Reproduction of a physics-informed neural network to solve optimization tasks

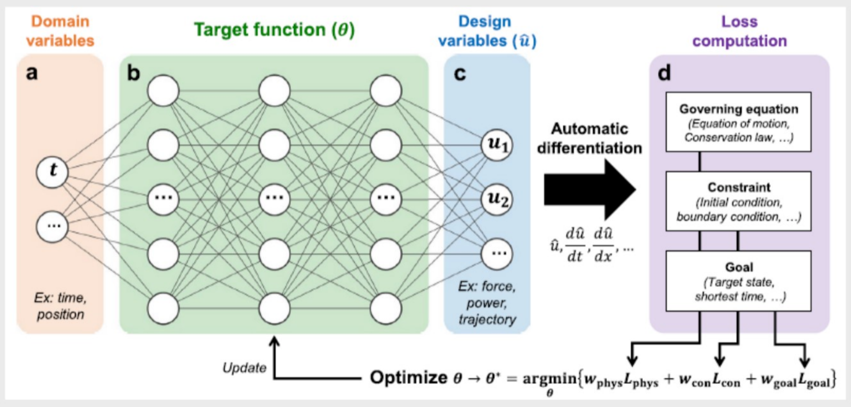
Authors

Jose Cunha
Lucas Van Mol
Mathijs van Binnendijk
Nicolas Fajardo Ramirez

TU Delft
Delft University of Technology

Introduction

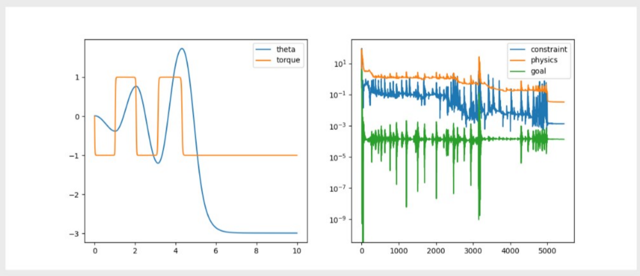
A PINN is a neural network that can solve differential equations and learn the underlying physics of a given problem while doing so. The network's architecture consists of an initial layer for the domain variables (a), multilayered perceptrons that compose the target function to be optimized (b), output layer of design variables (c), and the loss functions that are used to update the function (d). [1]
In this project, we worked on the replication of PINN in its application in four different problems: a swinging pendulum, a spacecraft swing by trajectory, a spacecraft landing trajectory, shortest path between two points (light refraction and under gravity).


Domain variables
Target function (θ)
Design variables (ũ)
Loss computation
a
b
c
d
Ex: time, position
Ex: force, power, trajectory
Automatic differentiation
Governing equation (Equation of motion, Conservation law, ...)
Constraint (Initial condition, boundary condition, ...)
Goal (Target state, shortest time, ...)
Update
Optimize θ → θ* = argmin{w_phys L_phys + w_con L_con + w_goal L_goal}

01

Pendulum

The paper introduces the use of PINNs for optimization tasks by taking the example of a pendulum powered by a low-torque actuator. The objective is to invert the pendulum, such that cos(θ) = -1 at 10 seconds, the challenge being that multiple swings are needed in order to accumulate enough energy.



The figure above shows our implementation working for this example. On the left, we see that the network has learned to modulate the torque, causing the maximum pendulum angle to increase until inversion (θ = π).

On the right, we see how the network optimizes:

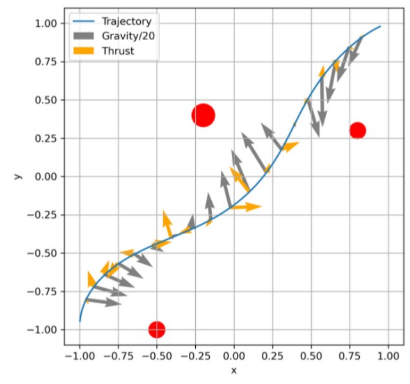
- physics loss, defined by the gravitational and pendulum governing laws
- constraint loss, ensuring the pendulum starts at θ = 0 with 0 velocity
- goal loss, ensuring the pendulum is at θ = π when t = 10.

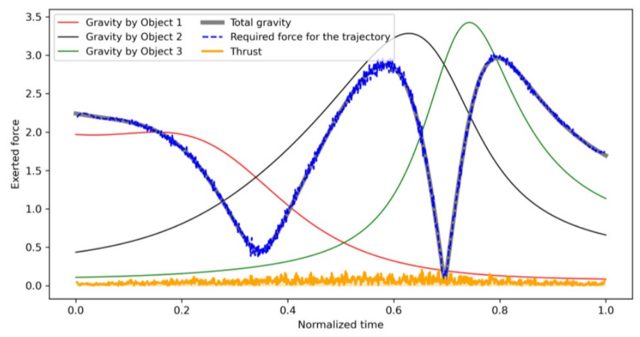
We are able to match the results of the paper using the same hyperparameters. It must be noted that there is quite a lot of brittleness related to the convergence of the model, both in the paper and our own implementation. Indeed, our model seems to converge only about 30% of the time, meaning multiple runs are needed before getting a "nice" result as above. The code included in the paper shares similar problems, and we feel that the author fails to address such an obvious issue. As a result, metrics such as training time are misleading.

02

Spacecraft Swingby

In the paper, the problem of finding the swingby trajectory of a spacecraft that can reach the given destination using the least amount of thrust is presented. A spacecraft must fly between a series of astronomical bodies, leveraging the gravity felt due to each body, and using thrust to adjust its trajectory to reach a certain final point. Two loss functions are defined, one to define the physics of the problem (which encodes the goal of also minimizing the thrust), and another to define the boundary conditions.
The following two figures show the trajectory and force results of this PINN implementation:



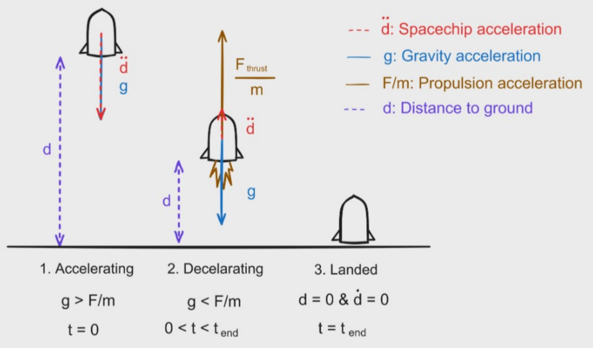


The spacecraft trajectory matches the results from the paper quite accurately, and the thrust required for the manoeuvre is relatively low, though noisier than that of the paper. The training hyperparameters also vary from the paper, where a higher weight is attributed for the constraint loss (10 instead of 1), the learning rate was increased (to 0.004) and the L-BFGS training epochs were quadrupled.

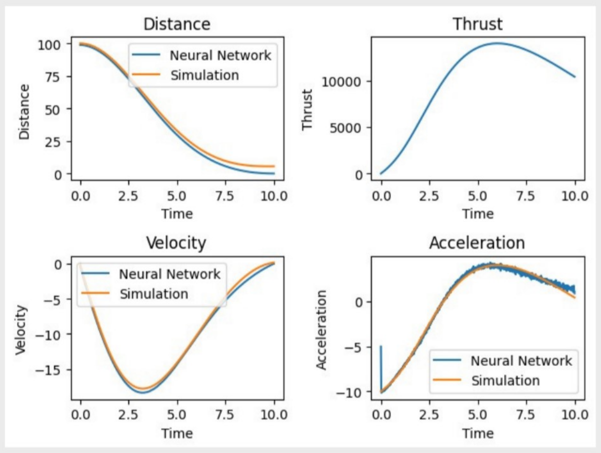
03

Spacecraft Landing

In order to see if the architecture proposed in the paper would extend to new use cases, an original experiment was created. This experiment was getting the PINN to predict the landing trajectory of a spacecraft in one dimension.



Here the PINN takes a time as input and generates the according thrust force and distance to ground required for a successful landing. The following figure shows the results:



The PINN results seem to largely overlap with the simulation performed based on the initial conditions and thrust curve. This indicates the technique discussed in the paper extends to use cases apart from those mentioned in the paper. To determine to what degree this is the case requires more research.

Comparison & Conclusion

Overall, our reproduction is able to match the results of the paper quite accurately. There is, however, a considerable percentage of runs where the loss would diverge, rendering useless results. To finalize our reproduction, we aim to empirically quantify which percentage of training runs are able to converge. Considering the training hyperparameters for the spacecraft swingby case, these values had to be heavily modified such that the loss would converge to a reasonable value, in a reasonable length of time. For both the spacecraft swingby and shortest path cases, an external trainable parameter also had to be included as a normalization factor to the physics losses. The paper, however, completely omits how this parameter is trained. The paper also omits the fact that the input vector is resampled after a certain number of epochs to improve generalization. Thankfully, a repository of the original code existed, which aided the reproduction.

In conclusion, the paper by Seo, J. [1] was successfully reproduced using PyTorch instead of DeepXDE and Tensorflow. A new example of an optimization task was also successfully solved with the reproduced PINN architecture, though it did require tweaks to the original architecture in the form of adding additional weights. Having one weight for physics, constraints, and the goal was not enough and instead one weight for all the different elements of these loss aspects had to be implemented.

[1] Seo, J. (2024). Solving real-world optimization tasks using physics-informed neural computing. Sci Rep 14, 202