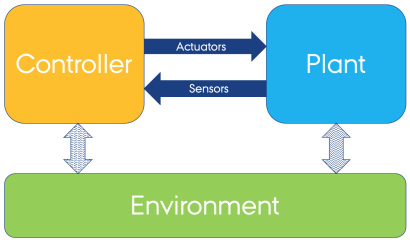

Fig. 1: The necessary components of a CPS operating in its environment.

detail of the multi-models, and the properties of the CPS that we are interested in. However, we can provide tools and methods that make it easier to specify allowed discrepancies.

In this paper, we try to answer the above question based on a case study representing an agricultural vehicle. The data used is simulated data, but representative of the real system. The reproducibility code for the results can be found in [15]. This work is a stepping stone towards bridging numerical and statistical simulation techniques with the runtime monitoring and verification field, where important techniques have been developed to synthesise monitoring algorithms [2,5,8]. We show how, by embedding noise into the model of the system, we may understand how that noise impacts the system's behaviour (e.g., are safety boundaries still respected?), and devise a runtime monitor that checks whether the noise observed in practice still corresponds to the modelled noise. When this condition is not observed, actions need to be taken to ensure that the system continues to operate correctly.

After this introduction, Section 2 motivates the need to operate with tolerances in a digital twin context. Afterwards Section 3 presents a case study with an agricultural vehicle. Finally Section 4 discusses how it is possible to talk about safety in the presence of statistical inaccuracies and provides a few concluding remarks and points towards the future work we wish to carry out.

2 The Need for Digital Twins and Tolerance

A simplified[3] representation of a CPS is depicted in Fig. 1. It comprises a digital component that controls a physical asset operating in some environment. In order to study the behaviour of the CPS, having a good model of the environment is as important as having a model of both the digital and the physical component, as the environment affects the behaviour of both.

[3] In practice, there may be many sub-components of each of the controller, plant, and environment, elements, and they may have complex interactions.

As stated in Kritzinger et al. [18] and Tao et al. [30], a digital twin can be used to monitor the conformance of the real CPS to these models, and possibly affect the real CPS when such conformance is violated.

Here we define the digital twin as the system that ensures the correct functioning of the system, aptly named the physical twin. For some systems, the digital twin can be realised with sensory data directly. For more complex systems, a combination of state estimators, data fusion algorithms, and numerical simulation, might be required to get a more comprehensive state of the system. As a simple example, suppose we want to monitor the torque acting on an electrical agricultural vehicle's wheels. The digital twin can read the current on the motor of each wheel. It measures this from sensory data. If the digital twin has a well-calibrated torque constant, then the torque is just the multiplication of the torque constant by the current measured.

Discrepancies between the values observed from the system and the values computed by the digital twin may occur for the following reasons:

1. Sensor data represent delayed discrete samples of the system;
2. Sensors (and actuators) have inaccurate and noisy readings.
3. The models used by the digital twin (e.g., used to derive data) do not fully represent the physical twin.
4. There are processing delays in the digital twin.

The statistician George Box is famous for the quote: "All models are wrong, but some are useful" [3]. This is sometimes used as an excuse for bad models, but that is not what Box meant. Box clarifies this: "Remember that all models are wrong; the practical question is how wrong do they have to be to not be useful?" [4]. The same question applies to discrepancies between digital twin and the system.

To give an example, suppose we need to check whether the physical twin satisfies some safety property. In the face of uncertainty of noisy sampled sensor data, how do we know that a violation is not a false positive? Or that a non-violation is a real non-violation? These questions highlight the need to incorporate noise models, which can be considered environment models, into the digital twin, and monitor whether those noise models correspond to the noise observed in practice.

If we successfully address the above questions, we can use the detection of violations for:

- Switching system operation to a fail-safe mode (as in Mitsch et al. [24]).
- Alerting human operators.
- Triggering a recalibration of the digital twin in a tracking simulator (e.g., [23,25,20]) (if the violation represents a change in the environment).

– Triggering maintenance activities (if the violation represents a change in the system's components, e.g., due to wear and tear).
– Triggering root-cause analysis [28] (e.g, by running multiple simulations with varying parameters that try to explain the violation observed).

3 The Agricultural Vehicle Case Study

Autonomous agricultural systems are a prime application domain for digital twin technology. They are complex systems working in remote environments with a high degree of intrinsic uncertainty.

As example of the need to monitor these vehicles, consider the following.

Example 1. The speed controllers on the wheels of an agricultural vehicle are tuned such that, for a particular static and viscous friction profile, they quickly settle on the speed that is commanded by the user. This is highly beneficial not just from a safety point of view, but also because it enables the use of path planning and job scheduling algorithms that rely only on kinematic[4] models. With wear and tear, manufacturing variability, unexpected operating temperatures, etc, the friction profile used to tune the speed controller becomes outdated. This makes the controllers perform poorly, and may invalidate the kinematic models used for planning.

Monitoring alleviates the problem highlighted in Example 1 by, e.g., measuring the current drawn and speed of the motors to detect when the controllers have deviated from the original behaviour (obtained through simulations and initial experimentation with the system). We must tolerate certain deviations because they might be due to noisy current measurements. We can specify a threshold as a function of the uncertainty in current measurements. Deviations that exceed the threshold trigger one of the following actions:

1. Alert human operators to lubricate the joints of the vehicle (i.e., change the physical system to match the kinematics).
2. Alert human operators to re-tune the controller parameters (i.e., calibrate the controller's models of friction).
3. Recalibrate the kinematic models, finding new acceleration and speed profiles.

[4] Some terminology. *Kinematics* is the space of all possible configurations of a system at one time without considering the forces acting on the system. For example, invariants between state variables that follow from conservation laws. *Dynamics* is how configurations change as a function of time, due to the forces acting on the system.

Each of the above actions has a different impact in the operations of the vehicle. For instance, option (3) invalidates all prior path planning and optimisation results.

We now show two analyses to quantify the uncertainty in the measurements of the physical system. The first analysis is a well known Montecarlo simulation, which consists of running several simulations, with noise drawn from statistical distributions that characterise the environment of the system. The second consists of combining what-if analysis with Montecarlo co-simulations. It enables designers to understand which noise sources have the biggest impact in the safety of the vehicle. Both analyses use models of the system environment (in this case, the noise).

We introduce a simple agricultural vehicle and derive its kinematic and controller equations. We can generalise the analysis described here to the more complex system described in Foldager et al. [11] and Macedo et al. [22].

3.1 Vehicle Kinematics

We derive simple kinematic equations for the agricultural vehicle. We consider it to be a bicycle model[5] with front steering and centre of gravity at the centre of the rear axis. For a more general derivation see Rajamani [26, Section 2.2].

Figure 2 gives a diagram of the agricultural vehicle, inspired by Alur [1]. The vehicle has four wheels, two at the back that have a fixed direction and two at the front that the driver can steer towards the left or the right. We define the vehicle's current position to be the centre of the rear axle (x, y). The distance between the front and back axles is $L\,\mathsf{m}$. The vehicle is moving forward at a speed of $v\,\mathsf{ms}^{-1}$. The front wheels are at an angle of $\theta°$ to the frame of reference (the x-y axes in the diagram). The driver has turned the front wheels to the right by an angle of $\delta°$. Because of its steering geometry, this is causing the vehicle to follow a circle with the origin at point C. We determine the centre of the turning circle by producing two lines, one following the rear axle and the other following the front axle rotated clockwise by the steering angle δ.[6]

[5] The term "bicycle model" is used because the equations assume that the angle of each front wheel is the same, as happens with vehicles that have only one front wheel. In practice, the angle between the two front wheels differs and the difference is proportional to the distance between them (as in Ackermann steering geometry).

[6] We assume that the vehicle moving at a constant speed describes a circle about its forward travelling frame of reference. In reality, steering is more complicated than this. It depends on whether the vehicle is front or rear-wheel driven; whether the front or rear-wheels steer the vehicle; what particular steering geometry is present; what the tyre characteristics are; and whether there are differential axles. A standard configuration is for the vehicle to have front-wheel steered driving wheels with Ackermann steering geometry (Zhao et al. show how to

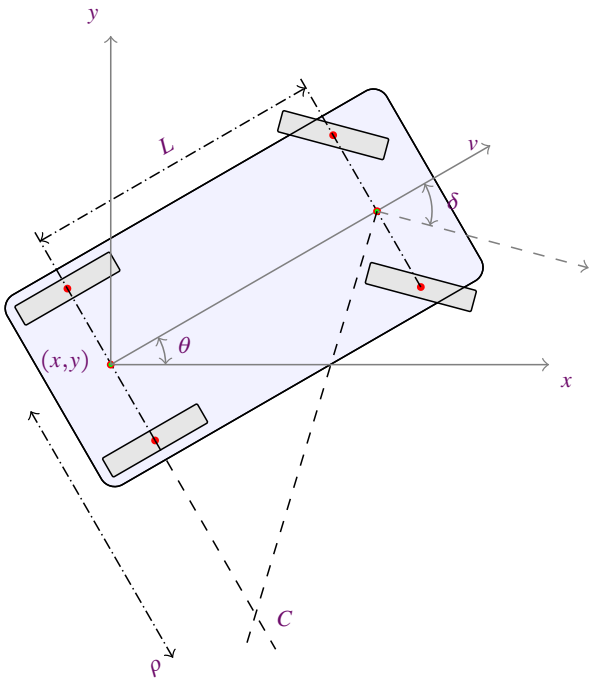

Fig. 2: Agricultural vehicle.

We deduce some facts about the vehicle's motion. The geometric interpretation of the derivative of a function $f(x)$ at $x = x_0$ is the slope of the graph of f at that point. This is defined to be the slope of the tangent line to the graph at x_0.[7] Thus, by definition we have:

$$\begin{aligned} & \frac{dy}{dx} = \tan\theta \\ = & \quad \left\{ \text{ by the chain rule of differentiation: } \frac{dy}{dt} = \frac{dy}{dx} \cdot \frac{dx}{dt} \right\} \\ & \frac{dy/dt}{dx/dt} = \tan\theta \\ = & \quad \{ \text{ changing from Leibniz's notation to Newton's for conciseness } \} \end{aligned}$$

derive this geometry in a mechanical engineering setting [33]). This approximates idealised steering along a circular arc.

[7] More formally, it is the limit of the secant lines through $(x_0, f(x_0))$ and nearby points $(x, f(x))$ as x approaches x_0. For the tangent line to be well-defined, the graph of f at x_0 must be continuous. The tangent line must not be vertical: a vertical line is not a function and so does not have a slope.

$$
\begin{aligned}
& \dot{y}/\dot{x} = \tan\theta \\
= & \quad \{ \text{ trigonometry: } \tan\theta = \sin\theta/\cos\theta \ \} \\
& \dot{y}/\dot{x} = \sin\theta/\cos\theta \\
= & \quad \{ \text{ arithmetic } \} \\
& \dot{y}\cos\theta = \dot{x}\sin\theta
\end{aligned}
$$

This equation is satisfied by $\dot{x} = \cos\theta$ and $\dot{y} = \sin\theta$ and their scalar multiples, which correspond to the vehicle's velocity v. That means $\dot{x} = v\cos\theta$ and $\dot{y} = v\sin\theta$.

Now we want to find the angular speed of the vehicle. The angle between the lines that meet at C is also δ. The distance between the centre of the axle and C is ρ (see Figure 2). Therefore,

$$
\begin{aligned}
& \tan\delta = L/\rho \\
= & \quad \rho = L/(\tan\delta)
\end{aligned}
$$

The vehicle follows a circular path. Suppose that the vehicle travels s metres along its circular trajectory and s is half the distance it takes to intersect the $y = x$ axis again. Given that θ is the angle subtended by this arc, identical to the one measured at the centre of the rear axle, and ρ is the radius of this circular trajectory, then we have $s = \rho\theta$. This comes directly from the formula that relates degrees of arc and radians: angle $\theta = \dfrac{s}{\rho}$ radians. Now calculate:

$$
\begin{aligned}
& s = \rho\theta \\
\Rightarrow & \quad \{ \text{ differentiate both sides (assumption: monotonic functions) } \} \\
& \frac{ds}{dt} = \frac{d(\rho\theta)}{dt} \\
= & \quad \{ \text{ constant circle radius } \rho \ \} \\
& \frac{ds}{dt} = \rho\frac{d\theta}{dt} \\
= & \quad \left\{ \text{ replacing linear velocity } \frac{ds}{dt} = v \text{ and angular velocity } \frac{d\theta}{dt} = \dot{\theta} \right\} \\
& v = \rho\dot{\theta}
\end{aligned}
$$

To summarise, the kinematic equations of our simplified vehicle are:

$$
\begin{aligned}
\dot{x}(t) &= v(t)\cos\theta(t) \\
\dot{y}(t) &= v(t)\sin\theta(t) \\
\dot{\theta}(t) &= (v/L)\tan\delta(t)
\end{aligned}
\tag{1}
$$

where the controlled inputs are t and $\delta(t)$, and the initial values for the states x, y, θ are known.

3.2 Controller

To keep the explanation simple, we develop a controller whose purpose is to drive the vehicle in a straight line, in the x direction with a constant speed.

Given a position (x, y) and orientation of the vehicle θ, the controller computes δ and v that get the vehicle closer to the intended y coordinate $y_{target} = 0$ and orientation $\theta_{target} = 0$, as follows:

$$\delta_{error}(t) = -(k_t\theta(t) + k_p y(t))$$

$$\delta(t) = \begin{cases} \delta_{max} & \text{if } \delta_{error}(t) > \delta_{max} \\ -\delta_{max} & \text{if } \delta_{error}(t) < -\delta_{max} \\ \delta_{error}(t) & \text{otherwise} \end{cases} \tag{2}$$

$$v(t) = 1$$

where $k_t > 0$ and $k_p > 0$ are tunable constants and δ represents the steering angle limited by the maximum steering wheel angle, represented by δ_{max}. Figure 3 shows an example solution of the closed (no inputs) system formed by putting together Eqs. (1) and (2).

3.3 Deployed System

The system of equations derived so far in Eqs. (1) and (2) does not reflect the actual deployment of the system. This is because: (i) the deployed controller will sample the position and orientation of the vehicle every $H > 0$ seconds; and (ii) there is noise in the sensor measurements of x, y, θ and in the actuators v, δ.

To represent the sampled system, we note that, between samples, the actuation of the controller is constant, and the only continuous behaviour is the vehicle's. In the time interval $\tau \leq t < \tau + H$, the system's motion is given by Eqs. (1) and (2) with the difference that the vehicle speed and steering angle are kept constant throughout the interval, and are computed according to the orientation and position of the vehicle at the beginning of the interval:

$$\begin{aligned} \dot{x}(t) &= v(\tau)\cos\theta(t) \\ \dot{y}(t) &= v(\tau)\sin\theta(t) \\ \dot{\theta}(t) &= (v(\tau)/L)\tan\delta(\tau) \end{aligned} \tag{3}$$

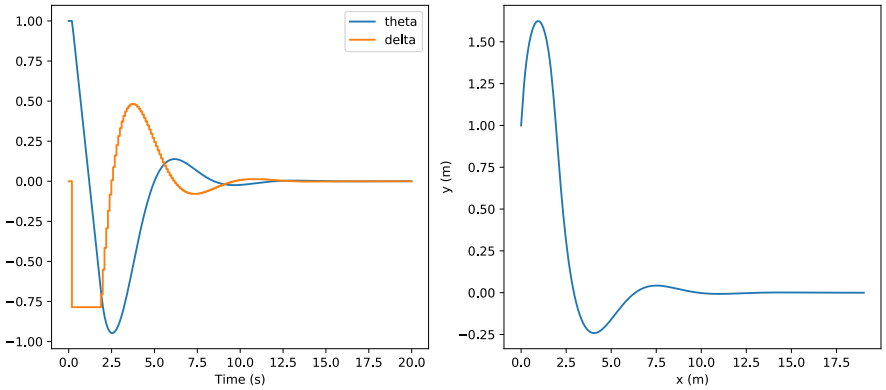

Fig. 3: Example simulation where vehicle starts 1 m away from the straight line (right), and with wrong orientation (left). As can be seen, the controller corrects this (saturating δ initially) and settles the vehicle in the straight line after a few seconds. Here, the parameters $k_t = k_p = 1$, and theta and delta correspond to θ and δ.

where the initial state values $x(\tau), y(\tau), \theta(\tau)$, are given, and $\delta(\tau), v(\tau)$ are computed as in Eq. (2) (or Eq. (4) if there's noise) from the initial state values. Note that throughout the interval, $\dot{\theta}(t)$ is constant.

The behaviour of the system is computed in a co-simulation involving the controller and the vehicle kinematics as follows:

1. At time $t = \tau$, the controller gets the values for position and orientation from the sensors, calculates the steering wheel angle and speed, and sets those values through the actuators.

2. We use a numerical solver to solve Eq. (3) in the interval $\tau \leq t \leq \tau + H$.

3. At the end of the interval, when $t = \tau + H$, the controller gets new values for position and orientation from the sensors, recalculates the steering wheel angle and speed, and sets those values through the actuators (this is Step 1 for the new interval $\tau + H \leq t \leq \tau + 2H$), and the cycle is repeated.

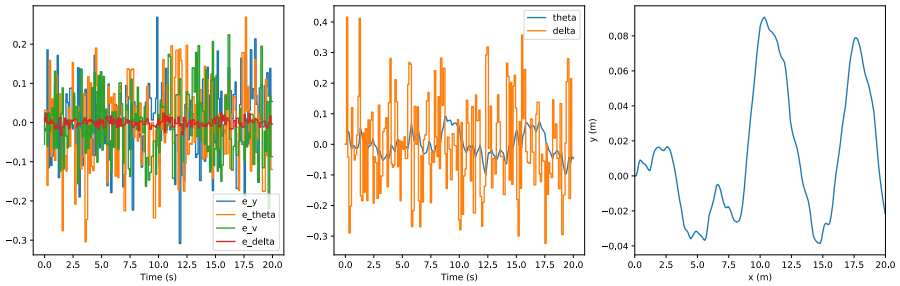

Fig. 4: Results of the co-simulation using the controller with noise (Eq. (4)) and the system dynamics in Eq. (3). The noise terms are drawn from statistical distributions. The communication step size $H = 0.1$ seconds.

To account for noise, we introduce noise terms by reformulating Eq. (2) as follows:

$$\delta_{error} = -(k_t(\theta + \epsilon_\theta) + k_p(y + \epsilon_y)) + \epsilon_\delta$$

$$\delta = \begin{cases} \delta_{max} & \text{if } \delta_{error} > \delta_{max} \\ -\delta_{max} & \text{if } \delta_{error} < -\delta_{max} \\ \delta_{error} & \text{otherwise} \end{cases} \tag{4}$$

$$v = 1 + \epsilon_v$$

We assume that the noise terms introduced are constant during the continuous evolution of the system's kinematics, but may change at each control execution cycle. The system behaviour is computed by the same co-simulation algorithm as described above, except when the controller computes δ, v, it does so using new values for the noise terms. The values for these terms can be taken from a statistical distribution. They will perturb the system and cause deviations on its correct behaviour, even when the vehicle starts in the target position and orientation. The simulation results in Fig. 4 illustrate an example.

3.4 Statistical Analysis

In the previous subsections, we derived a simple model of how the deployed agricultural vehicle behaves. In the following subsections, we introduce the statistical analysis used to quantify the uncertainty of the vehicle's behaviour with respect to the uncertainty in the environment of the vehicle. The methods we introduce here can be applied automatically for different parametrisations of the

system, and therefore they can be used to guide design choices and parameter tuning.

To illustrate the importance of these analyses with respect to safety, suppose that we wish the system to always stay within a maximum distance of the center of road.

$$\forall\, t, Y(t) < y_{max} \tag{5}$$

where Y denotes the trace of y coordinates of the system, and y_{max} is a constant reflecting, for example, the half-width of the road where the vehicle will drive. The more general property $\forall\, t, |Y(t)| < y_{max}$ could be used, but the analyses are analogous.

Assuming that the noise in the system can be characterised as a statistical distribution, then running Montecarlo co-simulations may help us determine the expected system's behaviour with respect to its safety properties.

For instance, suppose that each time the controller computes the velocity and steering angle, in Eq. (4), the noise values are drawn from a Normal distribution with zero mean and standard deviation that's been estimated from real measurements:

$$\epsilon_p \backsim \mathcal{N}(0, \sigma_p) \text{ for } p \in \{\theta, y, \delta, v\} \tag{6}$$

Then running many simulations, as Fig. 5 exemplifies, will yield estimates on important measures of expected violations. Those measures can then be used in a runtime monitor that checks whether the violations of the real system agree with the simulation data. To exemplify this, we first need to define the quantities of interest for the case study used.

Definition 1 (Violation Events, Peak, and Duration). *Given a trajectory of the vehicle $Y(t)$, the i-th violation event, denoted as a tuple with the start time and end time $[\underline{t}_i, \bar{t}_i]$, represents the interval during which the vehicle is violating that property. Formally, $\forall\, t \in [\underline{t}_i, \bar{t}_i], Y(t) \geq y_{max}$. The peak of the i-th violation event is given by $\max\{Y(t) - y_{max} : t \in [\underline{t}_i, \bar{t}_i]\}$. The maximum peak of a trace is the maximum of the peak of each violation event.*

Figure 6 illustrates these concepts.

Within the same trace, violation events are not Independent and Identically Distributed (i.i.d.), because of the dynamics of the vehicle. To see why, note that, right after a violation event, the probability of another event occurring depends on the dynamics of the control system. In particular, the control system is steering the vehicle towards the line $y = 0$, regardless of the values that the noise terms may take (the noise terms are i.i.d.). The further the vehicle is from the line, the stronger the control action, and the smaller the set of possible noise

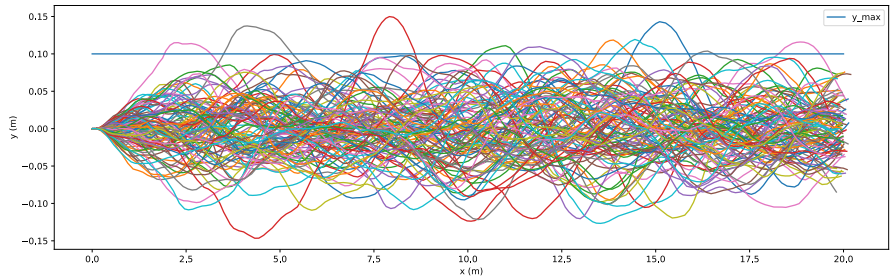

Fig. 5: Results of 100 co-simulations using the controller with noise (Eq. (4)) and the system kinematics in Eq. (3). The noise terms are drawn from statistical distributions with the following standard deviations: $\sigma_\theta = \sigma_y = \sigma_v = 0.1, \sigma_\delta = 0.01$. The communication step size matches the controller execution cycle $H = 0.1$ seconds. The straight line represents the property limit $y_{max} = 0.1$. As can be seen, there are a few violations.

terms that will make the vehicle violate the property. This set gets bigger as the control action gets weaker, e.g., as the vehicle gets closer to the line $y = 0$, affecting the probability of a subsequent violation event.

However, between two traces, the violation events are i.i.d.. Hence, we can compute statistics between traces for:

- Maximum violation peak within the same trace.
- Maximum violation duration.

Figure 7 shows the results of these quantities, computed from simulations like the ones in Fig. 5.

3.5 Runtime Monitoring

Having estimated the quantities in Fig. 7, a run-time monitor can be synthesised that will perform the following, at each monitoring step:

1. Compute moving average of each sensor and actuation signal.
2. Compute moving standard deviation of each sensor and actuation signals (such standard deviations represent the noise standard deviations observed).
3. Compare each observed noise standard deviation with the noise standard deviation used to obtain the simulation results.
4. If the observed noise parameters are too different from the noise parameters used to create the simulations, give an alert to the user (because the simulation results used the wrong assumptions and should be repeated).
5. Quantify how safe the system is (e.g., evaluate Eq. (5), compute for how long it does not hold, maximum violation, etc.).

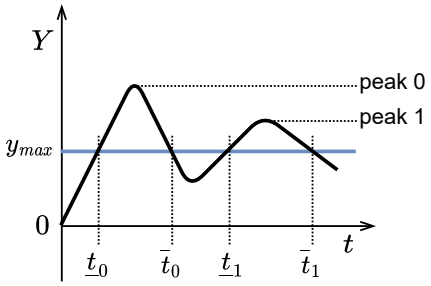

Fig. 6: Illustration of the concepts in Definition 1. There are two violation events. The peak of each violation event is the maximum distance between the trace and the safety line y_{max}. The maximum peak is peak 0.

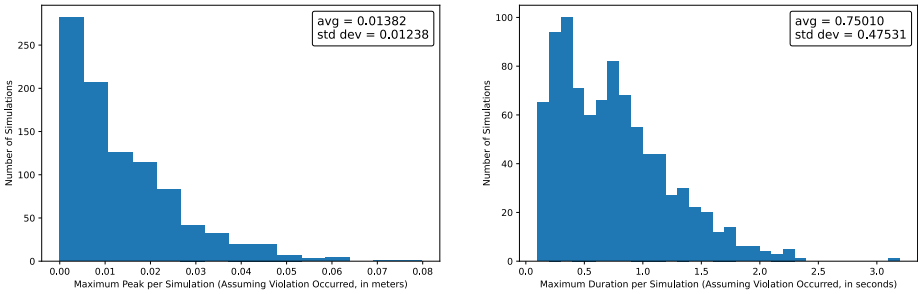

Fig. 7: Statistics of quantities of interest for runtime monitoring, computed from 10000 co-simulations. The parameters are the same as in Fig. 5.

6. If such quantification does not match what was observed with the simulation data (in Fig. 7), then alert the user that the system is unsafe.

3.6 What-if Analysis

The property in Eq. (5) reflects a characteristic of the road in which the vehicle operates. These properties may come from industrial regulations, etc., and are independent of the way the vehicle is designed. The previous subsection shows how to characterise and quantify the violation events caused by the vehicle and control system characteristics, in particular, the noise in the sensors and actuators. This subsection shows how the same technique can be used to identify which noise sources have a bigger impact in the system operation.

Suppose we are trying to decide whether to upgrade the vehicle with a better GPS receiver, or a better servomotor for the steering wheel. With only enough

budget for one purchase, we would like to know which makes the system safer. Repeating the statistical analysis with the new noise characteristics provides estimates on the impact of each configuration.

For brevity, we show the results and conclusions of the statistical for both new GPS configuration and new servo, in Fig. 8.

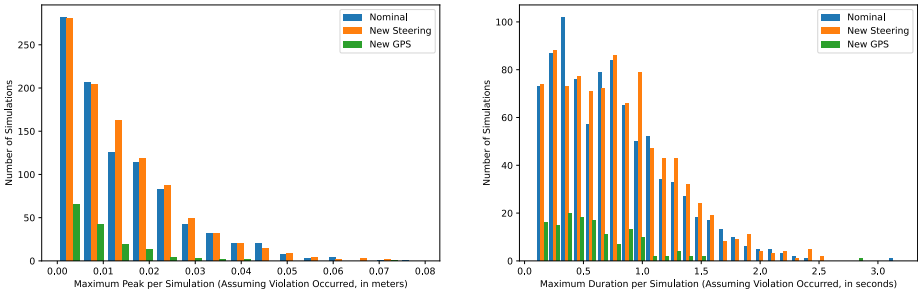

Fig. 8: What-if analysis results. The nominal standard deviations for the noise terms are as in Fig. 5: $\sigma_\theta = \sigma_y = \sigma_v = 0.1, \sigma_\delta = 0.01$. As can be seen, the new GPS configuration reduces both the mean and standard deviation of the violation events more than the new servo configuration.

4 Conclusion

We set out to describe some of the techniques that can be used to measure discrepancies between a CPS and a digital twin. We have shown how, for a given safety property Eq. (5), to quantify and characterise the violation events of that property. The analysis is based on a model of the system, and can then inform a runtime monitor that checks whether the noise and violations observed fall within the expected statistical distributions. For this analysis, we assume that the noise, initial states, and inputs to the system must be such that violation events can occur. If the system is already safe with respect to the property, regardless of the noise, then the analyses presented here confirm that and there's nothing to be tolerated.

Additionally, the results presented here allow the engineers to determine the best system configuration through what-if analysis. For instance, the statistical analysis could be used to calculate the minimum accuracy of the GPS signal, so that the probability of a violation event occurring is below some threshold.

We now summarise the limitations and assumptions of this case study:

1. The statistical analysis can be computationally expensive (even though the simulations can be run in parallel).

2. The analyses presume that the standard deviations of the noise can be obtained from empirical data. They are supposed to make use of the models to relieve but not eliminate the need for some physical experimentation with the real vehicle.
3. In each simulation, we assume that the noise terms are constant in between controller samples. In practice, noise may affect the physical system in between those samples.
4. The performance of the co-simulation plays a fundamental role in enabling the Montecarlo co-simulations.

Additionally, we have assumed that it is possible to quantify the violation degree of a safety property. Signal Temporal Logic (STL) is a formalism for specifying the requirements of CPSs [7]. It mixes models of discrete and analogue components and a continuous environment. STL has a quantitative semantics and is a natural candidate to quantify safety. Algorithms exist for offline computation of quantitative semantics and methods for online analysis monitoring satisfaction during simulation [6].

Our work is related to the field of run-time verification. These techniques checks that a run of a system satisfies or violates a given correctness property [21]. This is the natural technology for checking behavioural deviations between a physical asset and its digital twin. Techniques exist for robust online monitoring of properties described in STL [6].

Regarding the performance of the co-simulation, if the example is sufficiently complex, it won't be feasible to run the 10 000 co-simulations. To this end, researchers have employed many system identification [17], grey-box modelling [29], and surrogate modelling techniques [12], to create faster simulation models that can be used for the analysis.

Ongoing work is focusing on generalising this method into a framework that allows user to describe the safety properties, described tolerable violations, and generate runtime monitors that enforce such tolerances. In particular, we will develop an ontology based on multiple case studies, which will in turn inform us of the most common tolerable violations for safety properties.

Acknowledgements. We acknowledge the European Union for funding the INTO-CPS project (Grant Agreement 644047), which developed the open tool chain and the INTO-CPS Application; the Poul Due Jensen Foundation that funded subsequent work on taking this forward towards the engineering of digital twins; and the European Union for funding the HUBCAP project (Grant Agreement 872698). We acknowledge support from the UK EPSRC for funding for the RoboCalc (EP/M025756/1) and RoboTest projects (EP/R025479/1). Finally, we acknowledge support from the Royal Society and National Natural Science

Foundation of China for funding for the project Requirements Modelling for Cyber-Physical Systems IEC/NSFC/170319. Early versions of the ideas in this paper were presented to the Digital Twin Centre in Aarhus in December 2019 (twice) and to the RoboStar team in York in January 2020. We are grateful for their feedback.

References

1. Alur, R.: Principles of Cyber-Physical Systems. The MIT Press (2015)
2. Bartocci, E., Falcone, Y., Francalanza, A., Reger, G.: Introduction to Runtime Verification. In: Bartocci, E., Falcone, Y. (eds.) Lectures on Runtime Verification, vol. 10457, pp. 1–33. Springer International Publishing, Cham (2018)
3. Box, G.E.P.: Robustness in the strategy of scientific model building. In: Launer, R.L., Wilkinson, G.N. (eds.) Robustness in Statistics, pp. 201–236. Academic Press (1979)
4. Box, G.E.P., Draper, N.R.: Empirical Model-Building and Response Surfaces. Wiley (1987)
5. Cassar, I., Francalanza, A., Aceto, L., Ingólfsdóttir, A.: A Survey of Runtime Monitoring Instrumentation Techniques. Electronic Proceedings in Theoretical Computer Science 254, 15–28 (Aug 2017)
6. Deshmukh, J.V., Donzé, A., Ghosh, S., Jin, X., Juniwal, G., Seshia, S.A.: Robust online monitoring of signal temporal logic. Formal Methods Syst. Des. 51(1), 5–30 (2017)
7. Donzé, A.: On signal temporal logic. In: Legay, A., Bensalem, S. (eds.) RV 2013: 4th International Conference on Runtime Verification, Rennes, 24–27 September 2013. Proceedings. Lecture Notes in Computer Science, vol. 8174, pp. 382–383. Springer (2013)
8. Falcone, Y., Krstić, S., Reger, G., Traytel, D.: A Taxonomy for Classifying Runtime Verification Tools. In: Colombo, C., Leucker, M. (eds.) Runtime Verification, vol. 11237, pp. 241–262. Springer International Publishing, Cham (2018)
9. fitzgerald, j., Gamble, C., Larsen, P.G., Pierce, K., Woodcock, J.: Cyber-Physical Systems Design: Formal Foundations, Methods and Integrated Tool Chains. In: Formal Methods in Software Engineering (FormaliSE), 2015 IEEE/ACM 3rd FME Workshop On. pp. 40–46 (2015)
10. Fitzgerald, J.S., Larsen, P.G., Pierce, K.G.: Multi-modelling and co-simulation in the engineering of cyber-physical systems: Towards the digital twin. In: ter Beek, M.H., Fantechi, A., Semini, L. (eds.) From Software Engineering to Formal Methods and Tools, and Back - Essays Dedicated to Stefania Gnesi on the Occasion of Her 65th Birthday. Lecture Notes in Computer Science, vol. 11865, pp. 40–55. Springer (2019)
11. Foldager, F., Larsen, P.G., Green, O.: Development of a Driverless Lawn Mower using Co-Simulation. In: 1st Workshop on Formal Co-Simulation of Cyber-Physical Systems. Trento, Italy (September 2017)
12. Forrester, A., Sobester, A., Keane, A.: Engineering Design via Surrogate Modelling: A Practical Guide. John Wiley & Sons (2008)
13. Gibson, J.P., Larsen, P.G., Pantel, M., Fitzgerald, J.S., Woodcock, J.: Cyber-physical systems engineering: An introduction. In: Margaria, T., Steffen, B. (eds.) ISoLA 2018: 8th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation. Distributed Systems. Limassol, Cyprus, 5–9 November 2018, Proceedings, Part III. Lecture Notes in Computer Science, vol. 11246, pp. 407–410. Springer (2018)
14. 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)

15. INTOCPS Association: Uncertainty quantification repository (2020), https://gitlab.au.dk/clagms/2020.isola.uncertaintyquantification, accessed 21th December 2020
16. Jantsch, A., Sander, I.: Models of computation and languages for embedded system design. IEE Proceedings - Computers and Digital Techniques 152(2), 114–129(15) (2005)
17. Keesman, K.J.: System Identification: An Introduction. Springer Science & Business Media (2011)
18. Kritzinger, W., Karner, M., Traar, G., Henjes, J., Sihn, W.: Digital twin in manufacturing: A categorical literature review and classification. IFAC-PapersOnLine 51, 1016–1022 (01 2018)
19. Larsen, P.G., Fitzgerald, J., Woodcock, J., Gamble, C., Payne, R., Pierce, K.: Features of integrated model-based co-modelling and co-simulation technology. In: Bernardeschi, Masci, Larsen (eds.) 1st Workshop on Formal Co-Simulation of Cyber-Physical Systems. LNCS, Springer-Verlag, Trento, Italy (September 2017)
20. Legaard, C.M., Gomes, C., Larsen, P.G., Foldager, F.F.: Rapid Prototyping of Self-Adaptive-Systems using Python Functional Mockup Units. p. to appear. SummerSim '20, ACM New York, NY, USA (2020)
21. Leucker, M., Schallhart, C.: A brief account of runtime verification. J. Log. Algebr. Program. 78(5), 293–303 (2009)
22. Macedo, H., Nilsson, R., Larsen, P.: The Harvest Coach Architecture: Embedding Deviation-Tolerance in a Harvest Logistic Solution. Computers 8(2) (April 2019)
23. Martínez, G.S., Karhela, T., Vyatkin, V., Miettinen, T., Pang, C.: An OPC UA based architecture for testing tracking simulation methods. In: 2015 IEEE Trustcom/BigDataSE/ISPA. vol. 3, pp. 275–280 (2015)
24. Mitsch, S., Platzer, A.: ModelPlex: Verified runtime validation of verified cyber-physical system models. Formal Methods Syst. Des. 49(1-2), 33–74 (2016)
25. Nakaya, M., Li, X.: On-line tracking simulator with a hybrid of physical and Just-In-Time models. Journal of Process Control 23(2), 171–178 (2013)
26. Rajamani, R.: Vehicle Dynamics and Control. Springer Science & Business Media (2011)
27. Rajhans, A., Bhave, A., Ruchkin, I., Krogh, B.H., Garlan, D., Platzer, A., Schmerl, B.: Supporting Heterogeneity in Cyber-Physical Systems Architectures. IEEE Transactions on Automatic Control 59(12), 3178–3193 (Dec 2014)
28. Rooney, J.J., Heuvel, L.N.V.: Root cause analysis for beginners. Quality progress 37(7), 45–56 (2004)
29. Sohlberg, B., Jacobsen, E.: GREY BOX MODELLING – BRANCHES AND EXPERIENCES. IFAC Proceedings Volumes 41(2), 11415–11420 (2008)
30. Tao, F., Zhang, H., Liu, A., Nee, A.Y.C.: Digital Twin in Industry: State-of-the-Art. IEEE Trans. Ind. Inf. 15(4), 2405–2415 (2019-04)
31. 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 (2013)
32. Vangheluwe, H.: Foundations of Modelling and Simulation of Complex Systems. Electronic Communications of the EASST 10 (2008)
33. Zhao, J.S., Liu, Z.J., Dai, J.: Design of an Ackermann type steering mechanism. Journal of Mechanical Engineering Science 227, 2549–2562 (11 2013)