PINNball Wizard: Conjuring Digital Twins with Physics-Informed Neural Networks

Ted McRobie
Thales UK
Reading, Berkshire
ted.mcrobie@uk.thalesgroup.com

ABSTRACT

Machine Learning (ML) models have been adopted for simulation due to increased efficiency, development savings, and reduced computational requirements in production. However, there is a resistance to adopting ML due to a lack of trust in non-deterministic results that some of these algorithms produce. Crafting Digital Twins (DTs) for Defense applications is no exception, where computationally expensive Maths and Physics (MaP) based models are used to control virtual versions of real assets. Therefore, a careful blend of the nuanced understanding provided by Subject Matter Experts (SMEs) with the adaptive learning capabilities of data-driven ML solutions would be ideal to create DTs.

Physics-Informed Neural Networks (PINNs) are capable of producing dynamic models which can surpass MaP simulation performance. Similar to ML, PINNs learn from historical data to capture unique behaviours of assets as opposed to the ideal conditions MaP simulators assume. This is tempered by the influence of SMEs through physical equations which PINNs utilize as a guide in areas of the domain space where there is no training data, as well as a benchmark for data trustworthiness.

These dynamics models can be utilized by human operators or Reinforcement Learning (RL) algorithms in autonomous control systems. This approach to creating DTs presents a transformative solution to rapid and cost-effective development of next-generation integrated training and readiness applications, tailored to contemporary threat environments, assuring deterrence.

This paper explores what PINNs are and their utility in creating Digital Twins. We demonstrate a practical example by training a PINN on telemetry logs of an in-service Unmanned Aerial System (UAS) and SME knowledge to create a flight dynamics model. We find PINNs produce predictions with a 57.01% reduction in Mean Absolute Error compared to Neural Networks, which rises to 67.01% in out-of-distribution predictions.

ABOUT THE AUTHORS

Ted McRobie graduated from the University of Portsmouth with a B.Sc.(Hons) in Physics, Astrophysics, and Cosmology and went on to graduate from the University of St Andrews with a M.Sc. in Astrophysics. During his time at University he completed projects on Theoretical Astrophysics, developing novel methodologies to model and simulate the first Stars. He is a Machine Learning Scientist at Thales Training and Simulation working on Artificial Intelligence topics. In his spare time he develops Machine Learning algorithms to predict asteroid orbits and impacts.

PINNball Wizard: Conjuring Digital Twins with Physics-Informed Neural Networks

Ted McRobie
Thales UK
Reading, Berkshire
ted.mcrobie@uk.thalesgroup.com

INTRODUCTION

Since their anticipation (Gelernter, 1991) and consequent commercial introduction in 2002 (Grieves, 2016), Digital Twins (DTs) have emerged as a disruptive technology with profound implications across various industries, offering a virtual mirror of physical reality for iteration, analysis, and optimization. However within the defense and simulation sectors, the implementation of DTs promises to revolutionize traditional paradigms of operational planning, training, and asset management. As opposed to siloed DTs in closed loop simulations, entire Synthetic Environments (SEs) with a smörgåsbord of interconnected and interactive DT entities are being designed to model battlespaces globally to unprecedented fidelity levels (Budning, Wilner, and Cote, 2022). Examples of these complex tools are being developed rapidly in industry (Smith, 2023 and Catapult, 2024), and thus shows market interest in these tools and their swift development.

SEs and the DTs they host are incredibly complex, and therefore require significant investments both fiscally and resourcefully - exponentially more so when developed in a short space of time. Machine Learning (ML) offers a compelling rationale for creating DTs in defense owing to its unparalleled capacity to deliver realism, scalability, agility, and expedited development cycles. These algorithms can replicate complex real-world scenarios with remarkable fidelity (Krishnababu, Valero, and Wells, 2021), thus enabling defense practitioners to simulate and analyze various operational scenarios with a high degree of accuracy within DTs (Kapteyn and Willcox, 2020). The ML development cycle facilitates rapid iteration and continuous development of DTs which would allow them to adapt and incorporate new data seamlessly, thereby ensuring their relevance and effectiveness in dynamic military environments. On larger scales this capability enhances training and simulation exercises and empowers decision-makers with invaluable insights for optimizing resource allocation, strategic planning, and mission execution. In an ever-evolving threat landscape, this represents a strategic imperative offering unprecedented realism and agility to support mission-critical objectives. However the scope of this article is limited to showcasing how these techniques can be utilised to deliver individual, high-fidelity DTs.

Physics-Informed Neural Networks (PINNs) are a disruptive technology capable of enhancing simulation capabilities and product development across a range of disciplines. Representing a core part of cutting-edge DT development, this formidable ML technique offers a potent synergy between data-driven methodologies and unassailable physical principles.

Digital Twins

The precise definition of DTs is inconsistent between industries and companies. Consequently, it is paramount that we establish what we mean when referring to "a Digital Twin" due to its significance in product development (Argolini, Bonalumi, and Deichmann, 2023) and bedrock relationship in this article. When referring to a Digital Twin in this article, we mean a highly accurate virtual representation of a physical entity recreated in a synthetic environment in real-time, faster than real-time, or simulated. They are used explicitly - but not exhaustively - for purposes such as tactical development, mission rehearsal, command and control, or testing and evaluation. The North Atlantic Treaty Organization (NATO) also describes a symbiotic relationship between a DT and its physical counterpart. There is no reason why DTs created according to the methodology later established in this article cannot be used in this way, but as we look at the context of generating DTs this element of its definition is disregarded.

Machine Learning and Neural Networks

Neural networks (NNs) are ML models inspired by the structure and function of the human brain. In the context of defense, they are powerful tools used for tasks ranging from image recognition and natural language processing to decision-making and autonomous systems. Essentially, NNs learn from data to recognize patterns, make predictions, and even adapt to changing conditions. This makes them invaluable for enhancing military capabilities such as target identification, cybersecurity, and - the topic of this article - unmanned systems.

On a more technical level, we can approximate how we have used NNs by describing them as complex regression algorithms or "function-fitters". Regression involves predicting numerical values based on input data, allowing for tasks like estimating weapon effectiveness or predicting equipment maintenance needs. Function fitting is the process of constructing a curve to fit to a set of data points, today we employ algorithms to do this which are then tasked in modelling behaviour or optimizing resource allocation. These capabilities empower defense applications to make informed decisions and enhance operational effectiveness.

The defense industry's cautious approach to the production and utilization of ML stems from several factors:

- First, there is a concern about reliability. Unlike traditional software with clear rules and transparent decision-making processes, NNs can be opaque[1]. This makes it challenging to understand how conclusions are drawn. This lack of interpretability raises questions and instigates further research regarding trustworthiness (Huang et al., 2020), especially in high-stakes scenarios where there is a risk to human life or national security.

- Secondly, there is apprehension about the potential for bias in ML algorithms. If not properly mitigated, biases present in training data can perpetuate and amplify errors in Artificial Intelligence (AI) systems. This can lead to unfair or discriminatory outcomes which are unacceptable in defense operations due to the ethical and legal ramifications of these decisions (Alcántara Suárez and Monzon Baeza, 2023). This is compounded by the rapid pace of technological advancement in AI. This is a key concern for the domain of simulation as it is key to not give AI any advantage over human operators when attempting to replicate decisions which are at their core, human.

- Finally, there is a substantial "gray area" when training AI models due to data ownership. This raises ethical and legal concerns on Intellectual Property (Žukovs and Upenieks, 2023).

While recognizing the potential benefits of ML and NNs, we proceed cautiously through the use of DTs developed with ML, prioritizing testing, validation, and transparency to ensure reliability, security, and ethical use of these technologies.

Physics-Informed Neural Networks

Recently invented PINNs (Raissi, Perdikaris, and Karniadakis, 2017), combine the flexibility of NNs with mathematics and physics to improve reliability and trustworthiness in modelling complex real systems, such as aerodynamics and projectile motion - topics covered later in this article.

Unlike traditional NNs which learn patterns and relationships solely from data resulting in potential exposure to bias, known physics principles are integrated into PINNs. This integration ensures that the network learns from the underlying physics of the problem while retaining the realism captured within training data. By comparing solutions to partial differential equations (PDEs) describing the physical world to the derivative of a prediction with respect to inputs during the training process, PINNs can effectively leverage both data-driven insights and fundamental physical principles, thereby improving accuracy and reliability.

Using PINNs (and derivations of them) offer several advantages over traditional methods:

1. **Physical Consistency:** Through guidance with physical laws, PINNs can produce predictions that are more representative of known principles. This reduces the risk of generating unrealistic or erroneous results. This is

[1]This is a common misconception as many ML methodologies allow for rigorous testing and validation, the popular exception to this is Generative AI, however with the rise of explainable AI (XAI, Das and Rad, 2020) these models are becoming more transparent.

robust to the extent that these techniques are being integrated with robotics (Ramesh and Ravindran, 2023) - a field renowned to be a challenging use-case of ML due to fidelity-level and approximation complexity.

2. **Improved Generalizations:** Integrating physics constraints helps prevent overfitting and improves the generalization ability of the NN (Nabian and Meidani, 2018; Raymond and Camarillo, 2021). This predominantly assists predictions in the domain space where there has been limited or no collected data, or where data contains noise such that predictions are prone to bias and are mislead to inaccurate predictions.

3. **Interpretability:** PINNs provide greater transparency and interpretability, as the inclusion of real physics equations allows users - more specifically Subject Matter Experts (SMEs) - to understand and validate reasoning behind PINN model predictions (Nabian and Meidani, 2018; Ramabathiran and Ramachandran, 2021; Wang et al., 2023).

As explored in this article, PINNs offer a promising approach for developing DTs providing more trustworthy and reliable predictions combining data-driven ML with the fundamental laws of physics.

METHODOLOGY

In the realm of developing Digital Twins, the process of creating representative flight behavior models proves to be an intricate and resource-intensive endeavor. Each model is tailored to a specific platform, requiring SME resources with a deep understanding of both the underlying physical principles and the intricacies of the aircraft being twinned. To address these challenges, we propose the utilization of PINNs as an innovative approach to expedite the creation of accurate DT flight models. By leveraging this advanced algorithm framework, we can achieve rapid model development without compromising on accuracy.

In this study, we curate 77 historical flight telemetry logs obtained from an Unmanned Aerial System (UAS) drone. These datasets, combined with detailed knowledge of aerodynamics, serve as the foundation for training the PINN algorithm. We then present the outcomes of this endeavor as a DT interacting with a synthetic environment as it would be intended in production. Through this demonstrative showcase, we aim to highlight the potential of PINNs in revolutionizing the process of creating DT behaviour models, paving the way for enhanced efficiency and accuracy in simulations.

Concept Demonstration

A simple demonstration of the effectiveness of PINNs is through a simple two-dimensional projectile motion prediction algorithm based on the well-regarded Runge-Kutta (RK) numerical approximation method.

Projectile motion, a fundamental concept in classical mechanics, describes the trajectory of an object under the influence of few fundamental forces and initial conditions. Traditional numerical methods have been widely employed to solve equations of motion for projectile systems with an iterative, theoretical approach (Society, 2012). One commonly used approach is the fourth-order RK method (El-Fouly and Misra, 1918; Butcher, 2015)), where the slope of projectile trajectory is calculated a number of times over a known step-size (h) and an average of these slopes are used to calculate projectile trajectory state at the next time-step (s_{n+1}), as described by Equation 1 and Figure 1.

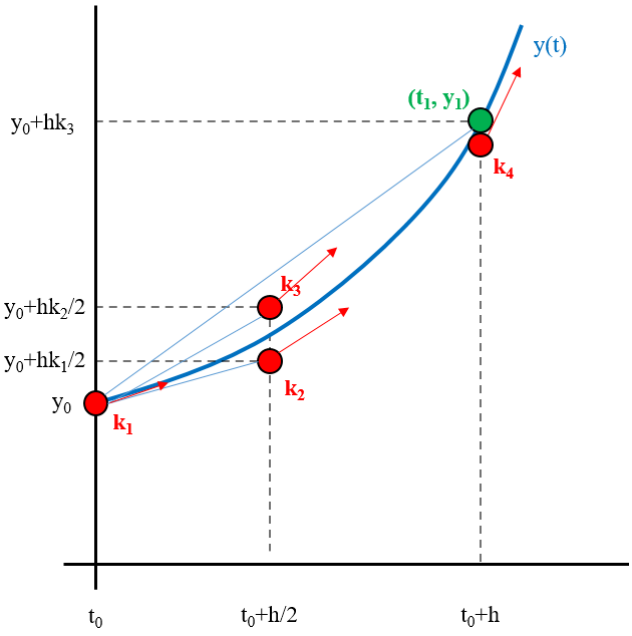

Figure 1: Illustration of the Runge-Kutta method to calculate the projected positions along a curve from (t_0, y_0) to (t_1, y_1).

$$\vec{s_{n+1}} = \vec{s_n} + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{1}$$

Why go to the great expense to create synthetic data (or as we see later in this article, acquire platform data) to train a ML model? The rationale is that these numerical solutions are not necessarily appropriate for these styles of problems, especially with the growing need for faster than real-time simulation for tasks such as operational analysis. Despite being so foundational that these methods are ingrained in defense development standards (Society, 2012), there are severe limitations to these numerical approximation:

- Computational expense at high fidelity levels.
- Limited scaleability - as scenarios become more complex, they can become exponentially more complex to develop.
- Exponential growth in error as predictions extend farther into the future[2]. To be precise, due to it being a fourth-order method this suffers from local-truncation error - which would be of the order $\mathcal{O}(n^5)$ and the total accumulated error is of the order $\mathcal{O}(n^4)$.
- Difficulty in implementation given complex scenarios with varying forces and irregular geometries.

For 2-dimensional projectile motion, the gradients (k_n) are calculated by using fundamental equations of motion (Galilei, 1914, Whelan and Hodgson, 1978 to approximate the velocities and accelerations at the discretised points in each timestep. We create synthetic data through the explicit use of Euler's method (Barnett, 2009), and we extract a sample from this. To give a realistic representation of what data collected looks like; noisy, discontinuous, and disparate, we add gaussian noise and limit data collection to just 10 randomly selected data points from the first 50% of the RK numerical model.

We can apply ML to this problem through the assertion that there exists some function which approximates the path of the projectile; Equation 2. Therefore we can implement a NN to simply fit to this function, using the 10 data points sampled as seen in Figure 2. We implement early stopping in training to mitigate the NNs susceptibility to overfitting.

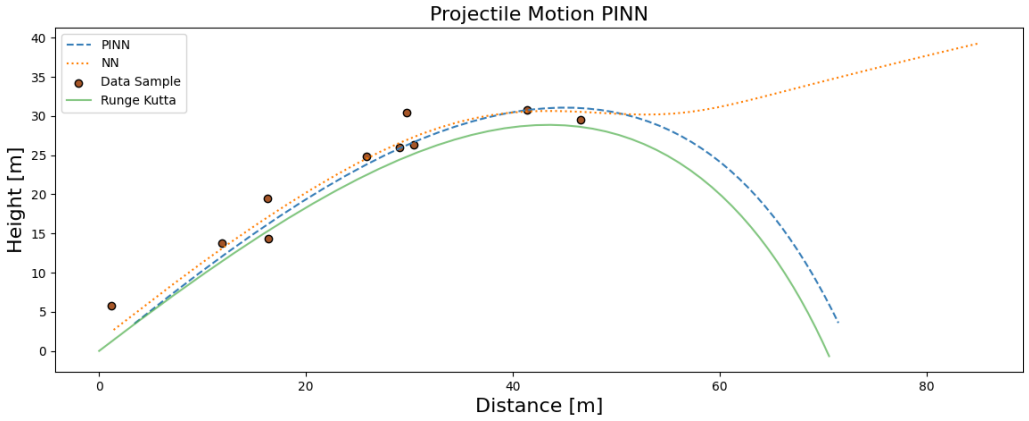

Figure 2: Comparison between PINN prediction (*blue, dashed*) and NN prediction (*orange, dotted*), where the NN is trained on 10 sample data points (*red*) and the PINN is trained on the sample data points and a physics equation (*Equations 3 and 4*) describing its environment (*green, solid*).

$$f(t) = \vec{s} = (x, y) \tag{2}$$

[2]This is not to say that ML methods do not suffer from exponential error growth, but that these errors can be reduced; Shen, Cheng, and Liang, 2020

Network	PM Epochs	PM MAE	PM In Distribution MAE	PM Out of Distribution MAE	UAS MAE
NN	770	9.86	1.99	17.73	0.0193
PINN	3650	4.14	2.42	5.85	0.6650

Table 1: Statistics comparing Projectile Motion (*PM*) and UAS DT network Mean Absolute Error (*MAE*) between network predictions and RK population.

This solution can be improved with a little physics knowledge. Velocity can be expressed as the derivative of $f(t)$ with respect to time; $\frac{\partial f(t)}{\partial t}$, and acceleration as the second-order derivative of $f(t)$ with respect to time: $\frac{\partial^2 f(t)}{\partial t^2}$. Equation 3 can be derived from Newton's Second Law and Drag equation.

$$\vec{a} \equiv \frac{\partial^2 f(t)}{\partial t^2} = -\mu |\frac{\partial f(t)}{\partial t}| \frac{\partial \vec{f}(t)}{\partial t} - \vec{g} \tag{3}$$

We can calculate acceleration through the utilization of the autodifferentiation capability in ML frameworks, differentiating the outputs of our NN with respect to its input. This gives us a calculated value for acceleration and an analytical value for acceleration, which we can use to implement an additional loss function;

$$\mathscr{L}_{phys} = \frac{1}{n} \sum_{i=1}^{n} \left(-\mu \left|\left| \frac{\partial f_i}{\partial t_i} \right|\right| \frac{\partial f_i}{\partial t_i} - \vec{g} - \frac{\partial^2 f_i}{\partial t_i^2} \right)^2 \tag{4}$$

This term compels the network to minimize errors between its *gradients* as well as its prediction errors, and its implementation transforms a NN into a PINN.

Figure 2 and Table 1 illustrate the differences between NNs and PINNs, and the generalizability of PINNs. We can see that the NNs have the advantage of more accurately fitting to the RK population where the predictions are in-distribution. However, it is clear that NN predictions made out-of-distribution are much poorer. PINNs on the other hand, show improved accuracy with a significant reduction in prediction MAE. it is important to note here that PINNs produce poorer predictions in-distribution compared to the NN, but significantly more accurate out-of-distribution predictions.

UAS Twin

In the following sections, we describe the process of creating a DT of a UAS drone using PINNs to learn a high fidelity flight model.

In the context of UAS drones, creating accurate flight models is of paramount importance for trajectory planning and flight control (Sebbane, 2015). Traditional methods of simulating UAS flight dynamics are analagous to the RK example visited before, being based on first principles and with the same disadvantages and challenges which developers tackle today. PINN models therefore offer a promising alternative to manually generated flight models, integrating physics equations into neural networks to accurately predict UAS dynamics directly from observed data while adhering to the underlying physical laws.

The complexity of this environment surpasses that of the previous two-dimensional projectile motion problem. This fact cannot be understated - for additional context; high-performance computers have been employed to address the computational fluid dynamics (CFD) challenges associated with modelling such intricate environments at a desirable level of accuracy for decades (Jansson et al., 2024; Lawson et al., 2012; Braaten, 1991). This makes it a suitable use-case for the application of PINNs due to its inter-platform agnosticism, ease of development, and the well understood physics of the platform environment.

Data Processing

To apply this technique to real-world platforms it is a requirement to have substantial representative telemetry data and a rudimentary knowledge of flight dynamics to train on. To feed our UAS twin PINN, we leverage platform data that describes aircraft state such that the PINN can learn patterns and predict the following state. In this use-case, we utilized the following fields:

- Altitude
- Positional vectors
- Velocity vectors
- Angular velocity vectors
- Operator commands
- Weight
- Wind vectors
- Humidity
- Pressure

We then need to get meaningful outputs which can be used to compare predictions to real results, as well as be differentiated against with respect to an input and expressed analytically. This was determined as:

- ΔV vectors $(\delta v_x, \delta v_y, \delta v_z)$
- $\Delta\Omega$ vectors $(\delta \omega_x, \delta \omega_y, \delta \omega_z)$

This raw data was cleaned to ensure a common usable level of data quality, and to filter out irrelevant sections of flight; for example idling on runway. The data also had to be reconstructed due to mismatched sensor frequencies; this was upsampled through cubic spline fitting and interpolated where necessary. Anomaly removal was handled by implementing a density-based spatial clustering of applications with noise (DBSCAN) methodology over the six-degrees-of-freedom and their first order values and applied to segments of flight tracks with time deltas greater than expected.

Finally, data transformation is necessary prior to model training to increase model learning performance. This involved converting Euler angles and world coordinates to body-relative vectors, followed by standardization to improve training performance.

The Physics Equation

In this simulation we adopt a simplified approach by focusing solely on a single equation that describes the vertical translational kinematics of the aircraft. As the PINN architecture revolves around a methodology of differentiating some or all of the outputs of the network with respect to some or all of its inputs, the $a_z(\Delta v_z)$ output has been selected to be differentiated with respect to the v_z input due to vertical motion being simpler to model. Analytically, this term can be expressed as Equation 5. This quantity itself is meaningless and with units of s^{-1} it is abstract to conceptualise, but it is important to realise that as long as it can be estimated both numerically *and* analytically its physical meaning bears little relevance.

$$\frac{\partial a_z}{\partial v_z} = \frac{v_z}{m} C_L \rho S \tag{5}$$

Validation

To validate this equation we investigated the difference between the analytical prediction compared to values computed using the finite difference method from the test data. This produced significant noise and large errors for short periods in both datasets. The significance of this noise can be explained by the following:

1. Calculating $\frac{\partial a_z}{\partial v_z}$ through finite difference approximations yields frequent and heavy noise as velocity differences which tend to zero results in $\frac{\partial a_z}{\partial v_z}$ tending to infinity. This also highlights this problem for numerical approximation techniques in simulation.

2. The finite difference approximation asserts constant acceleration between time-steps. This assertion therefore erroneously assumes that higher derivatives (jerk, snap, crackle, and pop) are equal to 0.

3. The equation was analytically derived from fundamentals where assumptions such as a windless environment and smooth, continuous flight were assumed. This model also disregards minor physical forces such as vertical drag and the Coriolis force.

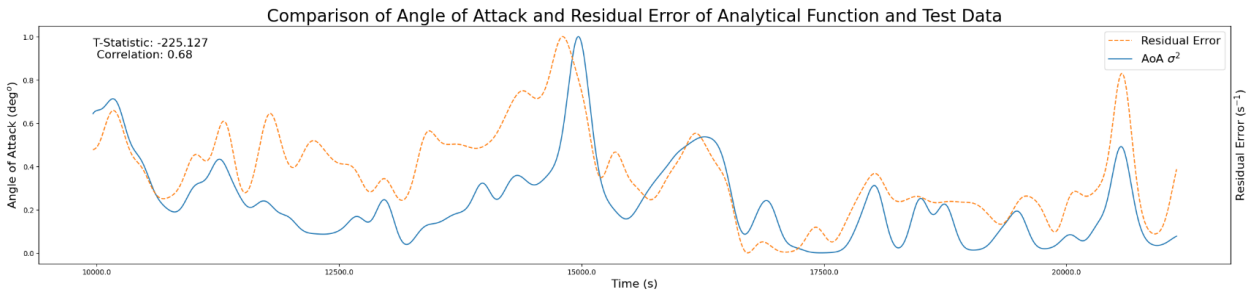

Figure 3: Comparison of Equation 5 Accuracy with Angle of Attack Variance, with Data Processed through a Smoothing Function.

Smoothing functions were utilized to process the test data and compute the differences between the observed test data and the analytical predictions. These differences were subsequently assessed in relation to the variability in the angle of attack (AoA), thereby providing a measure of flight instability. In Figure 3, a strong correlation (~ 0.68) is depicted between the accuracy of the analytical model and the desired flight motion characteristics. This indicates a strong relationship between the phenomena, thus suggesting that the model we are developing best reflects the UAS drone in stable flight.

RESULTS

In this work, we derived a pipeline to create a flight model using PINNs. This pipeline can be utilized in future to create further airborne platform DTs. While it is not recommended to take a "one size fits all" approach to these high fidelity models, this work serves as a proof of concept for a methodology to create a diverse range of DTs in a fraction of the time and cost of creating unique DTs of specialized platforms. Furthermore, this approach could be developed further for different domains; Space being a key area of interest due to the challenge of in-environment testing and evaluation.

Table 1 shows that the Projectile Motion PINN in-distribution predictions produce 21.6% higher MAE, however their out-of-distribution predictions produce 67.0% lower MAE. The PINN predictions for the UAS DT produced almost 35 times the amount of MAE! At first glance, this would suggest that the PINN underperforms relative to the traditional NN - this would be an oversight as the PINN is designed to output more generalized results. Therefore, the results we generated are to be expected as the PINN should not bias strongly towards the data it is presented with and identify what data to "trust" as opposed to representing learned patterns from the data exclusively like a NN.

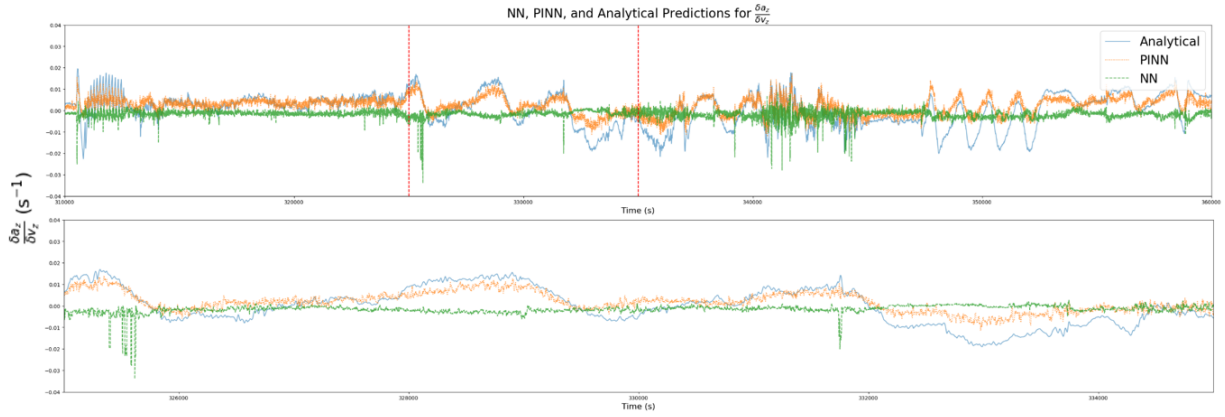

Figure 4: Comparison of NN (*green, dashed*) and PINN (*orange, dotted*) predictions against Test Dataset Sample (*blue, solid*). The section bounded by the red dashed lines have been expanded for improved visibility. The predictions from the PINN demonstrate a closer alignment with the test data compared to the NN.

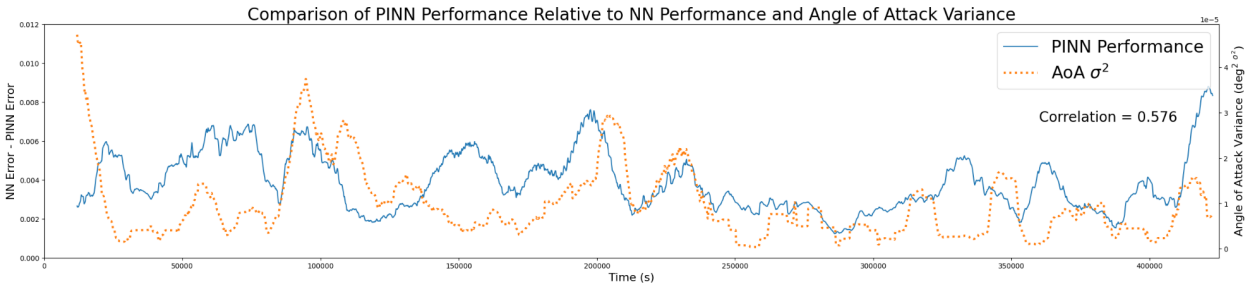

Figure 5: Comparison of PINN performance against the AoA variance. The correlation between PINN performance and AoA variance indicates that the model's performance is linked to flight stability.

Figure 4 illustrates that the PINN model outperforms the traditional NN in fitting to Equation 5, demonstrating the algorithms awareness of the mechanics of its environment. Figure 5 confirms that the relationship between the PINNs conformity to the training data and variance in the platform's AoA is statistically significant[3] as hypothesized during methodology validation.

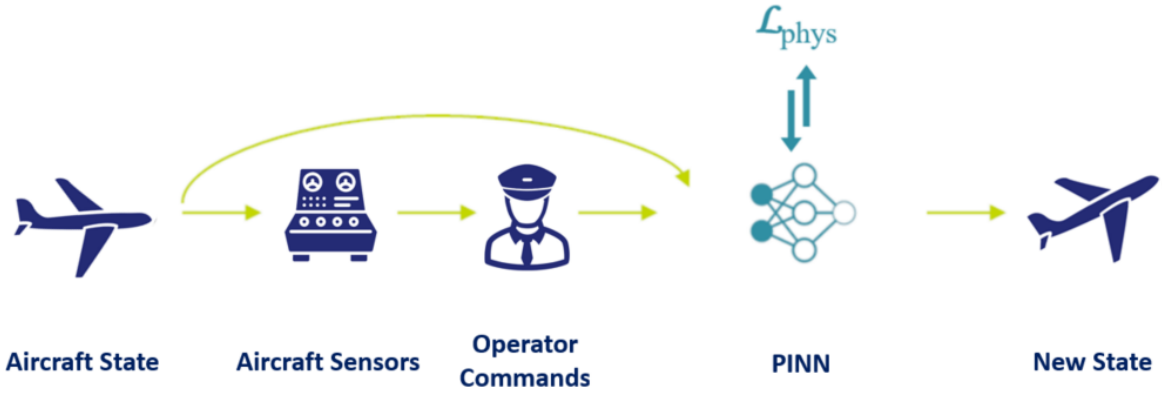

Figure 6: Illustration of the DT flight system, utilizing the PINN as its flight model.

The Digital Twin

The PINN developed in this research approximates the flight control of a UAS DT. By integrating the PINN described above into a Reinforcement Learning (RL) framework, we can leverage the complimentary strengths of learning optimal control policies and dynamics modelling to fly the UAS DT to a set of waypoints in a SE. This system is depicted in Figure 6.

The DT entity was managed through a UAS drone platform ground control system; QGroundControl (Figure 7, *right*, Inc., 2019), and the data stream fed into a High-Level Architecture (HLA) federation. This was then detected and visualized by the Thales Image Generator (Figure 7, *left*).

This DT not only accurately replicates its physical counterpart to a high fidelity level, but also exhibits the unique behaviours and systemic errors present in the sensor data of its twin. This includes limitations of altitude gain and turning rates which affect the platforms ability to meet waypoints, but also to the lower levels of aileron position systemic errors which may affect the decisions made by an operator. This implies that flight models generated using ML methodologies can produce DTs to a level of representation and nuance which traditional simulation methods cannot.

[3]It is a rule of thumb in statistics that variables with correlation magnitude of ≥ 0.3 are related.

Figure 7: Snapshot from UAS Digital Twin from the Thales Training and Simulation (TTS) Digital Twin Image Generator (IG), live in deployment (right). Q Ground Control Service acting as the platform ground control; visualizing the waypoints for the DT to fly to, as well as the historical flight path.

Conclusions and Future Work

In conclusion, Physics-Informed Neural Networks (PINNs) engender a transformative paradigm capable of revolutionizing predictive modelling and product development across a spectrum of disciplines. Not only is there clear utility in utilizing ML pipelines to create DTs rapidly and cost-effectively, but in addition there is a clear performance improvement when using PINNs to improve realism for the general use of unique platforms. However, while it is rare where acquired data adequately represents platform behaviour in all conditions in these instances traditional neural networks provides a superior level of realism. Ultimately, the discoveries from this research shows that in either case of using PINNs or traditional NNs, traditional numerical modelling simulation can be outperformed while retaining the advantages of cost-effectiveness, realistic platform behaviour, and rapid development.

There is further investigation to be done on PINNs. One area of interest is exploring their resillience to different types of complex noise. Noise collected from the real world often display complex embedded patterns. This work only reflects PINN resilience to random gaussian noise, therefore investigation into PINN performance with the inclusion of patterned noise would be of interest to establish what data processing is compulsary prior to training. PINNs can also be used to learn parameters used in the equation derived by configuring them as trainable parameters. In the case of UAS platforms, PINNs can be used to approximate the value of the coefficient of lift C_L. This could have applications in early product development where instead of compulsary use of large wind tunnels and significant financial investments into materials and labour for prototyping, a DT of a platform can be created to acquire these parameters. Alternatively, this can be used for ISR purposes to gain an advantage on adversarial forces by creating DTs of platforms from intelligence data and injecting them into training simulations.

The PINN developed for this use case can almost certainly be improved with the addition of physics equations describing more about the environment. The network developed could be improved through the provision of more information about the UAS drone behaviour in non-stable flight and more extreme conditions.

The pipeline used in this research can be generalised to be agnostic to training data. At the time of writing, there is work being undertaken to use this technique to approximate dead reckoning for maritime entities from Automatic Identification System (AIS) streams and airborne entities from Automatic Dependent Surveillance-Broadcast (ADS-B) streams.

ACKNOWLEDGEMENTS

The work conducted in this research would not have been possible without members of Thales Training and Simulation (TTS) community first laying the foundations. Undoubtedly, Ian Henderson has made this work possible through his work in generating the first iteration of the flight model used in this article to benchmark the PINN. In addition, the visualizations of the UAS DT created in a custom SE exists solely because the UKDTC team at Thales developed these components and infrastructure. Feedback from the whole of TTS has been iteratively given throughout the process of conducting this work and writing this article.

I would like to thank all of those mentioned above for their hard work and generosity with their time.

References

Alcántara Suárez, Evaldo Jorge and Victor Monzon Baeza (2023). "Evaluating the Role of Machine Learning in Defense Applications and Industry". In: *Machine Learning and Knowledge Extraction* 5.4, pp. 1557–1569. ISSN: 2504-4990. DOI: 10.3390/make5040078. URL: https://www.mdpi.com/2504-4990/5/4/78.

Argolini, Roberto, Federico Bonalumi, and Johannes Deichmann (2023). "Digital twins: The key to smart product development". In: *Annals of Mathematics*.

Barnett, Janet Heine (2009). "Mathematics goes ballistic: Benjamin Robins, Leonhard Euler, and the mathematical education of military engineers". In: *BSHM Bulletin: Journal of the British Society for the History of Mathematics* 24.2, pp. 92–104. DOI: 10.1080/17498430902820887. eprint: https://doi.org/10.1080/17498430902820887. URL: https://doi.org/10.1080/17498430902820887.

Braaten, Mark E (1991). "Development of a parallel computational fluid dynamics algorithm on a hypercube computer". In: *International journal for numerical methods in fluids* 12.10, pp. 947–963.

Budning, Kevin, Alex Wilner, and Guillaume Cote (2022). "From physical to virtual to digital: The Synthetic Environment and its impact on Canadian defence policy". In: *International Journal* 77.2, pp. 335–355. DOI: 10.1177/00207020221135302. eprint: https://doi.org/10.1177/00207020221135302. URL: https://doi.org/10.1177/00207020221135302.

Butcher, J. C. (2015). "Runge–Kutta Methods for Ordinary Differential Equations". In: *Numerical Analysis and Optimization*. Ed. by Mehiddin Al-Baali, Lucio Grandinetti, and Anton Purnama. Cham: Springer International Publishing, pp. 37–58.

Catapult, Digital (2024). *UK Digital Twin Centre*. April 5th, 2024. URL: https://www.digicatapult.org.uk/expertise/programmes/programme/uk-digital-twin-centre/.

Das, Arun and Paul Rad (2020). "Opportunities and Challenges in Explainable Artificial Intelligence (XAI): A Survey". In: *CoRR* abs/2006.11371. arXiv: 2006.11371. URL: https://arxiv.org/abs/2006.11371.

El-Fouly, Shams El-Din and Siddharth Misra (1918). "8.4-Runge Kutta Methods (2nd & 4th Order)". In.

Galilei, Galileo (1914). *Dialogues concerning two new sciences*. Dover.

Gelernter, David Hillel (1991). *Mirror worlds, or, The day software puts the universe in a shoebox– : how it will happen and what it will mean*. New York: Oxford University Press.

Grieves, Michael (Aug. 2016). "Origins of the Digital Twin Concept". In: DOI: 10.13140/RG.2.2.26367.61609.

Huang, Xiaowei et al. (2020). "A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability". In: *Computer Science Review* 37, p. 100270. ISSN: 1574-0137. DOI: https://doi.org/10.1016/j.cosrev.2020.100270. URL: https://www.sciencedirect.com/science/article/pii/S1574013719302527.

Inc., Linux Foundation (2019). *Dronecode Project*. Version 4.2.1. URL: https://www.qgroundcontrol.com.

Jansson, Niclas et al. (2024). "Neko: A modern, portable, and scalable framework for high-fidelity computational fluid dynamics". In: *Computers & Fluids* 275, p. 106243. ISSN: 0045-7930. DOI: https://doi.org/10.1016/j.compfluid.2024.106243. URL: https://www.sciencedirect.com/science/article/pii/S0045793024000756.

Kapteyn, Michael G. and Karen E. Willcox (2020). *From Physics-Based Models to Predictive Digital Twins via Interpretable Machine Learning*. arXiv: 2004.11356 [cs.CE].

Krishnababu, Senthil, Omar Valero, and Roger Wells (2021). "Ai assisted high fidelity multi-physics digital twin of industrial gas turbines". In: *Turbo expo: power for land, sea, and air*. Vol. 84935. American Society of Mechanical Engineers, V02DT36A007.

Lawson, SJ et al. (2012). "High performance computing for challenging problems in computational fluid dynamics". In: *Progress in Aerospace Sciences* 52, pp. 19–29.

Nabian, Mohammad Amin and Hadi Meidani (2018). "Physics-informed regularization of deep neural networks". In: *arXiv preprint arXiv:1810.05547* 746.

Raissi, Maziar, Paris Perdikaris, and George E. Karniadakis (2017). "Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations". In: *CoRR* abs/1711.10561. arXiv: 1711.10561. URL: http://arxiv.org/abs/1711.10561.

Ramabathiran, Amuthan A. and Prabhu Ramachandran (2021). "SPINN: Sparse, Physics-based, and partially Interpretable Neural Networks for PDEs". In: *Journal of Computational Physics* 445, p. 110600. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2021.110600. URL: https://www.sciencedirect.com/science/article/pii/S0021999121004952.

Ramesh, Adithya and Balaraman Ravindran (2023). "Physics-Informed Model-Based Reinforcement Learning". In: *Learning for Dynamics and Control Conference*. PMLR, pp. 26–37.

Raymond, Samuel J. and David B. Camarillo (2021). *Applying physics-based loss functions to neural networks for improved generalizability in mechanics problems*. arXiv: 2105.00075 [physics.comp-ph]. URL: https://arxiv.org/abs/2105.00075.

Sebbane, Yasmina Bestaoui (2015). *Smart autonomous aircraft: flight control and planning for UAV*. Crc Press, pp. 19–95.

Shen, Xing, Xiaoliang Cheng, and Kewei Liang (2020). *Deep Euler method: solving ODEs by approximating the local truncation error of the Euler method*. arXiv: 2003.09573 [math.NA]. URL: https://arxiv.org/abs/2003.09573.

Smith, Mr Nicholas (2023). "The OdySSEy to Transform Mission Readiness and Training Effectiveness". In.

Society, IEEE Computer (Dec. 2012). "IEEE Standard for Distributed Interactive Simulation - Application Protocols". In: *IEEE Standard* 1278.1-2012, pp. 673–689.

Wang, Fujin et al. (2023). "Inherently Interpretable Physics-Informed Neural Network for Battery Modeling and Prognosis". In: *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15. DOI: 10.1109/TNNLS.2023.3329368.

Whelan, Patrick Michael and John Michael Hodgson (1978). "Essential principles of physics". In.

Žukovs, Artūrs and Kristaps Upenieks (2023). "Data ownership in training artificial intelligence in healthcare". In: *Socrates. Rīga Stradiņš University Faculty of Law Electronic Scientific Journal of Law.* 2023.1-27, pp. 81–86. DOI: doi:10.25143/socr.27.2023.3.81-86. URL: https://doi.org/10.25143/socr.27.2023.3.81-86.