Simulation-Augmented Physics-Aware Neural Networks for Nonlinear Inverse Problems

Sidney Besnard, Frédéric Jurie, Jalal M. Fadili

Simulation-Augmented Physics-Aware Neural Networks for Nonlinear Inverse Problems

Sidney Besnard[a,b], Frederic Jurie[b], Jalal Fadili[b]

[a]*Safran Data Systems, Colombelles, Normandie, France*
[b]*Normandie Univ, UNICAEN, CNRS, GREYC, Normandie, France*

Abstract

This study proposes a novel approach to inverse problems in the aerospace domain, with a particular focus on orbit determination. By leveraging deep learning, our framework infers the underlying physical parameters from observed data, even in the presence of highly nonlinear forward models. Crucially, we consider settings where the dimensionality of the physical parameters is substantially lower than that of the observations.

Our method trains a neural network with a hybrid loss function that combines observational data and simulated data derived from an imperfect physical model. This joint utilization of actual and simulated data enables effective learning by pairing simulated observations with their corresponding parameter values. Experimental evaluations, including orbit determination tasks, highlight the advantages of our Simulation-augmented Physics-Aware Neural Networks (SimPANNs) over traditional methods that rely solely on observational data. Not only does our approach enhance both accuracy and robustness, but it also demonstrates a promising path for solving nonlinear inverse problems in reduced-parameter spaces by unifying physics-based modeling with data-driven techniques.

Keywords: Inverse problems, orbit determination, nonlinear problems, machine learning, neural network

1. Introduction

Data-driven methods for inverse problems Let $\mathcal{X} \subset \mathbb{R}^d$ be the space of parameters of some (physical) model, and let $\mathcal{Y} \subset \mathbb{R}^m$ be the space of observations. An inverse problem consists of reliably recovering the parameters $x \in \mathcal{X}$ from noisy indirect observations:

$$y = f(x) + \varepsilon \tag{1}$$

where the mapping $f : \mathcal{X} \to \mathcal{Y}$ is the forward operator, and ε stands for some additive noise that captures the measurements and possibly the modeling error.

Inverse problems play a crucial role in science by allowing to unravel the hidden properties and processes behind observed data y. They allow scientists to infer and understand phenomena that are otherwise difficult or impossible to observe or measure directly. These problems involve determining the parameters of a system from the available measurements. Inverse problems have far-reaching applications spanning a wide spectrum ranging from biomedical imaging [1], to seismology [2] or astronomy [3].

Data-driven methods, and more prominently deep learning-based ones, have emerged as a powerful approach for addressing inverse problems [4, 5, 6, 7, 8]. They form a data-driven alternative to variational model-based approaches. Inverse problems often involve complex and nonlinear relationships between the observed data y and the underlying parameters or causes x as materialized by the mapping f [9, 10, 11]. Machine learning methods can learn these relationships from large datasets, allowing them to make predictions and infer the unknown parameters. One common technique is to train a neural network (NN) using a dataset consisting of n training input-output pairs $\{y_i, x_i\}_{i=1}^n$, where the inputs y_i are the observed data and the outputs are the corresponding parameters x_i. The neural network learns the mapping between the input and output, enabling it to predict outputs for new observations. This approach leverages the ability of neural networks to accurately approximate complex functions and to capture intricate patterns and nonlinear dependencies in the data. In practice, this type of technique is only applicable when training pairs are available. This is barely the case in most critical applications. Moreover, in the context of inverse problems, composing the forward mapping with the learned inverse mapping may not reproduce a faithful version of the observation.

Data-driven physics-aware methods To cope with these limitations, data-driven methods for inverse problems that take into account the observation forward model have been proposed; see [12, 13, 14] and references therein. Physics-aware neural network-based methods have also gained significant attention in recent years to solve partial differential equations (PDE). These include PINNs [15, 16, 17, 18], DeepONet [19, 20], neural operators [21] or PEDS [22]. One of the core ingredients of all these methods is to supplement the neural network training with information stemming from the observation formation model. To avoid confusion with PINNs, we use the abbreviation PANNs for all physics-aware NN-based methods. PANNs combine the strengths of physics-based modeling and NN learning to address inverse problems in a more robust and interpretable manner. In PANNs, the governing equations and physical constraints (e.g. boundary or initial value conditions, geometry of the domain, symmetries) of the underlying system are explicitly taken into account during the training phase of the neural network. By incorporating these known physical principles

into the training process, PANNs can effectively learn the relationship between the observed data and the unknown parameters while complying with the fundamental laws of the system. The combination of physics-based modeling and NN learning allows for more accurate and reliable solutions while leveraging the domain knowledge encoded in the underlying physics. Generally speaking, PANNs are based on observed data y only, and therefore have the advantage of not requiring training data pairs. They can therefore be considered as single-instance unsupervised approaches.

Inverse problems have two intrinsic peculiarities that pose specific difficulties. First, inverse problems are mostly ill-posed, i.e. multiple solutions are generally compatible with a single observation. In addition, they raise the question of stability in the presence of noise. Regularization techniques seek to alleviate this ill-posedness by incorporating prior knowledge about the sought-after parameters in the form of some low-complexity (e.g., smoothness, sparsity) [23]. This allows one to restrict the space of candidate solutions, hence prevent overfitting, and ensures stability to noise. Second, as mentioned above, inversion – i.e. the estimation of parameters from observations – is a highly nonlinear process, even with a linear forward model, let alone the case of nonlinear forward models. These are challenging and difficult problems to solve numerically.

Motivation from orbit determination Our main motivation comes from a novel problem of orbit determination, where the objective is to solve an inverse problem for an orbit propagator using images obtained from a simulated sensor. This problem presents several compelling challenges. First, the underlying physics and the forward operator involved are highly nonlinear, which constitutes a primary focus of our work. Second, an orbit can be defined by six orbital elements, while the received data exists in a significantly larger space, such as an image space in our case. Consequently, the forward operator maps from a smaller parameter space to a substantially larger image space. The third aspect of this problem pertains to the challenging nature of obtaining training data, as it requires integrating raw acquisition with non-trivial evaluation and determination of orbit parameters.

Contributions In this paper, we propose a novel hybrid approach that combines both physics-aware and data-driven methodologies by utilizing simulated data. Given the difficulty of obtaining viable training pairs (observations-parameters) for many real-world problems (e.g., orbit determination here), simulations offer a convenient way of providing the missing complementary information. Our objective is to demonstrate the effectiveness of neural networks in handling nonlinear problems while presenting a method that accounts for the forward operator when optimizing for the neural network weights and biases by minimizing the reconstruction error in the observation space. Through our investigation, we will show that leveraging the information provided by physics is sufficient to achieve accurate reconstruction and generalization without requiring training data. In addition, we introduce a hybrid version that balances the trade-off between parameter fidelity and physics (observation) fidelity by minimizing a positive weighted sum of reconstruction errors both in the observation and the parameter spaces. Our method, that we coin Simulation-augmented Physics-Aware Neural Networks (SimPANNs for short) improves performance even further, especially when observations are limited. This is true even when the physical model used in the simulation is only known approximately. To this end, we have carried out experiments on different types of inverse problems (including orbit determination), for different levels of noise in the observations and model approximation, enabling us to draw some preliminary but consistent conclusions on the usefulness of such hybrid models.

2. Related work

2.1. Deep learning for inverse problems

The literature of inverse problems has witnessed a flurry of research activity on methods using deep neural networks. Our goal here is by no means to be exhaustive but rather to discuss the ones most closely related to our work. The interested reader may refer to, e.g., the surveys [24, 25, 13, 14, 26]. The diversity of these methods is motivated by the variety of scenarios that are encountered based on the available data for model training (matched parameters-observations pairs, unpaired parameters-observations, or observation's only), knowledge of the operator to be inverted, or whether the operator is linear or not.

The case where the forward model is fully known is the most extensively studied scenario in the literature, with different approaches depending on the accessibility of the data for model training/optimisation. When paired observations/parameters are leveraged, iterative schemes [24, 6, 27], plug-and-play based methods [4, 6, 25], and unrolling/unfolding are widely used [28, 24, 6]. When unmatched (x, y) pairs are available, two methods are generally used. The first is to construct matched training samples as the parameter set and forward model are known. The second approach involves the use of generative models such as GAN's [29, 30]. Generative models have also proven very fruitful when only observations y are available as first advocated in [7]. Though in most of these works the forward model is linear, extensions to the nonlinear case have appeared, see e.g., [10, 9].

A crucial observation is that in all these methods, the dimensionality of the observation and parameter space is something important. Typically, the data for the observation and the parameters come in the form of a structured signal or image. The discrepancy in dimensionality and structure between the observation space and the parameter space is a distinctive feature of our work. Our focus is to address problems characterized by a significant variation in dimensionality, where the observation space, in the context of our study, corresponds to the image space, while the parameter space is confined to a few number of parameters.

2.2. Physics knowledge in data-driven methods

Incorporating physics knowledge when solving inverse problems using neural networks is not new, and several works have

been (and are being) proposed along these lines; see e.g., the review papers [12, 13, 14] and references therein. Physics-aware neural network-based methods have also been proposed in recent years to solve PDEs [15, 16, 17, 18, 19, 20, 21, 22]. In the context of PDEs, the core idea is to force the output of the neural network evaluated pointwise on the domain of the PDE to comply with the physical model as described by the PDE, including the boundary and initial value conditions, the geometry of the domain, etc. In turn, this provides an implicit regularization mechanism. These ideas have been used to solve inverse problems in PDEs as well [31, 32].

In our method SimPANN, we will rely on ideas similar to the ones used in PANNs[1]. More precisely, we train the parameters of the neural network with a hybrid risk which incorporate the forward model physics not only on the observations but also using simulation-generated data where only an approximation of the underlying nonlinear physics model is known. These are distinctive features with respect to the literature. While the core ideas were introduced in a short conference paper [33], this journal article provides an in-depth exploration, along with numerous novel results.

2.3. Machine learning in orbit restitution context

Our research is primarily focused on applications in astrophysics. Machine learning has made significant progress in revolutionizing this field, providing data-driven approaches to various applications [34, 35], such as orbit restitution and rendezvous problems [36, 37]. The inclusion of physics in orbit restitution problems proves advantageous in several aspects [38]. For example, in [36], neural network predictions are used to refine parameter estimates and correct for errors that traditional methods may overlook, thereby facilitating synergistic integration of data from different sources. Approaches such as those described in [38] reconstruct orbit parameters directly from satellite positions, validated by simulations to embed physical principles in neural networks.

Our proposed method aligns with the framework of initial orbit determination (IOD) without relying on prior assumptions. Indeed, IOD methods are numerous and often tailored to specific types of measurements, such as Time Difference of Arrival (TDOA)[39] or angular measurements[38]. However, for many types of measurements, defining an IOD method is nontrivial or even infeasible. Classical approaches such as the Least Squares Method (LSM) [40, 41] and the Kalman Filter (KF) illustrate these challenges. LSM requires an accurate initial guess for the orbital parameters to ensure convergence toward the correct solution. Similarly, the Kalman Filter, while powerful, not only depends on a good initial guess but also requires a precise covariance matrix to model uncertainties and correctly weigh observational data. Without these prerequisites, both methods may fail to produce reliable results, particularly when dealing with noisy or incomplete measurements.

In our case, the neural network, trained on simulated data, provides the IOD capability, bypassing the limitations of traditional methods by adapting to diverse scenarios. By focusing on indirect measurements of satellite information from sensor outputs, our approach further highlights the potential of machine learning to overcome the inherent challenges of orbit restitution. This capacity to leverage simulated data, while embedding physical principles, underscores the transformative impact of combining machine learning and physics-driven methodologies in astrophysical applications.

[1] Recall the abbreviation physics-aware neural networks.

3. Proposed method

3.1. Preliminaries

Recall the forward problem in (1). Throughout, we assume that f is smooth enough (at least continuously differentiable) suitable for automatic differentiation. $\|\cdot\|$ will denote the euclidian norm, $^\top$ is the matrix transpose, and I is the identity matrix (whose dimension is to be understood from the context).

In the sequel, for a neural network with parameters (weights and biases) $\theta \in \Theta$ and input $y \in \mathcal{Y}$ its output is defined via the mapping

$$\psi : (y, \theta) \in \mathcal{Y} \times \Theta \mapsto \psi(y, \theta) \in \mathcal{X}.$$

Of course, another latent input variable z could be used as input but the choice of y will be discussed shortly.

Finally, we also assume that we have an approximate explicit model of f, sometimes referred to as a digital twin. This model, denoted $\hat{f}$, is obtained by modeling, though imperfectly, the physical phenomena that gave rise to the observations. It will be used later to generate simulated data.

3.2. Forward model in nonlinear inverse problems

Let ψ be a neural network and consider n_o-sample observation dataset $\{y_i : i = 1, \dots, n_o\}$. What we mean by incorporating physical knowledge to invert (1) amounts to solving the following minimization problem of the empirical risk:

$$\min_{\theta \in \Theta} \frac{1}{n_o} \sum_{i=1}^{n_o} \ell_o(y_i, \hat{f}(\psi(y_i, \theta))), \tag{2}$$

where $\ell_o : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ is the loss function in the observation space. This loss function leverages the information provided by the (approximate) physical forward model directly into the training risk. It is also a non-supervised method that relies solely on observations, without any knowledge of the actual parameter vector x_i corresponding to each y_i. Unfortunately, as was observed previously in the literature (e.g., in [42]), when f is not injective, there are infinitely many solutions $\psi(\cdot, \theta)$ which attain zero training error. This is because the forward model f may map multiple input vectors to the same output vector. For example, in the linear case, the action of f is invariant along its null space. This suggests that training a reconstruction network as (2) only from the observed data, without any additional assumptions or constraints, may not be viable. Possible workarounds include explicitly constraining the output of the reconstruction network through regularization (and we are back to the model-based world), or introducing invariances such as in [42]. In the forthcoming section, we will describe an alternative based on exploiting the forward model to simulate input-output pairs. This approach will help to regularize the training process and make it more immune to the non-injectivity of f.

3.3. Simulation-augmented training

In many areas of science, obtaining pairs of input (parameters)-output (observations) training data, can be a significant challenge. This can be due to various reasons, including that data are difficult or expensive to acquire. This is for instance the case in large instruments in physics. Furthermore, even if such pairs of data can be acquired, they are available only in limited quantity, which often impedes the use of data-intensive machine learning approaches.

There are however situations where even if such data is unavailable, it is possible to artificially generate input-output pairs by leveraging knowledge of the forward model given in (1), even if the latter is only approximately known. This involves generating n_s parameter/input vectors x_i sampled from the range of possible input values in the model or based on the known distribution of input data. We then compute the corresponding simulated observations $y_i = \hat{f}(x_i)$, i.e. without noise. It is important to recall that the forward model $\hat{f}$ serves anyway as an approximation of the genuine f it represents, and the simulated observation can only be considered as a perturbed version of the unknown observation due to model imperfections.

Summarizing our discussion above, we propose to train a neural network ψ by replacing (2) with

$$\min_{\theta\in\Theta} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_{\lambda_i}(x_i, y_i, \theta), \tag{3}$$

where $n = n_o + n_s$, n_o (resp. n_s) is the number of observations-only (simulated) and data, and

$$\mathcal{L}_\lambda(x, y, \theta) = \lambda \ell_o(y, \hat{f}(\psi(y, \theta))) + (1 - \lambda)\ell_s(x, \psi(y, \theta)), \tag{4}$$

where $\ell_s : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ is the loss in the parameter space. Here $\lambda_i \in]0, 1[$ balances between the two losses: fidelity to the observations and parameter reconstruction error. For simulated data, we set $\lambda_i = \lambda > 0$, while we use $\lambda_i = 0$ for observation-only data since x_i is unknown in this case. In plain words, only the first term, which focuses on fidelity to observations comes into play for observed data. It is anticipated that the ratio n_o/n_s will play an important role in the results, and thus, its influence will be thoroughly examined through experimental studies.

It is worth observing that when ℓ_s is the quadratic loss, the corresponding term in (4) can be viewed as a quadratic Tikhonov regularization on the network output, which is a standard regularization technique in inverse problems [23]. When both ℓ_o and ℓ_s are quadratic and the forward model is linear, i.e. $\hat{f}$ is an $m \times n$ matrix A, it is not difficult to show[2] that $\mathcal{L}_\lambda$ specializes to the convenient form

$$\mathcal{L}_\lambda(x, Ax, \theta) = \mathrm{I} - \lambda(\mathrm{I} - A^\top A)^{1/2}(x - \psi(y, \theta)))^2,$$

which again reveals the regularizing effect of the second term in (4).

[2] Observing in particular that $(\mathrm{I} - \lambda(\mathrm{I} - A^\top A))$ is symmetric positive-definite.

3.4. Deep unrolling/unfolding

The aim of deep unrolling, a.k.a. deep unfolding, is to convert an iterative algorithm, generally an optimization solver, into a neural network architecture by designing each layer to resemble a single iteration. Unrolling was originally proposed in [43] to learn iterative soft thresholding algorithm (ISTA) for sparse recovery. Deep unrolling have since been applied to various applications and in particular solve various inverse problems. In this context, unrolling offers a way for learned iterative schemes to integrate a knowledge model for how data are generated into a data-driven method for reconstruction. The latter is realized by unrolling a model-driven iterative scheme stopped after a finite number of iterations, where nonlinear operators in the updates are typically parameterized by neural networks. Thus, unrolling can be seen as an alternative way to (3) to account for forward model knowledge.

The application of unrolling to design a model-aided deep network is based on the following steps: (i) identify an iterative optimization algorithm which is useful for the problem at hand; (ii) fix the maximum number of iterations K in the optimization algorithm; (iii) design the neural networks to imitate the nonlinear parts of each iteration; (iv) train the overall resulting architecture end-to-end.

As an illustrative example that we will use here, we consider unrolling of the proximal gradient algorithm up to iteration K, where the proximal mapping is replaced by a neural network. For the forward model (1) with differentiable observation loss ℓ_o, this corresponds to the update

$$\begin{aligned} x^{k+1} = \psi x^k - \mu J_f(x_k)^\top \nabla \ell_o(y, f(x^k)), \theta^k, \\ \mu > 0, k = 1, \ldots, K, \end{aligned} \tag{5}$$

where J_f is the Jacobian of f. The case ψ is an auto-encoder was considered for instance in [6]. The step-size μ modulates the amount of correction applied at each step and its role is to ensure sufficient descent while preventing the iterative scheme from diverging. It is either chosen or learned to make the iteration mapping a contraction.

Computational feasibility of unrolling is directly tied to the number of iterates K and their size. These time and memory limitations impose that K is reasonably low and that the parameters $\theta^k \equiv \theta$ of the neural networks are shared between the iterates. If these parameters are shared and the step-size is fixed, the network ψ is trained in our case by solving

$$\min_{\theta\in\Theta} \frac{1}{n} \sum_{i=1}^{n} \lambda_i \ell_o(y_i, \hat{f}(x_i^K(y_i, \theta))) + (1 - \lambda_i)\ell_s(x_i, x_i^K(y_i, \theta)), \tag{6}$$

where $x_i^K(y_i, \theta)$ is the K-th iterate in (5) with input observation y_i.

$$x^{k+1} = \psi \hat{f} x^k - \mu J_f(x_k)^\top \nabla \ell_o(y, f(x^k), \theta.$$

With $\mu = 0$ and $K = 1$, one recovers SimPANNs. Moreover, as regularization does not really make sense in our applications, it makes more sense to regularize the reconstructed observation. This will also be tested and we will assess based on the results obtained.

4. Experimental results

As we argued in the introduction, our framework tackles the difficult case of nonlinear with little structure in the parameter space to exploit. In order to evaluate it effectively, it is necessary to address the limitations of existing experimental settings by considering problems relying on nonlinear forward operators. Specifically, we aim to examine scenarios where the dimension of $\mathcal{X}$ can be much smaller than that of $\mathcal{Y}$, i.e. $d \ll m$.

It is important to stress that despite the apparent simplicity as we have more equations than unknowns, the presence of nonlinearities breaks this intuition which makes our problems quite challenging. Indeed, the forward physical model in our setting comes with symmetries that yield non-trivial equivalence classes.

In what follows, we present a series of problems that we have developed for this purpose, with different degrees of difficulty and closeness to our motivating application: orbit determination.

4.1. Problems and datasets

4.1.1. BlendMNIST problem

Inspired by RotNet [44], we present a novel problem with the goal of accurately inferring a set of rotation angles from a single input image. In this unique problem formulation, the input image is generated by summing different MNIST images [45], where each individual image is assigned a randomly chosen rotation angle (see Figure 1 for an illustration). We coin this Blended Weighted Sum of MNIST digit images (abbreviated BlendMNIST).

We take a fixed set of 28×28-pixel images $\{\mathcal{I}_k\}_{k=1}^p$. The forward operator in this case takes the form

$$\begin{array}{rccc} f & : & [0, 2\pi[^p & \to & \mathbb{R}^{28\times 28} \\ & & (\alpha_1, \dots, \alpha_p) & \mapsto & \frac{1}{p}\sum_{k=1}^p \mathcal{I}_k^{\alpha_k}, \end{array} \tag{7}$$

where $\mathcal{I}^\alpha$ is the rotated image whose value at pixel $\mathbf{p}$ is

$$\mathcal{I}^\alpha[\mathbf{p}] = \mathscr{I}[\mathcal{I}]\mathscr{R}_{-\alpha}(\mathbf{p}).$$

$\mathscr{R}_{-\alpha}$ is the rotation operator of angle $-\alpha$ around the image center, and $\mathscr{I}[\mathcal{I}](\cdot)$ is the interpolation operator applied to image $\mathcal{I}$ evaluated at its argument. We choose bilinear interpolation in our experiments. Strictly speaking, rotation can take pixels outside the domain of the image, and we set those falling outside the domain to 0. This is a valid choice in our case with MNIST as the background of the MNIST digit images is 0, and rotation around the center will not make any part of the digits fall outside the domain.

Clearly, the parameter space $\mathcal{X} = [0, 2\pi[^p$ is p-dimensional with p much lower that the observation space, i.e. $p \ll 28^2$. However, the forward operator is highly nonlinear in the angles which are our unknowns to be recovered.

In our experiments, we take $p = 10$ and the images $\{\mathcal{I}_k\}_{k=1}^{10}$ correspond to the digits 0 to 9 taken from the MNIST database (see Figure 6). By choosing a value of p as such, we aim to create a sufficiently challenging problem that cannot be easily solved through brute force methods in a reasonable time and justify the use of neural network methods.

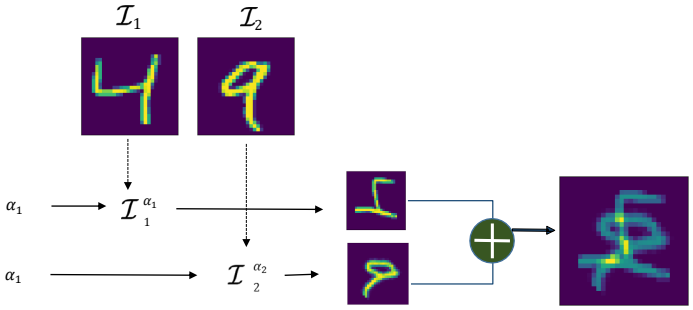

Figure 1: Example of f with $p = 2$ for the BlendMNIST problem.

4.1.2. Sparse BlendMNIST problem

This problem, dubbed S-BlendMNIST in the following, is a more challenging extension of BlendMNIST to a scenario where only a subset of the potential images in the mixture is activated. The corresponding forward model takes the form

$$\begin{array}{rccc} f & : & [0, 2\pi[^p \times \{0,1\}^p & \to & \mathbb{R}^{28\times 28} \\ & & (\alpha_1, \dots, \alpha_p, w_1, \dots, w_p) & \mapsto & \frac{1}{p}\sum_{k=1}^p w_k \mathcal{I}_k^{\alpha_k}. \end{array} \tag{8}$$

This model then explores the situation where certain model parameters belong to a discrete space. The objective is then twofold: infer the angle associated with each image in the mixture, and determine a binary variable indicating the presence or not of each image within the mixture.

4.1.3. BlendMNIST with approximate model

In various fields of application, it is common that the forward operator to be approximately known, and this deviation from a perfect model entails an additional modeling error in the observations. This discrepancy serves as a motivation for us to extend BlendMNIST to a scenario where the direct model is only approximately known (we coin this scenario AM-BlendMNIST). This extension enables us to assess the robustness of the proposed methods in presence of modeling errors in the forward model.

Specifically, the AM-BlendMNIST builds upon BlendMNIST with the modification that the centre of rotation varies. The actual forward operator is then defined as

$$\begin{array}{rccc} f & : & [0, 2\pi[^p \times [0, 27]^2 & \to & \mathbb{R}^{28\times 28} \\ & & (\alpha_1, \dots, \alpha_p, \upsilon) & \mapsto & \frac{1}{p}\sum_{k=1}^p \mathcal{I}_k^{\alpha_k, \boldsymbol{\upsilon}}, \end{array} \tag{9}$$

where $\mathcal{I}^{\alpha,\boldsymbol{\upsilon}}$ is the rotated image around center position $\boldsymbol{\upsilon}$ whose value at pixel $\mathbf{p}$ is

$$\mathcal{I}^{\alpha,\boldsymbol{\upsilon}}[\mathbf{p}] = \mathscr{I}[\mathcal{I}]\mathscr{R}_{-\alpha} \circ \mathscr{T}_{-\boldsymbol{\upsilon}}(\mathbf{p}).$$

$\mathscr{R}_{-\alpha}$ and $\mathscr{I}$ are as explained in Section 4.1.1 and $\mathscr{T}_{-\boldsymbol{\upsilon}}$ is the translation operator of parameter $-\boldsymbol{\upsilon}$.

In AM-BlendMNIST, the translation is only included in the actual forward operator when generating the observation data, but is not accounted for in the approximate forward model during training. Put formally, the approximate forward operator $\hat{f}$ remains the same as in the BlendMNIST case, i.e. the one in

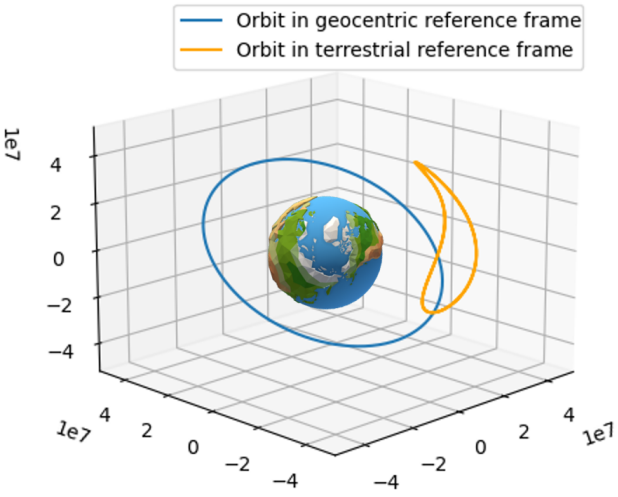

Figure 2: Orbit determination: representation of an orbit in multiple reference frames. In this example, the orbit presented have an inclination of 45 degrees, an eccentricity of 0.4 and a 42164×10^3 m apogee.

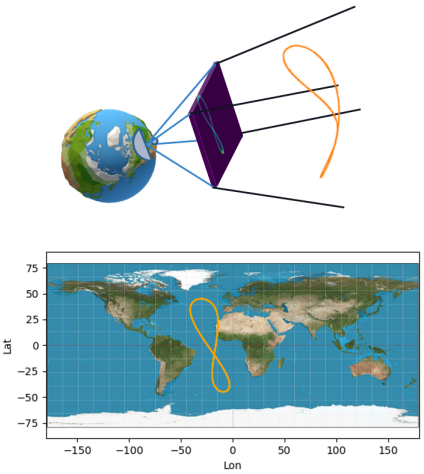

Figure 3: Orbit determination: simulation of the orbit projection, where the orbit is the one shown also on Figure 2. Top: simulated sensor with the orbit projected on it. Bottom: the ground track orbit projection.

(7). In the experiments, the dataset used for evaluation is generated by applying a translation $\boldsymbol{v}$, and thus $\sigma_{model} = \boldsymbol{v}$ can be seen as the modeling error magnitude.

4.1.4. Orbit determination problem

As argued in the introduction, orbit determination is our main motivating application and it encompasses all challenges put forward in this paper. In orbit determination, the forward operator consists in projecting an orbit in the Terrestrial Reference Frame (TRF) onto an image, akin to a ground track orbit projection. The data received are the projection of an orbit on a sensor placed on earth (see Figure 2 and 3).

For this problem, we opt to represent each orbit using 3 Keplerian parameters: inclination, eccentricity, and periapsis. This choice leads to a wide range of images and results, while avoiding trivial equivalence classes in the parameter set. We shall now explain in a bit more detail our (approximate) physical model.

Our physical model for the orbit projection problem can be separated in two components:

- The first one is the physical model that simulates the position of the orbit, known as the orbital propagator. Different orbital propagators may be used based on the required level of accuracy. Propagators like J2 [46] and SGP [47, 48] offer high precision, but are more intricate and computationally demanding. Conversely, simpler models such as the two-body propagator [49] can be easily implemented and are optimized for GPU usage. This is the model employed in our experiment.

- The second crucial component of the physical model encompasses the sensor measurement system. The sensor is a camera placed on earth which receives the projection of satellite positions onto the camera's field of view in a way akin to the MVP (Model View Projection) technique [50].

  In our specific scenario, the camera is positioned across earth in a uniformly randomized manner.

Let $\mathrm{Pos}^{\mathrm{ECEF}}_{e,i,\omega} : \mathbb{R}_+ \to \mathbb{R}^3$, the position of a satellite in the Earth-Centred Earth-Fixed (ECEF) reference system, parametrized by the triple (e, i, ω). Let $\mathrm{MVP} : \mathbb{R}^3 \to \mathbb{R}^2$, the MVP linear operator which maps the world space to the camera surface. The camera point spread function is considered as a 2D-Gaussian of standard deviation δ, that we denote φ_δ. Thus, the forward operator in this case reads

$$f \quad : \quad [0,1[\times[0,2\pi[\times[0,2\pi[\quad \to \quad \mathbb{R}^{64\times 64} \qquad (10)$$

where the received signal at pixel $\mathbf{p}$, is given by

$$f(e,i,\omega)[\mathbf{p}] = \int_{\left[-\frac{1}{2},\frac{1}{2}\right]^2} \int_0^{\frac{2\pi a^{3/2}}{\sqrt{GM}}} \varphi_\delta \mathbf{p} - \mathbf{x} + \mathrm{MVPPos}^{\mathrm{ECEF}}_{e,i,\omega}(t)\mathrm{d}t\mathrm{d}\mathbf{x},$$

where M is the mass of the earth, G is the gravitational constant and a is the elliptical semi-major axis of the orbit.

The approximate operator we take for simulations not only accounts for the fact that f is itself an actual approximation, but also for the error induced by discretizing the above forward model. For instance, to approximate the integrals, we use a standard Riemann sum over rectangles with small enough size.

4.2. NN models, training losses and experimental settings

4.2.1. BlendMNIST problem

The neural network employed for addressing the BlendMNIST problem is a feed-forward architecture, specifically a multi-layer perceptron, which does not incorporate dropout layers nor batch normalization (see Figure 4). The rationale behind this design choice is to deliberately over-parameterize the network for the given problem, ensuring a sufficiently large latent space. By doing so, we aim to provide the model with ample

Figure 4: BlendMNIST problem: the NN model used is a multi-layer perceptron with 5 hidden layers. It outputs the rotation angles for an input mixture image in the form of the BlendMNIST dataset with $p = 10$.

capacity to effectively capture the complexities inherent in the BlendMNIST dataset.

In the case of the unrolled scheme, the neural network architecture is the same as the previous. With a number of iterations $K = 5$ and shared parameters across the unrolled iterates.

The importance of the choice of the training loss is widely acknowledged as it can have a significant impact on performance and convergence. It is in particular crucial to choose the one adapted to the data underlying geometry. In our case, the neural network predicts a list of angles that lie in $[0, 2\pi[$. Thus, the distance used in our case is the shortest difference between the predicted angle $\tilde{\alpha}$ and the ground truth angle α:

$$\text{dist}_{angle}(\alpha, \tilde{\alpha}) = \pi - ||\tilde{\alpha} - \alpha| - \pi| \tag{11}$$

Thus, the loss between the predicted angles $\{\tilde{\alpha}_k\}_{k=1}^{10}$ and the true ones $\{\alpha_k\}_{k=1}^{10}$ is given by

$$\ell_{angles}(\{\alpha_k\}_{k=1}^{10}, \{\tilde{\alpha}_k\}_{k=1}^{10}) = \frac{1}{10}\sum_{k=1}^{10} \text{dist}_{angle}(\alpha_k, \tilde{\alpha}_k).$$

We point out that though $\mathcal{L}_{angles}$ is not differentiable, we will still apply autodifferentiation to it and this still makes sense in view of the results in [51].

To recap, for the BlendMNIST problem, we solve (3)-(4) with ℓ_o the quadratic loss and $\ell_s = \ell_{angles}$. For the unrolled scheme, the neural network is trained by solving (6) with the same losses just described.

4.2.2. S-BlendMNIST problem

In this problem, in addition to the rotation angles, the neural network must also output the weight w_k of each component in the mixture, recalling that the true w_k's are valued in the discrete set $\{0, 1\}$. To ensure this, we use the same neural network architecture as in the BlendMNIST problem, with the difference of the last layer where we added a sigmoid activation function to each output corresponding to a weight (see Figure 5). The temperature of the sigmoid is decreased gradually over epochs reaching a final temperature of 0.01. This setup ensures that the output is close to a binary variable at the end of the training while maintaining a smooth activation whose derivative can be back-propagated without jeopardizing the autodifferentiation process.

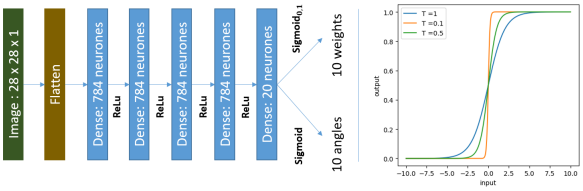

Figure 5: S-BlendMNIST problem: the NN model used for S-BlendMNIST is the same as the one in the BlendMNIST problem, with the difference of the last layer that outputs both the 10 angles and 10 weights, and the latter are obtained by applying a sigmoid activation on the output layer that has an eventual temperature parameter of 0.01.

For the S-BlendMNIST problem, we solve (3)-(4) with ℓ_o the quadratic loss, and ℓ_s as

$$\ell_s\{\alpha_k, w_k\}_{k=1}^{10}, \{\tilde{\alpha}_k, \tilde{w}_k\}_{k=1}^{10} = \sum_{k=1}^{10}(\tilde{w}_k - w_k)^2 + \frac{1}{\sum_{k=1}^{10} w_k}\sum_{k=1}^{10} w_k \text{dist}_{angle}(\alpha_k, \tilde{\alpha}_k).$$

The part of the loss measuring the error in angles takes into account only the active sources in the mixture.

4.2.3. Orbit determination problem

In this case, the observation space loss ℓ_o is again chosen as the quadratic loss. For the loss ℓ_s, it involves three terms, one for each of the three orbit parameters (e, i, ω):

- The eccentricity loss: the eccentricity of the orbit is bounded between 0 and 1, we have chosen to use the quadratic loss.
- The inclination and periapsis argument losses: these parameters being angles, the corresponding loss is given by dist_{angle}, see (11).

We use the same neural network architecture as for the BlendMNIST problem. The input has now dimension 64^2 as the images in this case are 64×64-pixel, and the output layer returns the three parameters of the orbit.

4.2.4. Implementation details

Neural network training in the context of inverse problems can be computationally expensive due to the fact that the physical model must be back-propagated along the neural network weights. To reduce this cost, we have proposed a GPU implementation of the physical model involved in each of the proposed problems. This implementation uses an analytical formulation of the Jacobian $J_{\hat{f}}$ that takes care of reducing the memory/computational footprint of autodifferentiation. For training, we use Adam [52] with a linear decaying step size (learning rate) of 10^{-3} to 10^{-5} and bias parameters 0.9 and 0.999.

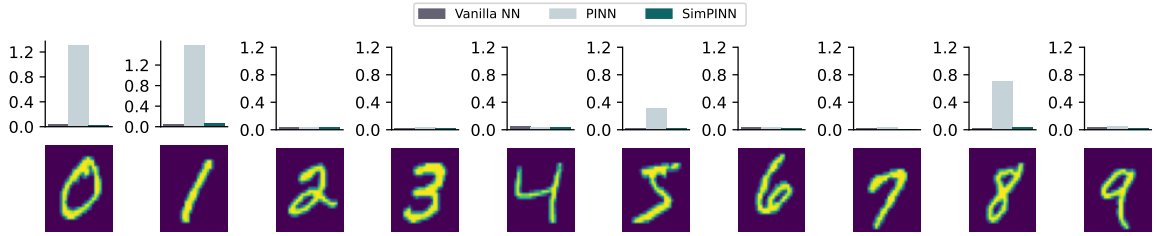

Figure 6: BlendMNIST problem: prediction error of predicted angle for each digit image.

4.3. Results and discussion

In this section, we describe and discuss our experimental results. In particular, we will evaluate the performance of SimPANNs with $\lambda \in]0, 1[$ in (4), that obtained by setting $\lambda = 1$, i.e. when simulation data are dropped (we coin the corresponding method PANNs again) and unrolling as detailed in Section 3.3.4. Additionally, we compare these two methods with a Vanilla-NN learning approach which is trained by minimizing the angle loss in a supervised way, i.e. with the actual true angles.

In our experiments, the parameter λ is not trivial to fix and find. In our case, we chose the best λ based on a cross-validation set.

4.3.1. Experiments on BlendMNIST

We remind from Section 4.4.1.1 that the objective of the BlendMNIST problem is to determine the rotation angles of 10 digit images solely from their linear mixture. Here, it is assumed that the direct model is perfectly known, i.e. $\hat{f} = f$ where the latter is given in (7).

Method	Image error $\times 10^{-4}$	Angles error (rad)
Vanilla NN	3.3 (2.1)	0.0310 (0.026)
PANNs	7.2 (4.3)	0.4130 (0.2635)
SimPANNs $\lambda = 0.01$	**2.6 (2.3)**	**0.0250 (0.0219)**
Unrolled SimPANNs (K=5)	3.5 (1.4)	0.0375 (0.0219)

Table 1: BlendMNIST problem: image and angles reconstruction errors (average (standard deviation)) over a test set for all proposed methods. The hyperparameter λ for SimPANNs is optimized using cross validation.

Methods comparison. The average reconstruction errors and their standard deviation (in parentheses) are reported in Table 1. The first notable observation is that though the PANNs method achieves a low image reconstruction error however the angle error remains relatively high. This is expected as PANNs is only trained with the observation-domain loss, and thus the predicted parameters may fall into an equivalence class. To confirm this, Figure 6 illustrates the error associated with the parameters for each number in the image. It is clear that the images depicting the digits 0, 1, and 8 have a symmetry characterized by a rotation of π. The digit 5 also suffers a similar phenomenon, though to a less extent as its symmetry is less marked. For the other digits, the reconstruction accuracy achieved by the PANNs method is comparable to the two supervised methods.

Strictly speaking, these symmetries entail that the PANNs can only predict the angles up to an equivalence class. This is unavoidable unless additional a priori information is included to bias the prediction towards the desired solution. For instance, explicit guidance can be provided by employing the SimPANNs method thanks to the angles loss. It is clear from Table 1 achieves the lowest reconstruction error not only on the angles but also in the image domain. Figure 7 showcases several examples of reconstructions in the observation domain achieved through the SimPANNs method for different mixtures. SimPANNs then appears as taking the best of both worlds providing the most effective NN-based technique for this problem.

We have also observed in our experiments that SimPANNs notably reduces training time compared to the PANNs method. We conjecture that this is due to the regularizing effect induced by the loss ℓ_s when training SimPANNs entailing faster convergence. As shown in Table 2, NN-based methods are very fast at the inference step. For comparison purposes, we have also implemented a brute force grid search method as well as an evolutionary algorithm to solve $\min_{\{\alpha_k\}_{k=1}^{p}} y - f(\alpha_1, \ldots, \alpha_k)$. Though the reconstruction errors were low, these methods have an excessive computational burden, making them impractical for use in this context. We will not explore them further in the rest of this paper

We experimentally observed that the unrolled version of the SimPANNs method does not give better performance than the standard SimPANNs. In addition, the unrolled version is more difficult to train, since the gradient magnitude depends on the number of unrolling steps (here $K = 5$): we observed that the gradients can sometimes explode or vanish, even in the presence of normalizing layers. Therefore, we have not retained

Method	Device	Inference time
NN-based	**GPU**	$\mathbf{4.42 \times 10^{-4}}$s
	CPU	5.28×10^{-4}s
Evolutionary algorithm	CPU	50.3s
Grid search	CPU	> 1h

Table 2: BlendMNIST problem: Inference time for various methods. For this experiment, the CPU used is an Intel(R) Xeon(R) Gold 6226R and the GPU is an Nvidia A30.

$\sigma_{model} = 0$	n_o					
n_s	0	100	1000	12500	25000	50000
0	-	1.55	1.55	1.51	1.48	1.46
100	1.38	1.36	1.40	1.36	0.87	1.34
1000	1.13	1.09	1.00	0.96	0.89	0.81
12500	0.38	0.38	0.40	0.32	0.29	0.26
25000	0.20	0.19	0.17	0.16	0.14	0.14
50000	0.10	0.10	0.10	0.10	0.10	**0.08**

$\sigma_{model} = 1$	n_o					
n_s	0	100	1000	12500	25000	50000
0	-	1.54	1.53	1.33	1.41	1.22
100	1.25	1.24	1.19	1.40	1.35	1.22
1000	1.14	1.16	1.04	1.21	1.32	1.12
12500	1.18	1.20	1.07	0.34	0.65	0.54
25000	1.17	1.15	1.05	0.44	0.53	0.24
50000	1.04	1.00	0.95	0.46	**0.14**	0.16

$\sigma_{model} = 2$	n_o					
n_s	0	100	1000	12500	25000	50000
0	-	1.52	1.51	1.50	1.51	1.50
100	1.54	1.51	1.51	1.52	1.50	1.49
1000	1.54	1.53	1.50	1.51	1.51	1.48
12500	1.53	1.54	1.50	1.45	1.50	1.44
25000	1.45	1.50	1.46	1.45	1.49	1.52
50000	1.45	1.45	1.45	1.46	1.49	1.48

Table 3: BlendMNIST problem: reconstruction error on angles (in radian) of the SimPANNs method as a function of the number of observations-only data (n_o), the number of simulated data (n_s) and the forward model error level.

this method for the following experiments.

Simulation-augmented training. Table 3 (resp. 4) shows the reconstruction error on angles (resp. observation space) of the SimPANNs method for various sizes of simulated data (n_s) and observation-only data (n_o), and different modeling error levels σ_{model}. $n_s = 0$ corresponds to the PANNs method. The color code indicates increasing error from light lemon to red.

In this paragraph, we restrict our discussion on the case $\sigma_{model} = 0$, and robustness to errors in the forward model will be discussed in a dedicated paragraph hereafter. Several observations are in order. Increasing n_s clearly improves performance regardless of the number of observation-only data n_o (even when $n_o = 0$). For fixed n_s, increasing n_o also decreases the reconstruction error, though less notably. Increasing both n_s and n_o leads to the best performance, but as expected, the influence of n_s is more prominent. Learning with the simulation-aware loss ℓ_s as in SimPANNs reduces the angles reconstruction error by 13 and the observation error by 2.

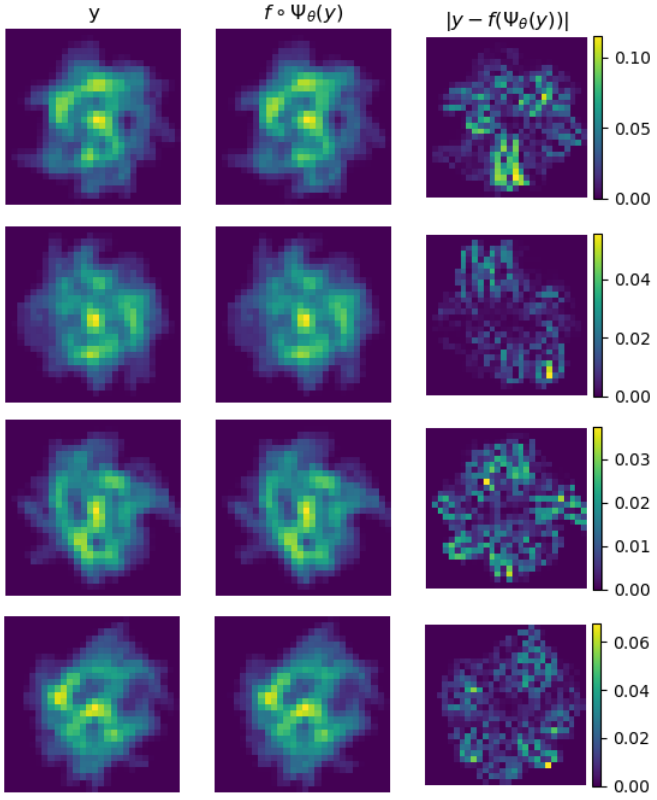

Figure 7: BlendMNIST problem: examples of reconstructions in the observation domain achieved through the SimPANNs method for different mixtures.

Robustness to noise. One significant concern in inverse problems is their resilience to noise on the observations. This is all the more important as neural network methods are used here. More specifically, we aim to assess the extent to which the proposed PANNs and SimPANNs approaches can effectively handle noise in observations for various types and levels of noise.

For the results depicted in Figure 8, three noise types are investigated, namely Gaussian noise, salt-and-pepper noise and Poisson noise, which are commonly encountered in imaging problems. SimPANNs is trained with $n_o = 1E5$ observations-only data and $n_s = 1E5$ simulated data. Though these noises have different properties, we use the popular Peak Signal-to-Noise Ratio (PSNR) to quantify the corresponding noise level. Figure 8 reveals that the SimPANNs method is more robust to noise, whatever the noise type, and its robustness properties are better at high noise levels compared to PANNs and Vanilla NN. Even with a PSNR of 15dB, the angle error remains at an acceptable level. The PANNs method yields lowest results among the three methods underscoring again the role of simulation-generated data. It is observed also that Poisson noise leads to the highest reconstruction error when compared to salt-and-pepper (S&P) or Gaussian noise. This discrepancy can be attributed to the fact that the Poisson noise is signal-dependent, and the highest noise concentrates on areas where the signal is strongest, which, in the case of BlendMNIST problem, is where

$\sigma_{model} = 0$	n_o					
n_s	0	100	1000	12500	25000	50000
0	-	0.96	0.70	0.48	0.46	0.39
100	1.00	1.02	1.00	0.76	0.75	0.50
1000	0.92	0.90	0.86	0.82	0.77	0.68
12500	0.43	0.42	0.43	0.38	0.37	0.34
25000	0.31	0.31	0.30	0.27	0.20	0.21
50000	0.14	0.14	0.14	0.14	0.13	**0.11**

$\sigma_{model} = 1$	n_o					
n_s	0	100	1000	12500	25000	50000
0	-	1.64	1.38	0.97	0.92	0.87
100	1.85	1.79	1.64	1.02	0.91	0.84
1000	1.76	1.71	1.62	1.14	0.97	0.85
12500	1.66	1.61	1.53	0.82	0.87	0.63
25000	1.56	1.55	1.49	0.79	0.73	0.73
50000	1.49	1.47	1.37	0.77	**0.62**	0.58

$\sigma_{model} = 2$	n_o					
n_s	0	100	1000	12500	25000	50000
0	-	2.13	2.13	1.50	1.45	1.45
100	2.13	2.13	2.16	1.58	1.50	1.45
1000	2.12	2.16	2.00	1.56	1.49	1.41
12500	2.07	2.13	1.88	1.51	1.49	1.38
25000	1.90	1.77	1.49	1.49	1.43	1.38
50000	1.88	1.56	1.41	1.41	1.41	**1.40**

Table 4: BlendMNIST problem: reconstruction error on the observations of the SimPANNs method as a function of the number of observations-only data (n_o), the number of simulated data (n_s) and the forward model error level.

Method	Image error	Angles error (rad)	Weights error
Vanilla NN	0.001 (0.001)	0.037 (0.032)	0.0031 (0.0097)
SimPANNs	**0.001** (**0.001**)	**0.032** (**0.027**)	**0.0001** (**0.0011**)
PANNs	0.001 (0.001)	0.087 (0.134)	0.020 (0.045)

Table 5: S-BlendMNIST problem: errors (average (standard deviation)) on the observations, angles and weights over a test set for all NN-based methods. SimPANNs is trained with $\lambda_{pred} = 0.1$ and $\lambda_{reconst} = 0.9$.

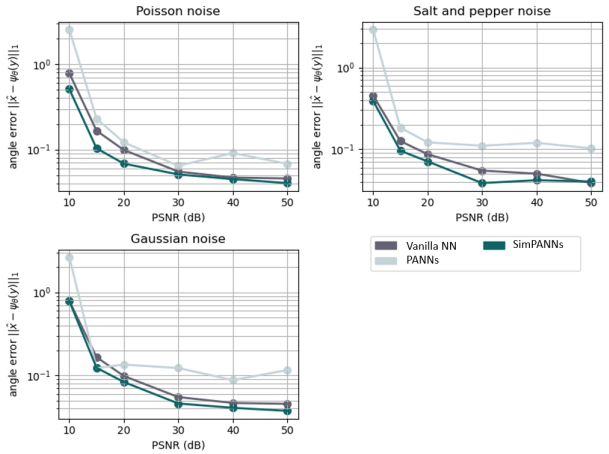

Figure 8: BlendMNIST problem: reconstruction error on angles vs the noise level for various noise types. The error is computed for the Vanilla NN, PANNs and SimPANNs methods.

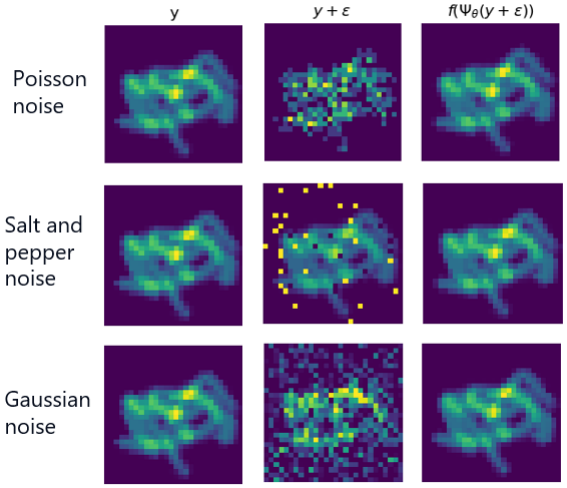

Figure 9: Noisy BlendMNIST problem: examples of reconstructions in the observation domain achieved through the SimPANNs method for different mixtures and various types of noise (PSNR=15dB).

the critical information is located. Figure 9 shows several examples of reconstruction obtained using the SimPANNs method under different types of noise.

4.3.2. Experiments on S-BlendMNIST

Recall that in this problem, the task at hand is to infer not only the rotation angles but also weights of the images in the observed mixture image, where the weights are binary.

The results are summarized in Table 5. Again, the SimPANNs method leads to the best results. A few examples of reconstructed images in this case are also shown in Figure 10 for visual illustration.

4.3.3. Experiments on AM-BlendMNIST

We now turn to the AM-BlendMNIST in which the forward model is assumed to be only known approximately: a translation of the rotation centre is applied to create the observations, but this translation is not included in the approximate forward model $\hat{f}$ that is used for the training. This allows to assess robustness of the methods when there is a mismatch between the true forward physical model and the one used for training.

From Tables 3 and 4, one can see that the performance of SimPANNs degrades as the noise level increases. Reconstruction in the observation space is more stable while that on angles degrades more rapidly. Increasing n_s and n_o for $\sigma_{model} = 1$ still improves recovery performance. We suspect that training $n_o = 0$ overfitting and does not give an acceptable error on the angles. On the other hand, training with $n_s = 0$ from the lack of regularization.

For $\sigma_{model} = 2$, the error on the angles stagnates presumably because the modeling error become overwhelming while the inverse problem is severely ill-posed. Nevertheless, the error in the observation domain is still acceptable and still decreases by

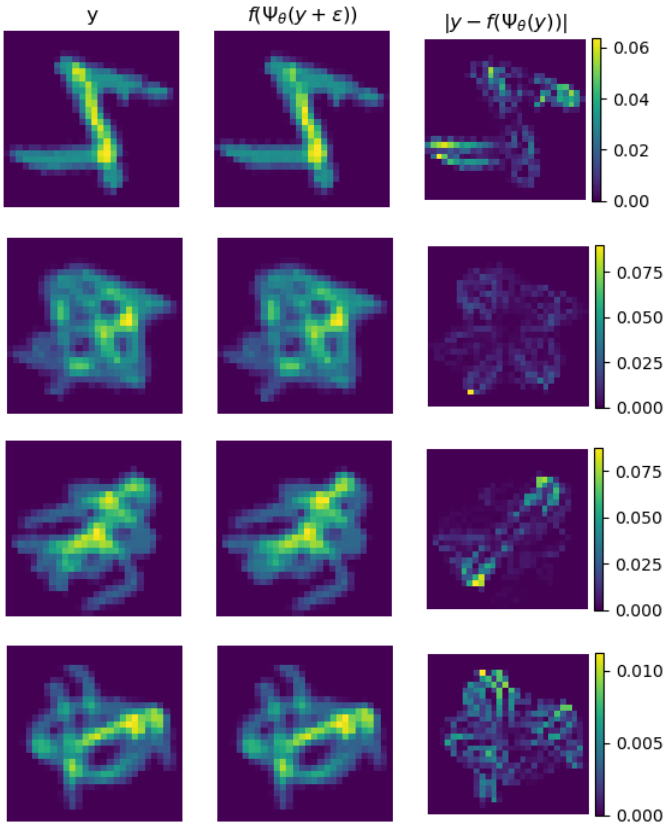

Figure 10: S-BlendMNIST problem: examples of reconstructed mixtures with the SimPANNs method.

ArgPer($\times 10^{-5}$)		n_o				
		0	1000	10000	20000	40000
n_s	0	-	2.4541	1.2970	0.9947	0.9081
	1000	2.0546	2.3412	1.1546	0.9731	1.1742
	10000	1.1470	2.1942	0.8591	0.7854	1.0644
	20000	0.8401	0.7721	0.6924	0.6571	0.6431
	40000	0.4541	0.4412	0.4201	0.4121	0.4212

Eccentricity($\times 10^{-5}$)		n_o				
		0	1000	10000	20000	40000
n_s	0	-	1.5587	1.4424	1.1232	0.8638
	1000	1.3545	1.5225	0.7542	1.1036	0.8452
	10000	1.6556	1.5245	0.7054	1.1023	0.7214
	20000	1.4684	1.0845	1.2781	0.6306	0.6251
	40000	0.3895	0.3856	0.3776	0.3435	0.2861

Inclination		n_o				
		0	1000	10000	20000	40000
n_s	0	-	0.0132	0.0021	0.0009	0.0004
	1000	0.0134	0.0114	0.0017	0.0012	0.0003
	10000	0.0022	0.0028	0.0024	0.0008	0.0007
	20000	0.0014	0.0007	0.0007	0.0007	0.0005
	40000	0.0002	0.0002	0.0002	0.0001	0.0001

Image domain		n_o				
		0	1000	10000	20000	40000
n_s	0	-	1.3243	1.1667	0.8742	0.3023
	1000	1.4581	0.7064	0.1314	0.0989	0.0852
	10000	0.2467	0.1510	0.1394	0.0920	0.0955
	20000	0.0966	0.0831	0.0968	0.0750	0.0701
	40000	0.0304	0.0305	0.0298	0.0271	0.0251

Table 6: Errors of SimPANNS for various parameters, using a test set comprising solely observations.

increasing n_s and/or n_o.

4.3.4. Experiments on the orbit determination problem

The objective of this work is to estimate orbital parameters from an image representing the projection of an orbit in the terrestrial reference frame, captured by a ground-based camera.

As summarized in Table 6, the proposed SimPANNs method effectively integrates information from both observational and simulated data. It consistently outperforms the PANNs method in reconstructing orbital parameters, with PANNs corresponding to the row where $n_s = 0$. Performance improves as the quantity of observational and simulated data increases, achieving an average reduction of orbital parameter errors by a factor of 2 compared to PANNs. Furthermore, SimPANNs provides a tenfold reduction in reconstruction error for the orbit's projected image. Figure 12 presents selected orbit projection images along with their reconstructions generated by SimPANN.

Incorporating orbital physics into the forward operator is crucial for improving image reconstruction in this context. Because orbital dynamics significantly increases the system's sensitivity, even minor variations in parameters can lead to large deviations in the satellite's trajectory, resulting in substantial differences in the projected image (i.e the orbit positions). Consequently, images with nearly identical parameters may display pronounced differences. In this scenario, the loss term ℓ_o plays a key role in capturing high-gradient variations that ℓ_s alone cannot account for, allowing the network to better handle the challenges introduced by these dynamics.

As shown in Figure 11, the model remains robust across most values of λ. However, when λ is too large, the physics-based correction overcompensates and can lead to poor solutions (due to equivalence classes and local minima). In our experiments, the best results are achieved with $\lambda = 0.6$, corresponding to a mean positional error of 36 km. Accordingly, this approach can be viewed as an initial orbit determination (IOD) technique that can subsequently be refined with methods such as least squares minimization (LSM). Furthermore, because this method only requires running a forward pass through the neural network, the inference time for obtaining an estimation is on the order of a few milliseconds (see Table 2), which makes it computationally efficient.

5. Conclusion

In an extensive series of experiments, this work has analyzed and shed light on the capability and robustness of three neural network-based methods for solving nonlinear inverse problems. Our work focuses on highly nonlinear inverse problems where the parameter space has a much smaller dimension than the observation space. In particular, this paper describes the SimPANNs approach that has shown very promising results. SimPANNs leverages the physics forward model to (approximately) simulate data that are using for training data while also taking into account that forward model with a hybrid loss. As such,

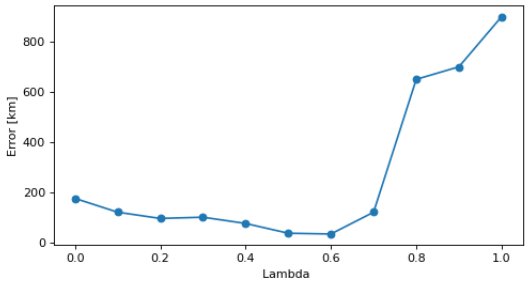

Figure 11: Evaluation of the position error (over 24h) of the reconstructed orbits evaluated for multiples values of lambda used for the training in a context of $n_s = n_0 = 40000$.

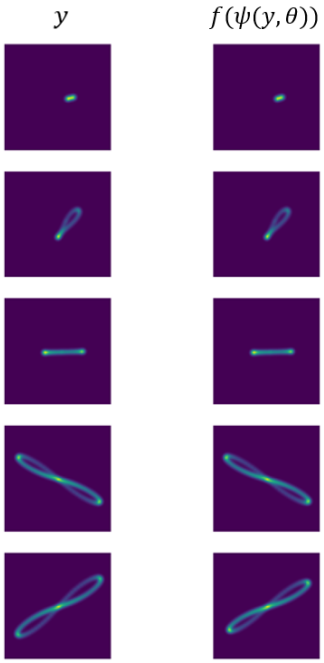

Figure 12: Reconstruction orbit images using the SimPANNs method.

SimPANNs takes the best of both worlds: observation-only and simulated data. The study demonstrates that simulation-aided training provides more information compared to conventional PANNs or supervised vanilla NN training. The forward model needs only to be known to a reasonable approximation without hampering the SimPANNs method performance. SimPANNs shows potential for addressing challenging inverse problems with limited data, underscoring the advantages brought physics-based knowledge when training neural networks.

References

[1] M. Bertero and M. Piana. "Inverse problems in biomedical imaging: modeling and methods of solution". In: *Complex Systems in Biomedicine*. Ed. by Alfio Quarteroni, Luca Formaggia, and Alessandro Veneziani. Milano: Springer Milan, 2006, pp. 1–33. ISBN: 978-88-470-0396-5. DOI: 10.1007/88-470-0396-2_1. URL: https://doi.org/10.1007/88-470-0396-2_1.

[2] Francesco Picetti. "How Deep Learning Can Help Solving Geophysical Inverse Problems". In: *Special Topics in Information Technology*. Ed. by Carlo G. Riva. Cham: Springer International Publishing, 2023, pp. 141–152. DOI: 10.1007/978-3-031-15374-7_12. URL: https://doi.org/10.1007/978-3-031-15374-7_12.

[3] J.-L. Starck and F. Murtagh. *Astronomical Image and Data Analysis*. 2nd edn. Springer, 2006.

[4] Singanallur V. Venkatakrishnan, Charles A. Bouman, and Brendt Wohlberg. "Plug-and-Play priors for model based reconstruction". In: *2013 IEEE Global Conference on Signal and Information Processing*. 2013, pp. 945–948. DOI: 10.1109/GlobalSIP.2013.6737048.

[5] Kyong Hwan Jin et al. "Deep Convolutional Neural Network for Inverse Problems in Imaging". In: *IEEE Trans. Image Process.* 26.9 (2017), pp. 4509–4522. DOI: 10.1109/TIP.2017.2713099. URL: https://doi.org/10.1109/TIP.2017.2713099.

[6] Pei Peng, Shirin Jalali, and Xin Yuan. "Solving Inverse Problems via Auto-Encoders". In: *IEEE J. Sel. Areas Inf. Theory* 1.1 (2020), pp. 312–323. DOI: 10.1109/jsait.2020.2983643. URL: https://doi.org/10.1109/jsait.2020.2983643.

[7] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. "Deep Image Prior". In: *International Journal of Computer Vision* 128.7 (Mar. 2020), pp. 1867–1888. DOI: 10.1007/s11263-020-01303-4. URL: https://doi.org/10.1007%2Fs11263-020-01303-4.

[8] Zihui Wu et al. "SIMBA: Scalable Inversion in Optical Tomography Using Deep Denoising Priors". In: *IEEE Journal of Selected Topics in Signal Processing* 14.6 (Oct. 2020), pp. 1163–1175. DOI: 10.1109/jstsp.2020.2999820. URL: https://doi.org/10.1109%2Fjstsp.2020.2999820.

[9] Christopher A. Metzler and Gordon Wetzstein. "Deep S3PR: Simultaneous Source Separation and Phase Retrieval Using Deep Generative Models". In: *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2021, pp. 1370–1374. DOI: 10.1109/ICASSP39728.2021.9413714.

[10] Christopher Metzler et al. "prDeep: Robust Phase Retrieval with a Flexible Deep Network". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 3501–3510. URL: https://proceedings.mlr.press/v80/metzler18a.html.

[11] Yair Rivenson et al. "Phase recovery and holographic image reconstruction using deep learning in neural networks". In: *Light: Science and Applications* 7.2 (Oct. 2017), pp. 17141–17141. DOI: 10.1038/lsa.2017.141. URL: https://doi.org/10.1038%2Flsa.2017.141.

[12] Jan Blechschmidt and Oliver G. Ernst. *Three Ways to Solve Partial Differential Equations with Neural Networks – A Review*. 2021. arXiv: 2102.11802 [math.NA].

[13] Gregory Ongie et al. *Deep Learning Techniques for Inverse Problems in Imaging*. 2020. arXiv: 2005.06001 [eess.IV].

[14] Simon Arridge et al. "Solving inverse problems using data-driven models". In: *Acta Numerica* 28 (2019), pp. 1–174. DOI: 10.1017/S0962492919000059.

[15] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational Physics* 378 (2019), pp. 686–707. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2018.10.045. URL: https://www.sciencedirect.com/science/article/pii/S0021999118307125.

[16] Maziar Raissi. "Deep hidden physics models: Deep learning of nonlinear partial differential equations". In: *The Journal of Machine Learning Research* 19.1 (2018), pp. 932–955.

[17] Ziya Uddin et al. "Wavelets based physics informed neural networks to solve non-linear differential equations". In: *Scientific Reports* 13.1 (Feb. 2023), p. 2882. ISSN: 2045-2322. DOI: 10.1038/s41598-023-29806-3. URL: https://doi.org/10.1038/s41598-023-29806-3.

[18] Zhongkai Hao et al. *Physics-Informed Machine Learning: A Survey on Problems, Methods and Applications*. arXiv:2211.08064. 2022. DOI: 10.48550/arXiv.2211.08064. URL: https://doi.org/10.48550/arXiv.2211.08064.

[19] Lu Lu et al. "Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators". In: *Nature Machine Intelligence* 3.3 (2021), pp. 218–229. DOI: 10.1038/s42256-021-00302-5. URL: https://doi.org/10.1038/s42256-021-00302-5.

[20] Lu Lu et al. "Multifidelity deep neural operators for efficient learning of partial differential equations with application to fast inverse design of nanoscale heat transport". In: *Phys. Rev. Res.* 4 (2 June 2022), p. 023210. DOI: 10.1103/PhysRevResearch.4.023210. URL: https://link.aps.org/doi/10.1103/PhysRevResearch.4.023210.

[21] Zongyi Li et al. "Fourier Neural Operator for Parametric Partial Differential Equations". In: *The International Conference on Learning Representations (ICLR)*. 2021.

[22] Raphaël Pestourie et al. "Physics-enhanced deep surrogates for partial differential equations". In: *Nature Machine Intelligence* 5.12 (2023), pp. 1458–1465. DOI: 10.1038/s42256-023-00761-y. URL: https://doi.org/10.1038/s42256-023-00761-y.

[23] O. Scherzer et al. *Variational methods in imaging*. Vol. 167. Springer, 2009.

[24] Hemant K. Aggarwal, Merry P. Mani, and Mathews Jacob. "MoDL: Model-Based Deep Learning Architecture for Inverse Problems". In: *IEEE Transactions on Medical Imaging* 38.2 (2019), pp. 394–405. DOI: 10.1109/TMI.2018.2865356.

[25] Yaniv Romano, Michael Elad, and Peyman Milanfar. "The Little Engine That Could: Regularization by Denoising (RED)". In: *SIAM Journal on Imaging Sciences* 10.4 (2017), pp. 1804–1844. DOI: 10.1137/16M1102884. eprint: https://doi.org/10.1137/16M1102884. URL: https://doi.org/10.1137/16M1102884.

[26] Nir Shlezinger and Yonina C. Eldar. "Model-Based Deep Learning". In: *Foundations and Trends in Signal Processing* 17.4 (2023), pp. 291–416. ISSN: 1932-8346. DOI: 10.1561/2000000113. URL: http://dx.doi.org/10.1561/2000000113.

[27] Chen Wei et al. "Edge Sparse Basis Network: A Deep Learning Framework for EEG Source Localization". In: *2021 International Joint Conference on Neural Networks (IJCNN)*. 2021, pp. 1–8. DOI: 10.1109/IJCNN52387.2021.9533968.

[28] Ali Mousavi and Richard G. Baraniuk. "Learning to invert: Signal recovery via Deep Convolutional Networks". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017, pp. 2272–2276. DOI: 10.1109/ICASSP.2017.7952561.

[29] Deniz Engin, Anil Genc, and Hazim Kemal Ekenel. "Cycle-Dehaze: Enhanced CycleGAN for Single Image Dehazing". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2018, pp. 938–9388. DOI: 10.1109/CVPRW.2018.00127.

[30] Ashish Bora et al. "Compressed Sensing using Generative Models". In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 537–546. URL: http://proceedings.mlr.press/v70/bora17a.html.

[31] Ameya D Jagtap et al. "Physics-informed neural networks for inverse problems in supersonic flows". In: *Journal of Computational Physics* 466 (2022), p. 111402.

[32] Yuyao Chen et al. "Physics-informed neural networks for inverse problems in nano-optics and metamaterials". In: *Optics express* 28.8 (2020), pp. 11618–11633.

[33] Sidney Besnard, Frederic Jurie, and Jalal Fadili. "Simpinns: Simulation-Driven Physics-Informed Neural Networks for Enhanced Performance in Nonlinear Inverse Problems". In: *2023 13th Workshop on Hyperspectral Imaging and Signal Processing: Evolution in Remote Sensing (WHISPERS)*. 2023, pp. 1–5. DOI: 10.1109/WHISPERS61460.2023.10430751.

[34] Dario Izzo and Sebastien Origer. "Neural representation of a time optimal, constant acceleration rendezvous". In: *Acta Astronautica* 204 (2023), pp. 510–517. ISSN: 0094-5765. DOI: https://doi.org/10.1016/j.actaastro.2022.08.045. URL: https://www.sciencedirect.com/science/article/pii/S0094576522004581.

[35] Xingyu Zhou et al. "A LSTM assisted orbit determination algorithm for spacecraft executing continuous maneuver". In: *Acta Astronautica* 204 (2023), pp. 568–582. ISSN: 0094-5765. DOI: https://doi.org/10.1016/j.actaastro.2022.09.041. URL: https://www.sciencedirect.com/science/article/pii/S0094576522005069.

[36] Hao Peng and Xiaoli Bai. "Fusion of a machine learning approach and classical orbit predictions". In: *Acta Astronautica* 184 (2021), pp. 222–240. ISSN: 0094-5765. DOI: https://doi.org/10.1016/j.actaastro.2021.04.017. URL: https://www.sciencedirect.com/science/article/pii/S0094576521001636.

[37] Okchul Jung et al. "Recurrent neural network model to predict reentry trajectories of uncontrolled space objects". In: *Advances in Space Research* 68.6 (2021), pp. 2515–2529. ISSN: 0273-1177. DOI: https://doi.org/10.1016/j.asr.2021.04.041. URL: https://www.sciencedirect.com/science/article/pii/S0273117721003409.

[38] Andrea Scorsoglio, Luca Ghilardi, and Roberto Furfaro. "A Physic-Informed Neural Network Approach to Orbit Determination". In: *The Journal of the Astronautical Sciences* 70.4 (Aug. 2023), p. 25. ISSN: 2195-0571. DOI: 10.1007/s40295-023-00392-w. URL: https://doi.org/10.1007/s40295-023-00392-w.

[39] Yang Zheng Bin et al. "Passive satellite localization using TDOA/FDOA/AOA measurements". In: *IEEE Conference Anthology*. 2013, pp. 1–5. DOI: 10.1109/ANTHOLOGY.2013.6784815.

[40] P.R. Escobal. *Methods of Orbit Determination*. J. Wiley, 1965. ISBN: 9780471245346. URL: https://books.google.fr/books?id=J4RTAAAAMAAJ.

[41] O. Montenbruck and E. Gill. *Satellite Orbits: Models, Methods, and Applications*. Physics and astronomy online library. Springer Berlin Heidelberg, 2000. ISBN: 9783540672807. URL: https://books.google.fr/books?id=hABRnDlDkyQC.

[42] Dongdong Chen, Julian Tachella, and Mike Davies. "Equivariant imaging: learning beyond the range space". In: *ICCV*. IEEE/CVF, 2021, pp. 4379–4388.

[43] Karol Gregor and Yann LeCun. "Learning Fast Approximations of Sparse Coding". In: *ICML*. Omnipress, 2010, pp. 399–406.

[44] *Correcting Image Orientation Using Convolutional Neural Networks*. https://d4nst.github.io/2017/01/12/image-orientation/. Accessed: 2023-03-15.

[45] Li Deng. "The mnist database of handwritten digit images for machine learning research". In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[46] A.H. de Ruiter, C. Damaren, and J.R. Forbes. *Spacecraft Dynamics and Control: An Introduction*. Wiley, 2012. ISBN: 9781118403327. URL: https://books.google.fr/books?id=mGSYHJIlDxEC.

[47] Felix R. Hoots and Ronald L. Roehrich. "Models for Propagation of NORAD Element Sets." In: *Models for Propagation of NORAD Element Sets*. Ed. by U.S. Air Force: Aerospace Defense Command. 1080. DOI: 10.1007/978-3-031-15374-7_12. URL: https://apps.dtic.mil/sti/citations/ADA093554.

[48] W.D. McClain and D.A. Vallado. *Fundamentals of Astrodynamics and Applications*. Space Technology Library. Springer Netherlands, 2001. ISBN: 9780792369035. URL: https://books.google.fr/books?id=PJLlWzMBKjkC.

[49] II George W. Collins. *The Foundations of Celestial Mechanics*. Case Western Reserve University, 2004. URL: https://ads.harvard.edu/books/1989fcm..book/.

[50] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. 2nd ed. New York, NY, USA: Cambridge University Press, 2003. ISBN: 0521540518.

[51] Jérôme Bolte and Edouard Pauwels. "Conservative set valued fields, automatic differentiation, stochastic gradient methods and deep learning". In: *Math. Program.* 188.1 (2021), pp. 19–51. DOI: 10.1007/s10107-020-01501-5. URL: https://doi.org/10.1007/s10107-020-01501-5.

[52] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6980.