Partical work Pinn's introduction

Sidney BESNARD

August 29, 2024

1 Introduction

In this practical work, the goal is to build a Physics-Informed Neural Network (PINN). The concept of PINNs involves training and guiding a neural network using a fixed physical model (PDEs, ODEs, etc.). Generally, the objective of PINNs is to approximate physical equations and leverage experimental data.

We focus on solving an orbit restitution problem, using the initial parameters of an acquisition described by a given physical model. In this example, the physical model consists of an orbit propagator and a projection model to generate a 2D projection and data acquisition. Let $x = f(y)$ represent the physical model that, based on the parameters of an orbit y, generates an image x corresponding to the long exposure of a satellite placed in this orbit, acquired by a sensor located on Earth.

The problem of finding y from x is known as an inverse problem, which is common to many problems in science. Traditionally, solving such inverse problems involves complex non-linear optimization techniques. However, neural networks can be used as an alternative approach to approximate the solution. While neural networks are powerful tools for such tasks, they can produce estimates that are not physically possible. This justifies the use of Physics-Informed Neural Networks (PINNs), which incorporate physical constraints into the neural network to ensure that the solutions are physically meaningful.

In this exercise, you will have to train a neural network to invert the direct model f, both with and without the use of PINNs. This will allow you to compare the performance and accuracy of the neural network in recovering the orbit parameters when physical constraints are incorporated versus when they are not. By doing so, you will be able to visualize the benefits of incorporating physical constraints into the neural network and understand the importance of PINNs in solving inverse problems.

Let $\Psi(x)$ be a neural network, trained using two loss functions.

- MSE Loss

$$\frac{1}{N}\sum_{k=1}^{N}(\Psi(x_k) - y_k)^2 \tag{1}$$

- Physical Loss

$$\frac{1}{M}\sum_{k=1}^{M}(f(\Psi(x_k)) - x_k)^2 \tag{2}$$

2 Setup

Required packages: Python 3, PyTorch, NumPy, Matplotlib.

(It is recommended to work in a Google Colab environment to avoid any issues.)

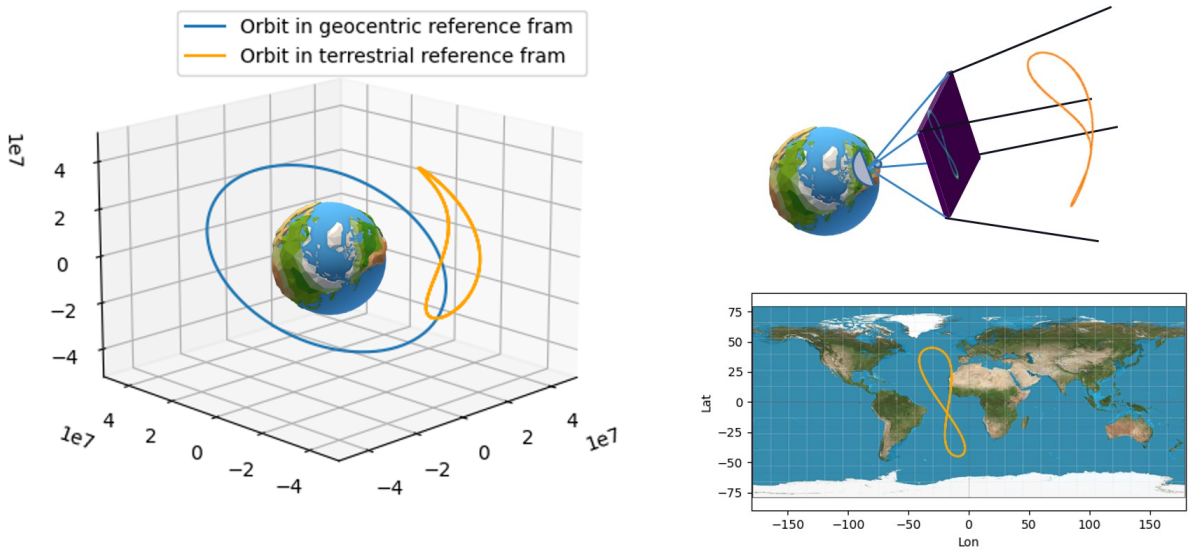

Figure 1: Left : Representation of an orbit in the terrestrial referential (ECEF). Right : Representation of an acquisition

1. Download the physical models files (TODO google drive url).

2. The model we will use is located in the `decoder.py` file. Refer to the Python documentation for more details and to understand how to use the generator function.

3 Data Visualization and Dataset Generation

1. Generate a single data point of size 32 with all parameters set to 0.

2. Plot the resulting image. You should see a centred single point.

3. Experiment with the orbit parameters to observe the possible deformations and results.

4. (Optional) Check if the model is fully differentiable using `.backward()`.

5. Generate a training dataset of 3,000 examples and a test dataset of 1,000 examples using `torch.utils.data.TensorDataset`, with the following constraints: $e \sim \mathcal{U}([0, 0.5])$, $i \sim \mathcal{U}([0, 20°])$, and $\Omega \sim \mathcal{U}([-10°, 10°])$.

4 PINNs Training

4.1 Model

The proposed model architecture for this work is a simple MLP with a depth of 5 layers and intermediate layers of size 16. The input size of the model should be 32×32, and the output should be a tensor of size 3.

1. Implement a neural network architecture accordingly (refer to `torch.nn.Sequential` and `torch.nn.Linear`).

4.2 Training

2. Create a standard supervised training loop (based on your previous work).

3. Train the neural network using the following loss function:

$$\mathcal{L}(x, y) = \lambda \frac{1}{N} \sum_{k=1}^{N} (\Psi(x_k) - y_k)^2 + (1 - \lambda) \frac{1}{N} \sum_{k=1}^{N} (f(\Psi(x_k)) - x_k)^2 \tag{3}$$

4. Compare the results with training using only a standard supervised learning loss (MSE loss).

5. Experiment with the trade-off between supervised learning and physics-based learning by varying the λ parameter.