Defining Surrogates for Industrial Use - The JuliaSim Surrogate Approach

Many people approach building surrogate models for industrial simulation as a problem of machine learning architectures. The surrogates problem is to generate a machine-learned model that approximates a physical model to a high degree of accuracy but runs orders of magnitudes faster than the original model. In this approach, the focus tends to be on the construction of new neural networks, like physics-informed neural networks (PINNs)[1], DeepONets[2], Continuous-Time Echo State Networks (CTESNs)[3], Fourier Neural Networks (FNO)[4], and more, which, when given sufficient data from a simulation, can reproduce their behavior. This does not address the underlying problem of integrating next-generation machine learning into industrial modeling and simulation workflows or the real problem of integrating surrogates into industrial processes and requirements infrastructure.

How do Surrogates Fit into Industrial Processes and Requirements Infrastructure?

These are the processes that are in place to make sure that an airplane that is designed by a controls engineer will stay in the air, ensure that drugs that enter the market are

[1] Raissi, Maziar, Paris Perdikaris, and George E. Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations." *Journal of Computational physics* 378 (2019): 686-707.

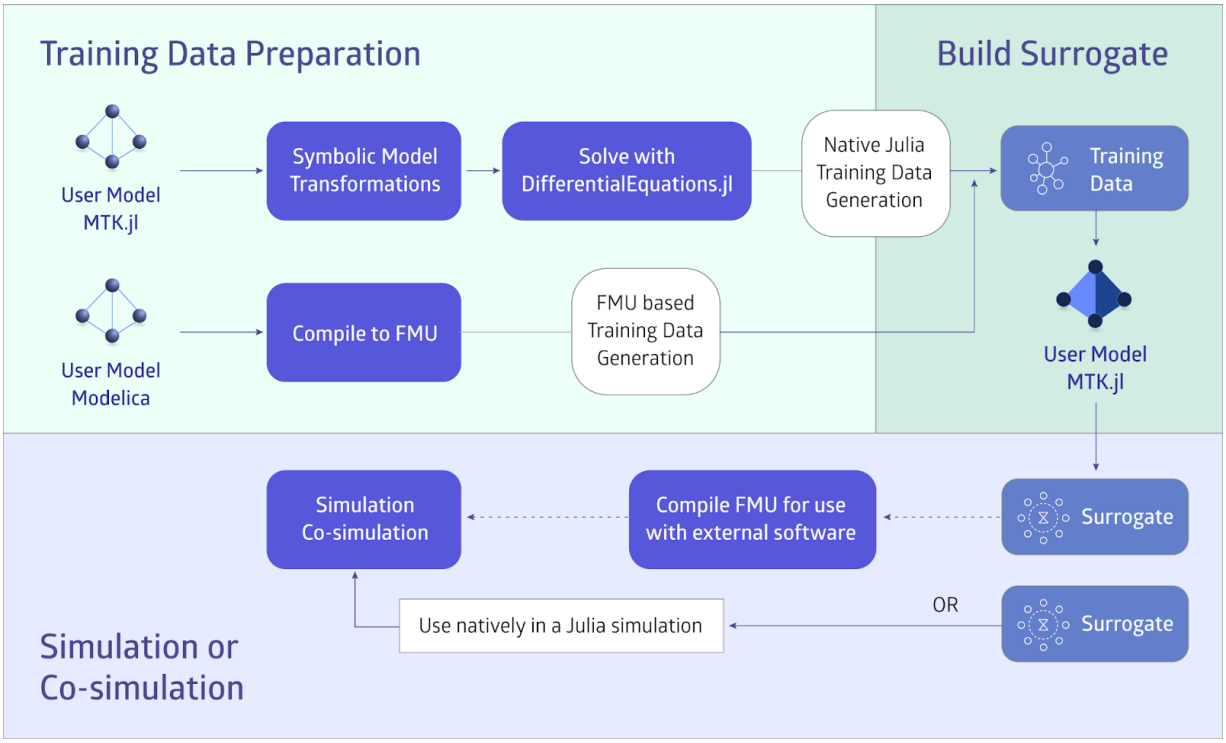
[2] Lu, Lu, et al. "Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators." *Nature machine intelligence* 3.3 (2021): 218-229.

[3] Anantharaman, Ranjan, et al. "Accelerating simulation of stiff nonlinear systems using continuous-time echo state networks." *arXiv preprint arXiv:2010.04004* (2020).

[4] Li, Zongyi, et al. "Fourier neural operator for parametric partial differential equations." *arXiv preprint arXiv:2010.08895* (2020).

safe, and make sure that the products being developed take into account the realities of supply chains to be built in robust but reproducible ways. As scientific and engineering industries have entered the digital age, these processes have been refined so that mathematical and computational contributions can enhance the full pipeline, from research and deployment all the way to the late stages of fleet management.

For the next generation of machine learning technologies to succeed in advanced engineering industries, they must be adapted to these processes and operate within the required infrastructure. At a high level, what are the requirements that are imposed through these systems?



Deployable Surrogate Systems Must Have Predictable Accuracy

Giving a neural network more data to learn from *should* improve its ability to make better predictions. However, this can be highly dependent on whether the right hyperparameters are chosen. As the optimization process falls into a different local minima, it may not have improved even with the enlarged dataset. Maybe a larger neural network is necessary but it's not always clear when increasing the size of the neural network will actually improve predictions.

Industrial use of machine learning needs to ensure that projects do not introduce a new vector that may seemingly randomly regress. Simulations of fluid flow through an airplane engine will be checked by engineers to reach a required prediction precision. If the system needs to be more accurate due to downstream requirements changes, the engineers know how to reliably increase this precision at the cost of compute power. The processes and requirements of simulation thus act first and foremost as a pull on precision. "We need this accurate to 1% on the landing forces", and with that, modeling and simulation are performed to give computational tools that reach the precision required by downstream teams for a successful handoff.

It's not a given that the best neural network architecture in test scenarios is also guaranteed to be reliably able to meet accuracy requirements. While machine learning is adequate for generative AI since the model does not need to change until the language changes (a slow process), surrogate models need to be retrained every time the specifications or questions change. Engineers need a machine learning process that guarantees desired accuracy and gives a speedup while hitting all requirements. Thus, what's important is not the neural architecture but a process that can reliably generate accurate neural approximations.

A machine learning process that can guarantee always having a model of desired accuracy and provide a speed up while hitting all of the requirements is what helps engineers. Thus, what's important in the industrial context is not the neural architecture but having a process that can reliably generate accurate neural approximations.

Deployable Surrogate Systems Require Clear Verification and Validation Standards

The right understanding of the surrogates problem asks: "What are the processes so that, if we need 1% accuracy of a safety critical prediction over the whole space, we can derisk our analyses as much as possible?". This question is tractable. It's a question that leads to concrete questions like:

- What kinds of plots and visualizations would best help us understand where the surrogate is most accurate and inaccurate?
- If the accuracy of a specific area of the parameter space needs to be improved, how can a retrain be enforced so as to improve accuracy without sacrificing its accuracy elsewhere?
- How can the algorithm indicate what additional data is required or what hyper parameters should be adjusted?
- What are the right metrics and numbers that should be tracked so that, if sufficiently large, should be treated as a warning that the surrogates may not be a reliable resource for a given parameter range or scenario?
- How can this whole process be simplified so that new employees with domain expertise and no machine learning experience can easily "do it right" with the right safeguards to ensure inaccurate models are not accidentally deployed?

These are the real questions that need to be addressed in order for an engineering team to safely develop internal processes and deploy neural surrogates throughout their firm.

Any Deployable Surrogate System Must Focus on Reliable Prediction Time

A common question about surrogates is whether the time saved compensates for the time spent training them. In other words, did the use of the surrogate decrease the overall compute over not using the surrogate? If you had to run the expensive simulation model 1,000 times in order to build an accurate surrogate model which was used in a downstream analysis, say in a parameter estimation loop that requires solving the model 5,000 times, did the surrogate architecture "pay for itself" or would it have been faster to simply run the original analysis on the slow model?

Looking at industry applications of surrogates reveals that this question is a red herring as to the utility of a surrogate in applications for a number of reasons.

Many applications of surrogates are enabling analysis that would be impossible without the surrogate's inference speed improvements. For example, in the control of power grids, there is a strict requirement that a model's evaluation be able to be run at X Hz in order to be deployed in the control scenario. This compute cost bound is thus the fundamental limiting factor of whether a model can be used in this context, and for this reason, often crude approximate models are used, which can lead to inefficient control strategies. Many billions of dollars can be gained by having a more accurate model. Accurate (partial differential equation) models exist which are much higher fidelity but are too slow to be run in this real-time scenario.

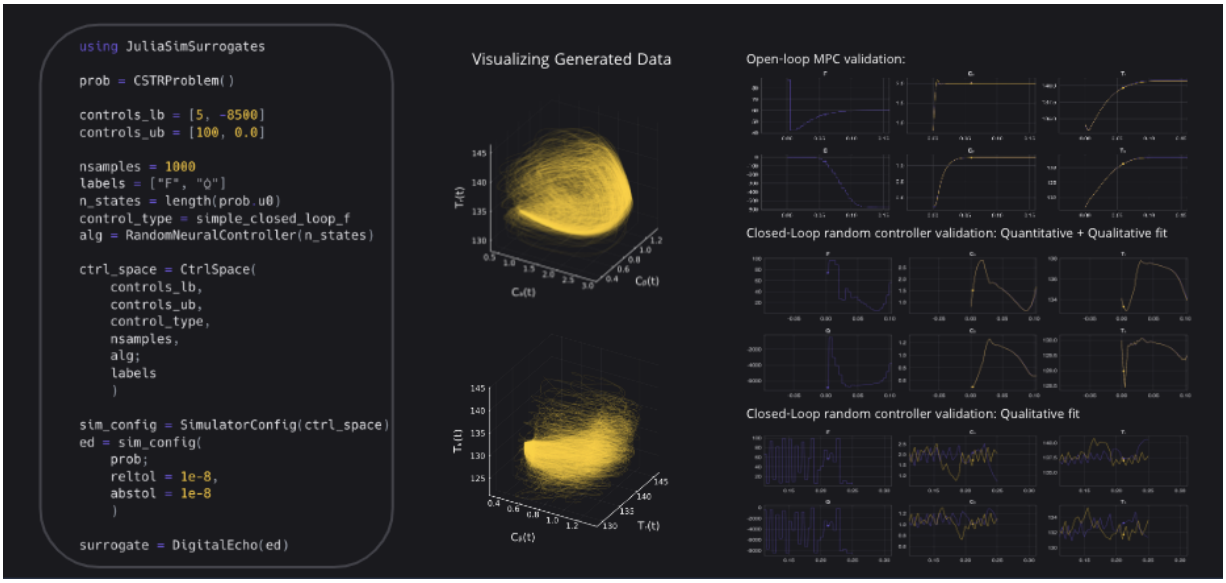
The critical factor here is not the time it takes to train the surrogate (whether a week or six months), but whether it can reliably provide the required accuracy within the specified time frame. This focus on meeting requirements takes precedence over architectural considerations aimed at saving training time.

The data generation process is not necessarily a serial process. In fact, in many cases, the data generation is simply independent runs of a simulator at new parameters, a process that is embarrassingly parallel. This means that in theory, the time it takes to run the model with 5,000 different parameters is simply the true time cost of running the model once with its most expensive configuration of the set.

With the advent of cloud, parallelism has no additional cost. As a result, the impact of data quantity on building surrogates is minimal. Instead, it becomes a question of value: is the value of the surrogate greater than the cost of its training? This question of value is again a question of how it's integrated into processes: how many engineers is this improving the workflows and productivity of? Given that we are talking about highly skilled engineers in high-tech industries, the effect of productivity increases can be a force multiplier when spread over a whole team or department. What matters most is that the engineers can rely on the surrogate to always work as expected and, as such, not give them the extra work of triple-checking its accuracy every time it's used. Surrogates could be a productivity boost as large as the original push to digitization if done correctly, or they can be a sideshow that puts entire teams on a wild goose chase if not.

The true value cannot be unlocked without addressing the core aspect of having surrogate training processes and requirements that focus on how to use the surrogate techniques of the day in order to unlock as much productivity out of engineers as possible.

Digital Echo: A Neural Surrogate Built Around Industrial Processes



Digital Echo - Generating Surrogates for Tuning Controllers using an ODEProblem Source

The JuliaSim approach addresses the surrogate question from the standpoint of processes and requirements. The core questions above drive the development of our JuliaSim surrogate trainer product.

Questions like:

- What are the neural architectures that will most reliably improve if trained on more data and are not hampered by issues associated with having too many hyperparameters to tune?
- How do we develop a visualization suite around the surrogate training processes so that scientists can be in the loop, easily understand the accuracy profile of the surrogate, and understand where or if the training must be improved?

- Through what means can we present an architecture that is reliable and automatically masks training costs to maximize productivity via effective use of cloud infrastructure?
- How do we take all of this and make it usable and accessible to domain scientists and engineers with no machine learning background?

These questions have driven us to develop our new architecture, Digital Echo. Its focus is not on the mathematical structures or the layer definitions; here we give absolutely no definition of the mathematics inside how it computes NN(x). It's not about the architecture; Digital Echo is about a process. A process for reliably creating the best surrogate it can, with processes to know when it needs to retrain, when it needs to warn you about accuracy, and processes for how to generate legible reports to document surrogate accuracy. Digital Echo is a completely different take on surrogates because it's about making sure someone can click a button and get a surrogate that meets the industrial requirements set out in their specifications sheet. Digital Echo is not about performance but productivity.