MSc in Mathematical Engineering

Course: Numerical Analysis for Partial Differential Equations

Lecturer: Prof Paola F. Antonietti
Teaching Assistant: Dr. F. Regazzoni
Tutor: Dr. C.B. Leimer Saglio

Type B project

A.Y. 2024/2025

Accelerated early warning systems for natural catastrophes by deep learning-driven simulations

---

Authors:

Andrea Rella,
Francesca Zambetti

Tutors:

Prof. Filippo Gatti,
Prof. Stefania Fresca

Abstract

This work presents a data-driven pipeline for generating synthetic seismic acceleration signals using a v-prediction denoising diffusion probabilistic model (v-prediction DDPM). Traditional physics-based simulations based on the elastodynamic equation often suffer from inaccuracies due to over-simplistic representations of the Earth and complex boundary conditions, making them difficult to incorporate into deep neural networks.
To overcome these limitations, we adopt a U-Net-based diffusion model trained as a denoiser, and introduce a novel regularization term in the loss function that captures the interfrequency correlation structure of seismic signals. This term is derived from empirical models describing the covariance of the logarithmic Fourier spectral components and is designed to improve spectral realism without explicitly imposing physical constraints.
We evaluate the model under three training configurations: with low-frequency conditioning, with the new loss term, and with both. Quantitative metrics such as Signal-to-Noise Ratio (SNR) and Structural Similarity Index (SSIM) confirm this trade-off. The best performance in terms of structural similarity is generally achieved when both the conditioning and the correlation-based loss term are combined, although at the cost of slightly lower consistency. In particular, the additional loss enhances spectral coherence, while the conditioning ensures that the generated signals remain physically plausible in the time domain.

Keywords: diffusion models, synthetic accelerograms, interfrequency correlation

Source: `https://github.com/francesca1606/Our_Diffusion_NAPDE/tree/main`

Contents

1 Introduction

Earthquakes are among the most disruptive natural phenomena, and much of the damage they cause stems from the high uncertainty associated with their prediction. It is therefore crucial to analyze all available data and information to improve forecasting capabilities and mitigate potential consequences. However, obtaining data that is both region-specific and tailored to the characteristics of a given earthquake is often challenging.
For this reason, in the field of seismic research, it becomes necessary to generate synthetic data derived from the available measurements. The generation of artificial earthquake data is essential for improving preparedness, enhancing safety measures, and advancing scientific understanding in the field of seismology.

A significant aspect of seismology is the development and application of methods to compute synthetic seismograms corresponding to realistic Earth models. Broadly speaking, the objective is to determine the signal that would be recorded by a seismograph at a designated receiver, assuming a precise representation of both the seismic source and the Earth's structure through which the waves travel. This constitutes a well-posed forward problem which, theoretically, admits an exact solution. However, in practical scenarios, synthetic seismograms are often subject to inaccuracies.
These errors can be classified into two main categories. The first category includes errors stemming from theoretical approximations employed during the computation of synthetic seismograms. For instance, ray theory [23] inadequately represents head waves, diffracted waves or the interaction between wave types at long periods. Additionally, numerical artifacts such as grid dispersion are common in finite difference approaches, due to spatial and time discretization. The second type of error corresponds to inaccuracies due to oversimplified Earth as a medium or source models. Even if the synthetics are computed exactly for such models, they may not faithfully represent the complexities of the actual Earth. These simplifications may be driven by the requirements of the numerical method or by limited knowledge of the true structure. Typical examples include adopting 1D models that neglect 3D heterogeneities, treating the source as a point rather than a finite rupture, or ignoring attenuation and anisotropy effects during the calculations.
The first type of error can often be mitigated by employing more accurate computational algorithms. However, in real-world applications, constraints on computational resources may limit the achievable precision, particularly for complex Earth models. The second type of error poses a greater challenge, as it typically arises from insufficient knowledge of the Earth's structure, making it impossible to replicate every detail observed in recorded seismograms.

To address this, researchers have turned to deep learning algorithms to generate new and unseen data, starting from available ones. Among the various tools available for data generation, particular attention has been devoted to diffusion models, which have demonstrated outstanding performance compared to other approaches. These models are trained by progressively adding noise to a given sample until the original information is effectively lost. They then learn to reconstruct the sample by gradually removing the noise. This learned reverse diffusion process is subsequently used to generate realistic samples from random noise, effectively "denoising" the input to produce novel data [8].
The range of applications for these models is extensive: they are employed in domains such as image generation [9], video synthesis [10] and text generation [13]. Another interesting field of application is shown in [29]: a GAN-based model is employed to reconstruct 3D velocity fields from unpaired 2D velocity measurements in turbulent flows, significantly reducing experimental complexity and storage costs. It is also possible to incorporate physical information by conditioning the model, as demonstrated by [24] in the context of fluid dynamics. Furthermore, the work

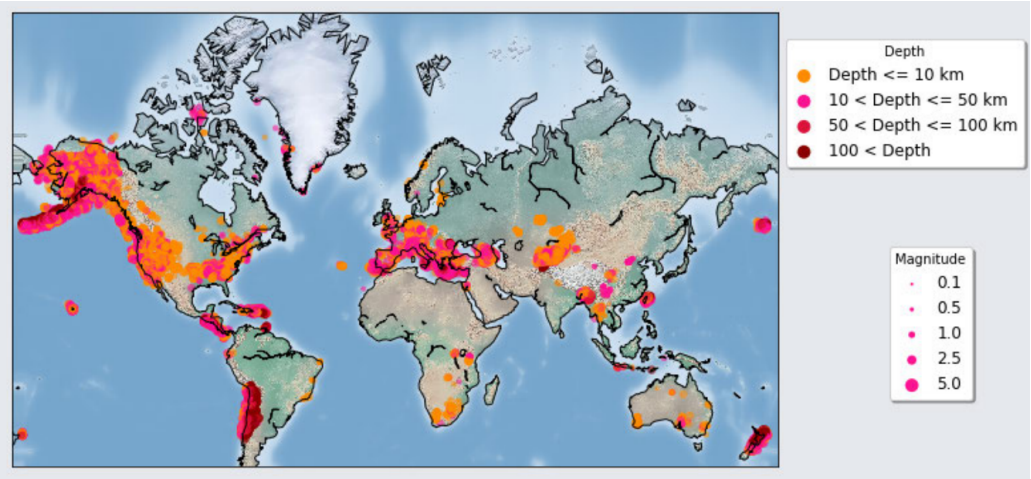


Fig. 1. Location, size and depth distribution of recorded earthquakes in STEAD [15]

in [2] shows that the generative capabilities of the model are enhanced when such information is included during training.

Generative models have been employed in [4, 27] to produce seismic acceleration signals that closely match the target distribution, under various types of constraints. The most important physical constraint is that the generated data should satisfy the elastodynamic equation along with the appropriate boundary conditions:

$$\rho\ddot{\mathbf{u}} - \nabla \cdot \sigma(\mathbf{u}) + \mathscr{G}(\mathbf{u}) = \mathbf{f} \quad \text{in } \Omega \times (0, T],$$

However, these works highlight the challenges of embedding such physical constraints into the architecture of generative models, particularly in seismic applications.
To address such limitations associated with incorporating physical constraints into generative models for seismic signal simulation, this work adopts a data-driven approach aimed at learning the underlying spatial and frequency-dependent structure of seismic acceleration signals. Rather than requiring explicit physical modeling of boundary conditions or physical parameters, the proposed framework focuses on reproducing the statistical properties observed in real ground motions.

In this context, the interfrequency correlation and variance models developed in [26] are used to construct a regularization term integrated into the loss function of the neural network. These empirical models describe the covariance structure of Fourier spectral ordinates across frequencies and are leveraged here to guide the network towards generating synthetic accelerograms that exhibit realistic interfrequency dependencies. This additional constraint ensures that the generated signals are not only consistent with the observed time-domain waveforms but also with the known statistical structure of their spectral representation.
In addition, a low-frequency condition is used to guide the generation process during training. The idea, at inference time, is to provide a cheap yet realistic low-frequency signal obtained from physic-based simulation (pbs) $\mathbf{x} \sim p_{\text{pbs}}$ to guide the generation and thus preserve and incorporate the minimum knowledge on the earthquake physics.

2 Theoretical Overview and Method

This section, after a brief introduction to the seismic wave equation along with its criticalities and outlines the structure of the diffusion model used to generate synthetic accelerograms. It also provides a detailed description of the empirical model that captures the interfrequency correlations among Fourier spectral ordinates of seismic signals [26] and the low-frequency conditioning.

2.1 The Seismic Wave Equation

We start our discussion with the one-dimensional wave equation

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2} \tag{1}$$

and its general solution

$$u(x,t) = f(x \pm ct)$$

which describes waves of arbitrary form propagating at speed c in both the positive and negative directions along the x-axis. This equation is widely encountered in physics and serves to model phenomena such as string vibrations or sound waves in ducts. The propagation speed c is governed by the physical properties of the medium through which the wave travels.
In the classification of linear partial differential equations, the wave equation is considered hyperbolic. Hyperbolic equations are notably difficult to solve due to the persistence of sharp features in their solutions, which may also reflect from boundaries. Unlike the diffusion equation, for instance, where solutions tend to smooth out over time, the wave equation yields smooth results only when the initial conditions are smooth. This feature poses significant challenges for both analytical and numerical solution techniques. As we will observe, the seismic wave equation introduces additional complexity beyond equation (1), as it operates in three dimensions and requires a full accounting of the stress-strain relationship in anelastic solids to relate force and displacement. Nonetheless, P and S seismic waves exhibit several similarities with the 1D wave equation solutions [23]. They consist of wave pulses of arbitrary shape that propagate at velocities dictated by the medium's elastic parameters and density, and are frequently analyzed through decomposition into harmonic components expressed in terms of sine and cosine functions.

Because seismic waves are inherently time-dependent and involve both velocities and accelerations, it is essential to incorporate the effects of momentum into our analysis. This is achieved by applying Newton's second law, $\mathbf{F} = m\mathbf{a}$, to a continuous medium.
To illustrate this, we examine the forces acting on an infinitesimal volume element (a cube) defined within a (x_1, x_2, x_3) coordinate system. The force acting on each face of the cube is determined by the traction vector multiplied by the corresponding surface area. For instance, the force acting on the face perpendicular to the x_1-axis is given by [23]:

$$\begin{aligned}\mathbf{F}(\hat{\mathbf{x}}_1) &= \mathbf{t}(\hat{\mathbf{x}}_1)\, dx_2\, dx_3 \\ &= \boldsymbol{\tau}\hat{\mathbf{x}}_1\, dx_2\, dx_3 \\ &= \begin{bmatrix}\tau_{11}\\ \tau_{21}\\ \tau_{31}\end{bmatrix} dx_2\, dx_3\end{aligned}$$

where $\hat{\mathbf{x}}_1 = (1,0,0)$, $\mathbf{F}$ is the force vector, $\mathbf{t}$ is the traction vector and $\boldsymbol{\tau}$ is the stress tensor. Net force will only be exerted on the cube if spatial gradients are present in the stress field. In this case,the net force from the planes normal to x_1 is

$$\mathbf{F}(\hat{\mathbf{x}}_1) = \frac{\partial}{\partial x_1}\begin{bmatrix}\tau_{11}\\ \tau_{21}\\ \tau_{31}\end{bmatrix} dx_1\, dx_2\, dx_3$$

and we can use index notation and Einstein notation to express the total force from the stress field on all the faces of the cube as

$$F_i = \partial_j \tau_{ij} \, dx_1 \, dx_2 \, dx_3 \tag{2}$$

where $\partial_j \tau_{ij}$ is the i-th component of the divergence of the stress tensor $\boldsymbol{\tau}$. There may also exist a body force on the cube proportional to the volume of material, that is,

$$F_i^b = f_i \, dx_1 \, dx_2 \, dx_3 \tag{3}$$

The mass of our infinitesimal cube is given by

$$m = \rho \, dx_1 \, dx_2 \, dx_3 \tag{4}$$

where ρ is the density. The acceleration of the cube is given by the second time derivative of the displacement $\mathbf{u}$. Substituting (2)–(4) into $\mathbf{F} = m\mathbf{a}$ we obtain[1]

$$\rho \frac{\partial^2 u_i}{\partial t^2} = \partial_j \tau_{ij} + f_i \tag{5}$$

This equation forms the foundation of much of seismological theory and is commonly referred to as the momentum equation or the equation of motion for a continuous medium. Each variable, u_i, τ_{ij}, and f_i, depends on both spatial position $\mathbf{x}$ and time. The body force term $\mathbf{f}$ typically comprises two components: a gravitational force f_g and a source term f_s. While gravity plays a significant role at very low frequencies, particularly in normal mode seismology, it is usually negligible in the analysis of body and surface waves at commonly observed seismic wavelengths. When body forces are absent, the equation of motion simplifies to its homogeneous form:

$$\rho \frac{\partial^2 u_i}{\partial t^2} = \partial_j \tau_{ij} \tag{6}$$

which governs seismic wave propagation in regions located away from the seismic source. Deriving solutions to equations such as (5) or (6) within the context of realistic Earth models is a central task in seismology. These solutions yield the expected ground motion at various locations relative to the source and are commonly referred to as synthetic seismograms.

To obtain solutions to (5) or (6), it is necessary to establish a constitutive relationship between stress and strain, allowing the stress tensor $\boldsymbol{\tau}$ to be expressed in terms of the displacement field $\mathbf{u}$. We recall here the linear, isotropic stress–strain relationship [23]:

$$\tau_{ij} = \lambda \delta_{ij} e_{kk} + 2\mu e_{ij} \tag{7}$$

where λ and μ are the Lamé parameters, which describe the mechanical behaviour of an isotropic elastic medium and characterize how the material responds to stress and strain, and the strain tensor is defined as $e_{ij} = \frac{1}{2}(\partial_i u_j + \partial_j u_i)$. Substituting for e_{ij} in (7) we obtain

$$\tau_{ij} = \lambda \delta_{ij} \partial_k u_k + \mu(\partial_i u_j + \partial_j u_i) \tag{8}$$

Equation (5) or (6) and (8) provide a coupled set of equations for the displacement and stress and can be compacted into

$$\rho \ddot{\mathbf{u}} - \nabla \cdot \sigma(\mathbf{u}) = \mathbf{f}$$

[1] When formulating the acceleration term, we approximate the total time derivatives of $\mathbf{u}$ by their partial time derivatives. This simplification corresponds to the small-deformation assumption, where the contributions involving spatial derivatives of $\mathbf{u}$ in the total derivative are neglected. While this approximation is typically acceptable in seismological contexts, it is worth noting that the spatial derivative (advection) terms play a significant role in fluid mechanics and cannot be disregarded in that field [23].

with

$$\sigma(\mathbf{u}) = 2\lambda\mathbf{e}(\mathbf{u}) + \mu\ \mathrm{Tr}(\mathbf{e}(\mathbf{u}))\mathbf{I}$$

Moreover, in real materials, especially viscoelastic ones (like polymers, biological tissues, or geological media), mechanical waves are attenuated. This means part of the wave energy is lost due to internal friction, molecular rearrangements, or other irreversible processes. In principle we would need to account for this and consider an additional rheological damping term [1]:

$$\rho\ddot{\mathbf{u}} - \nabla \cdot \sigma(\mathbf{u}) + \mathscr{G}(\mathbf{u}) = \mathbf{f} \tag{9}$$

with a possible modelisation based on the proportional damping, where damping is modeled as a combination of velocity-proportional (viscous damping) and displacement-proportional (stiffness-like damping) terms

$$\mathscr{G}(\mathbf{u}) = \begin{cases} 0 & \text{fully elastic} \\ 2\rho\zeta\dot{\mathbf{u}} + \rho\zeta^2\mathbf{u} & \text{viscoelastic} \end{cases}$$

with ζ damping coefficients $[s^{-1}]$.

As mentioned above, numerically solving (9) together with the appropriate boundary conditions presents several critical challenges:

- boundary definition: choosing a computational domain means encapsulating the phenomenon into a fixed limited region while earthquakes span great lengths;
- boundary conditions: accurately capturing the effects of geological boundaries or discontinuities (for example faults or layer interfaces) is essential for realistic seismic modeling, but remains difficult to incorporate in data-driven frameworks;
- heterogeneous soil composition: the variability in soil types, stratification, and layering significantly affects seismic wave propagation. However, modeling such spatially varying properties requires detailed input data, which is often unavailable or difficult to integrate;
- variability in elastic parameters: differences in Lamé parameters, density, and other mechanical properties introduce complex and spatially non-uniform dynamics in seismic responses. Representing such heterogeneity in a neural network model is particularly challenging.
- computational cost: higher frequencies (lower wavelengths) necessitate finer and finer grids, leading to more expensive simulations.

For these reasons, despite representing the most critical physical constraint that the generated data should fulfill, the direct use of the governing equation as a conditioning input remains limited. As a consequence, people have turned toward alternative physics-informed generative approaches for generating good quality data with accessible computational costs (considering both offline training of the algorithm and online inference cost).

2.2 v-prediction DDPM

A Denoising Diffusion Probabilistic Model (DDPM) [9] will serve as the foundation of our generative framework. DDPMs belong to a family of generative models that operate by progressively corrupting a data sample (such as an image, text or signal) with noise and then training a model to reverse this process, thereby enabling the generation of new samples. This framework comprises two main components:

1. a fixed forward diffusion process q, chosen a priori, which incrementally perturbs a data sample by adding Gaussian noise over multiple steps, ultimately resulting in a completely noisy signal;

2. a parameterized reverse process p_θ, where a neural network is trained to iteratively denoise the corrupted signal, starting from pure noise and recovering a coherent data sample.

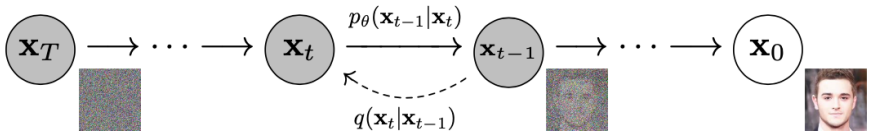


Fig. 2. Directed graph of the diffusion process [9]

So, let $q(\mathbf{y}_0)$ represent the distribution of real data (e.g., real images); we can draw samples from this distribution, denoted as $\mathbf{y}_0 \sim q(\mathbf{y}_0)$. The forward process $q(\mathbf{y}_t|\mathbf{y}_{t-1})$ is defined to inject Gaussian noise at each time step t, following a predefined variance schedule $0 < \beta_1 < \beta_2 < \cdots < \beta_T < 1$, as

$$q(\mathbf{y}_t|\mathbf{y}_{t-1}) = \mathcal{N}(\mathbf{y}_t; \sqrt{1-\beta_t}\mathbf{y}_{t-1}, \beta_t\mathbf{I})$$

Each new, slightly noisier image at time step t is drawn from a conditional Gaussian distribution with $\boldsymbol{\mu}_t = \sqrt{1-\beta_t}\mathbf{y}_{t-1}$ and $\sigma_t^2 = \beta_t$, obtained by sampling $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and then setting

$$\mathbf{y}_t = \sqrt{1-\beta_t}\mathbf{y}_{t-1} + \sqrt{\beta_t}\boldsymbol{\epsilon}$$

So starting from $\mathbf{y}_0$, we end up with $\mathbf{y}_1, ..., \mathbf{y}_t, ..., \mathbf{y}_T$, where $\mathbf{y}_T$ will be pure Gaussian noise, provided we set the schedule appropriately. A direct consequence of the constructed forward process q is that we can sample $\mathbf{y}_t$ at any arbitrary noise level conditioned on $\mathbf{y}_0$

$$q(\mathbf{y}_t|\mathbf{y}_0) = \mathcal{N}(\mathbf{y}_t; \sqrt{\bar{\alpha}_t}\mathbf{y}_0, (1-\bar{\alpha}_t)\mathbf{I}) \tag{10}$$

with $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$ [9]. This is very convenient since it is not necessary to apply q repeatedly in order to sample $\mathbf{y}_t$. Note that the $\bar{\alpha}_t$ are functions of the known β_t variance schedule and thus are also known and can be precomputed. This then allows, during training, to optimize random terms of the loss function $\mathscr{L}$ (or in other words, to randomly sample t during training and optimize $\mathscr{L}_t$).

To facilitate a reverse process from $\mathbf{y}_T$ to $\mathbf{y}_0$, the conditional distribution $p(\mathbf{y}_{t-1}|\mathbf{y}_t)$ is pivotal. However, computing this directly across all possible images is intractable. Thus, we employ a neural network parameterized by θ to approximate this distribution, denoted as $p_\theta(\mathbf{y}_{t-1}|\mathbf{y}_t)$, optimized via gradient descent.
Assuming a Gaussian reverse process, each distribution is characterized by mean $\boldsymbol{\mu}_\theta(\mathbf{y}_t, t)$ and variance $\boldsymbol{\Sigma}_\theta(\mathbf{y}_t, t)$. For simplicity, we set $\boldsymbol{\Sigma}_\theta(\mathbf{y}_t, t) = \sigma_t^2\mathbf{I}$, treating σ_t^2 as time-dependent constants. Following [9] we set $\sigma_t^2 = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$ so that

$$p_\theta(\mathbf{y}_{t-1}|\mathbf{y}_t) \sim \mathcal{N}\left(\boldsymbol{\mu}_\theta(\mathbf{y}_t, t), \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t\mathbf{I}\right) \tag{11}$$

The neural network then learns solely the mean $\boldsymbol{\mu}_\theta(\mathbf{y}_t, t)$ of the conditional distribution. Notably, the loss function simplifies by predicting the added noise instead of directly estimating the mean. Specifically, the mean can be computed as:

$$\boldsymbol{\mu}_\theta(\mathbf{y}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{y}_t - \beta_t\frac{1}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{y}_t, t)\right) \tag{12}$$

where $\boldsymbol{\epsilon}_\theta(\mathbf{y}_t, t)$ represents predicted noise. The objective function for a random time step t, with noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, becomes:

$$\mathbb{E}_{\mathbf{y}_0,t,\boldsymbol{\epsilon}}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{y}_0 + \sqrt{(1-\bar{\alpha}_t)}\boldsymbol{\epsilon}, t)\|_2^2\right]$$

where $\mathbf{y}_0$ denotes the original image. The neural network minimizes the MSE between true and predicted noise during training. Then, the training and sampling algorithms look as follows:

Algorithm 1 Training

repeat
  $\mathbf{y}_0 \sim q(\mathbf{y}_0)$
  $t \sim \text{Uniform}(\{1, \ldots, T\})$
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  Take gradient descent step on
  $\nabla_\theta \left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{y}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t\right)\right\|^2$
until converged

Algorithm 2 Sampling

$\mathbf{y}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T$ to 1 **do**
  if $t > 1$ **then**
    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  else
    $\mathbf{z} = \mathbf{0}$
  end if
  $\mathbf{y}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{y}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta\right) + \sigma_t\mathbf{z}$
end for
return $\mathbf{y}_0$

So, in standard DDPMs (like [9]), the model is trained to predict the noise $\boldsymbol{\epsilon}$ added at a certain timestep t. You take a clean image $\mathbf{y}_0$, add noise to it using the forward process, and the neural network learns to denoise by minimizing the MSE between the true noise and the predicted noise. This formulation is simple, but it's not necessarily the most optimal target from a learning or stability perspective. That's why what we will implement in the following it's variant called "**v**-prediction DDPM".

In v-prediction DDPMs (also called velocity prediction or hybrid prediction, introduced in [20, 21]), instead of predicting just noise or just the clean image, the model predicts a new quantity:

$$\mathbf{v}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{\epsilon} - \sqrt{1-\bar{\alpha}_t}\mathbf{y}_0$$

It is a linear combination of the clean image and the noise, and can be interpreted as lying along the line connecting the pure noise tensor $\boldsymbol{\epsilon}$ and the clean image $\mathbf{y}_0$. The tensor $\mathbf{v}_t$ essentially "points" in the direction from $\mathbf{y}_0 \to \boldsymbol{\epsilon}$, capturing not just a position but rather a direction of change, analogous to a derivative or velocity vector.
Predicting this velocity turns out to be easier to learn for the model and provides better gradients during training [20]. Moreover, given the noisy input $\mathbf{y}_t$, using (10) it's possible to compute both:

$$\mathbf{y}_0 = \sqrt{\bar{\alpha}_t}\mathbf{y}_t - \sqrt{1-\bar{\alpha}_t}\mathbf{v}_t$$

$$\boldsymbol{\epsilon} = \sqrt{1-\bar{\alpha}_t}\mathbf{y}_t + \sqrt{\bar{\alpha}_t}\mathbf{v}_t$$

so no information is lost: predicting $\mathbf{v}_t$ still allows us to recover everything we need for training and sampling. The mean of the stochastic backward process (12) can be reconstructed as

$$\boldsymbol{\mu}_\theta(\mathbf{y}_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{y}_t - \beta_t \frac{1}{\sqrt{1-\bar{\alpha}_t}} \left(\sqrt{1-\bar{\alpha}_t}\mathbf{y}_t + \sqrt{\bar{\alpha}_t}\mathbf{v}_\theta(\mathbf{y}_t, t) \right) \right) \tag{13}$$

In this new formulation the training objective becomes minimizing:

$$\mathscr{L}_\mathbf{v} = \mathbb{E}_{\mathbf{y}_0, t, \boldsymbol{\epsilon}} \left[\|\mathbf{v}_t - \mathbf{v}_\theta(\mathbf{y}_t, t)\|^2 \right]$$

and the training looks as follows:

Algorithm 3 Training (v-prediction DDPM)

repeat
  $\mathbf{y}_0 \sim q(\mathbf{y}_0)$
  $t \sim \text{Uniform}(\{1, \dots, T\})$
  $\boldsymbol{\epsilon} \sim \mathscr{N}(0, \mathbf{I})$
  $\mathbf{y}_t = \sqrt{\bar{\alpha}_t}\mathbf{y}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$
  $\mathbf{v}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{\epsilon} - \sqrt{1-\bar{\alpha}_t}\mathbf{y}_0$
  Take gradient step on
  $\nabla_\theta \|\mathbf{v}_t - \mathbf{v}_\theta(\mathbf{y}_t, t)\|^2$
until converged

Regarding the sampling algorithm, we will deploy the DDIM (Denoising Diffusion Implicit Model) paradigm [25]. DDIM changes the reverse process: instead of a stochastic (probabilistic) denoising process as in DDPM, it defines a deterministic one, in a way that preserves the same training objective $\boldsymbol{\epsilon}_\theta(\mathbf{y}_t, t)$ (or equivalently $\mathbf{v}_\theta(\mathbf{y}_t, t)$) and the same marginals $q(\mathbf{y}_t|\mathbf{y}_0)$. In particular, it sets:

$$\mathbf{y}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\hat{\mathbf{y}}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\eta^2\sigma_t^2} \cdot \boldsymbol{\epsilon}_\theta(\mathbf{y}_t, t) + \eta\sigma_t\mathbf{z} \qquad \text{with } \mathbf{z} \sim \mathscr{N}(\mathbf{0}, \mathbf{I})$$

with $\hat{\mathbf{y}}_0 = \sqrt{\bar{\alpha}_t}\mathbf{y}_t - \sqrt{1-\bar{\alpha}_t}\mathbf{v}_\theta(\mathbf{y}_t, t)$. Observe that η controls how much randomness (noise) is injected at each reverse step during sampling; when $\eta = 0$ the update collapses to $\mathbf{y}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\hat{\mathbf{y}}_0 + \sqrt{1-\bar{\alpha}_{t-1}} \cdot \boldsymbol{\epsilon}_\theta(\mathbf{y}_t, t)$, which is a fully deterministic function of $\mathbf{y}_t$, $\boldsymbol{\epsilon}_\theta(\mathbf{y}_t, t)$ and fixed schedule. Since each denoising step in DDIM deterministically moves toward the final output, it is possible to take larger steps during the inference process, thereby reducing the total number of denoising steps at the cost of some quality degradation. In general, by setting $\eta \in [0, 1]$ we can control the trade-off between diversity and inference speed. The inference algorithm looks as follows

Algorithm 4 DDIM Sampling with v-prediction

Initialize $\mathbf{y}_T \sim \mathscr{N}(\mathbf{0}, \mathbf{I})$
Set desired η
for $t = T$ down to 1 **do**
  if $t > 1$ **then**
    Compute $\hat{\mathbf{y}}_0 = \sqrt{\bar{\alpha}_t}\mathbf{y}_t - \sqrt{1-\bar{\alpha}_t}\mathbf{v}_\theta(\mathbf{y}_t, t)$
    Compute predicted noise: $\boldsymbol{\epsilon}_\theta = \sqrt{1-\bar{\alpha}_t}\mathbf{y}_t + \sqrt{\bar{\alpha}_t}\mathbf{v}_\theta(\mathbf{y}_t, t)$
    Sample noise: $\mathbf{z} \sim \mathscr{N}(\mathbf{0}, \mathbf{I})$
    Update: $\mathbf{y}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\hat{\mathbf{y}}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\eta^2\sigma_t^2} \cdot \boldsymbol{\epsilon}_\theta(\mathbf{y}_t, t) + \eta\sigma_t\mathbf{z}$
  else
    $\mathbf{y}_0 = \sqrt{\bar{\alpha}_1}\mathbf{y}_1 - \sqrt{1-\bar{\alpha}_1}\mathbf{v}_\theta(\mathbf{y}_1, 1)$
  end if
end for
return $\mathbf{y}_0$

2.3 Interfrequency correlations

Accurate modeling of the spectral content of seismic signals requires a joint characterization of amplitudes across different frequencies, accounting for their interfrequency correlation structure. Following the framework proposed by Stafford (2017) [26], accelerograms generated using this approach yield response spectral ordinates with variances that closely align with those predicted by empirical ground-motion models. In this framework, the logarithmic Fourier amplitudes are assumed to follow a multivariate normal distribution, whose covariance matrix is decomposed into three main components:

- between-event correlation $\rho_E(f_i, f_j)$,
- between-site correlation $\rho_S(f_i, f_j)$,
- within-event correlation $\rho_A(f_i, f_j)$.

The total correlation $\rho(f_i, f_j)$ between frequencies f_i and f_j is defined as:

$$\rho(f_i, f_j) = \frac{\rho_E(f_i, f_j)\sigma_E(f_i)\sigma_E(f_j) + \rho_S(f_i, f_j)\sigma_S(f_i)\sigma_S(f_j) + \rho_A(f_i, f_j)\sigma_A(f_i)\sigma_A(f_j)}{\sigma(f_i)\sigma(f_j)} \tag{14}$$

where each σ denotes the standard deviation of the respective component.

The three components are defined as follows:

$$\rho_E(f_i, f_j) = \exp\left(\gamma_E\left(\frac{f_{\min}}{f_c}\right) \cdot \log\left(\frac{f_{\max}}{f_{\min}}\right)\right) \tag{15}$$

$$\rho_S(f_i, f_j) = \rho_{S,0}(f_{\max}, f_{\min}) + (1 - \rho_{S,0}(f_{\min}, f_{\min})) \exp\left(-50 \log\left(\frac{f_{\max}}{f_{\min}}\right)\right) \tag{16}$$

$$\rho_A(f_i, f_j) = \rho_{A,0}(f_{\max}, f_{\min}) + (1 - \rho_{A,0}(f_{\min}, f_{\min})) \exp\left(-50 \log\left(\frac{f_{\max}}{f_{\min}}\right)\right) \tag{17}$$

$$\rho_{X,0}(f_{max}, f_{min}) = (1 - \eta_X(f_{min})) \exp\{\gamma_X(f_{min}) \log(\frac{f_{max}}{f_{min}})\}$$

where $f_{\min} = \min(f_i, f_j)$, $f_{\max} = \max(f_i, f_j)$ and f_c is the corner frequency (see paragraph 3.1). The corner frequency of a signal is the frequency at which the amplitude of the signal begins to significantly change, marking the transition between two different spectral behaviors. In the acceleration or velocity spectrum of an earthquake signal, the corner frequency often indicates a transition between low-frequency content (governed by source size or rupture duration) and high-frequency decay. A high corner frequency indicates a small, fast rupture, whereas a low corner frequency is typical of larger or slower events.

The decay coefficients γ_X and nugget coefficients η_X are parameterized using empirical sigmoid functions. $\eta_X(f_{min})$ were employed to reflect the fact that the exponential decay of the correlations near f_{min} is much stronger than for frequencies farther away.

$$\gamma_E(f) = \gamma_{E,0} + \frac{\gamma_{E,1}}{1 + \exp(-\gamma_{E,3} \log(\frac{f}{\gamma_{E,2}}))} + \gamma_{E,4} \log(\frac{f}{\gamma_{E,2}})$$

$$\gamma_S(f) = \gamma_{S,0} + \frac{\gamma_{S,1}}{1 + \exp(-\gamma_{S,3} \log(\frac{f}{\gamma_{S,2}}))} + \frac{\gamma_{S,4}}{1 + \exp(-\gamma_{S,4} \log(\frac{f}{\gamma_{S,6}}))}$$

$$\eta_S(f) = \eta_{S,0} \log \frac{\max(\min(f, 4), 0.25)}{0.25} + \eta_{S,1} \log \frac{\max(f, 4)}{4}$$

$$\gamma_A(f) = \frac{\gamma_{A,0}}{1+\exp(-\gamma_{A,2}\log(\frac{\min(f,10)}{\gamma_{A,1}}))} - 1 + \frac{\gamma_{A,3}}{1+\exp(-\gamma_{A,5}\log(\frac{f}{\gamma_{A,4}}))}$$

$$\eta_A(f) = \frac{\gamma_{A,0}}{1+\exp(-\gamma_{A,2}\log(\frac{f}{\gamma_{A,1}}))} + (1 + \frac{\gamma_{A,3}}{1+\exp(-\gamma_{S,5}\log(\frac{f}{\gamma_{S,4}}))})$$

Also the standard deviations to be used in (14) are modeled through sigmoid functions:

$$\sigma_E(f) = s_{E,0} + \frac{s_{E,1}}{1+\exp(-s_{E,3}\log(f/s_{E,2}))} + \frac{s_{E,4}}{1+\exp(-s_{E,6}\log(f/s_{E,5}))} \tag{18}$$

$$\sigma_S(f) = s_{S,0} + \frac{s_{S,1}}{1+\exp(-s_{S,3}\log(f/s_{S,2}))} + \frac{s_{S,4}}{1+\exp(-s_{S,6}\log(f/s_{S,5}))} \tag{19}$$

$$\sigma_A(f) = s_{A,0} + s_{A,1}(\log\frac{\max(f,5)}{5})^2 \tag{20}$$

All coefficients used in the implementation are empirically derived from regression on seismic datasets as presented in Stafford (2017) [26].

2.3.1 Application to the Model

The previous correlation functions are used to compute the empirical correlation matrix of the log-Fourier components extracted from synthetic accelerograms. This matrix is then compared with the theoretical correlation structure defined above, through the following loss function:

$$\mathscr{L}_{\text{corr}} = \frac{1}{N}\sum_{i,j}\left(\rho_{i,j}^{\text{obs}} - \rho_{i,j}^{\text{model}}(f_i, f_j, f_c)\right)^2 \tag{21}$$

This approach encourages the simulated signals not only to match the marginal spectral shape, but also to replicate the realistic frequency correlation structure observed in empirical seismic records. It has been developed with the aim of enhancing the physical plausibility of synthetic accelerograms, while avoiding alternative conditioning strategies, such as concatenating the input signal with its low-frequency component to better control its shape, or imposing physical constraints that may be difficult to model.

2.4 Low-Frequency Conditioning

During the successful rise of machine learning-based seismic signal generation, hybrid approaches that combine numerical simulations with deep learning have also been developed, aiming to better condition the generation process with underlying earthquake physics. Among the works describing these approaches are [5, 17] and the more recent [6].

The methodology adopted in this work is therefore to further guide the denoiser using low-frequency seismic waves; in fact, they can be reliably simulated using numerical models based on known physics, reaching up to a maximum frequency between 1 Hz and 10 Hz [7]. The idea is that these simulations are considered to encode the "minimum observable physics": they capture the essential behavior of seismic wave propagation based on real-world physical parameters (e.g., fault geometry, soil structure); moreover, they are cheaper to simulate, as longer wavelengths (lower frequencies) require coarser spatial grids.
Therefore, to support the denoiser, a low-frequency conditioning $\mathbf{x}$ is incorporated during the backward process. Such information is obtained at training time by low-pass filtering the input $\mathbf{y}_0(t)$ at 2π Hz (or equivalently 1 unit of angular frequency), with a Butterworth filter to obtain $\mathbf{x}_0 = \mathscr{F}_{f\leq 2\pi\ Hz}(\mathbf{y}_0(t))$. In practice, the model NN_θ receives as input $(\mathbf{y}_\tau, \mathbf{x}_0)$ concatenated along

the channel dimension, which is then fed to the denoiser architecture (U-Net) and helps to statistically align the denoised sample $\mathbf{y}_{\tau-1}$ with $\mathbf{x_0}$, effectively recreating the low-to-high frequency mapping $(\mathbf{y}_\tau, \mathbf{x}_0) \longrightarrow \mathbf{y}_{\tau-1}$ by learning $p_\theta(\mathbf{y}_{\tau-1}|\mathbf{y}_\tau, \mathbf{x})$.

When deployed in real world applications, at inference time, to the pure noise $\mathbf{y}_\tau \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ will be associated a low-frequency signal obtained from physics-based simulation (pbs) $\mathbf{x} \sim p_{\text{pbs}}$. To this purpose one can turn to a validated dataset of hybrid simulations of past-earthquakes called BB-SPEED created by Paolucci et al [18]. BB-SPEEDset is a broadband (0-30 Hz) earthquake ground motions dataset, that contains synthetic accelerograms obtained from the physics-based numerical simulation code called SPEED (SPectral Elements in Elastodynamics with Discontinuous Galerkin) [14], developed at Politecnico di Milano, Italy.

Clearly, ignoring conditioning would be ideal, but it is not always possible if you want to obtain a certain degree of precision in the backward process. In the following, we will also investigate the scenario in which the same architecture is trained without the presence of conditioning and draw the appropriate conclusions.

3 Implementation

The following section outlines the steps taken to adapt the provided pre-existing diffusion model to the newly processed data and to the modified loss function based on the inter-frequency correlation approach described in section 2.3.

3.1 Dataset and estimation of corner frequencies

The starting dataset was taken from the "STanford EArthquake Dataset (STEAD): A Global Data Set of Seismic Signals for AI" [15]. In particular, among the 5 available chunks, we considered chunk2 which we further processed to extract accelerograms, convert the data into torch tensors and extract the corner frequencies (see below) following a procedure that can be found in the `STEAD_processing` folder . The result is approximately 60.000 accelerograms of seismic signals, each recorded in three channels corresponding to the spatial components (see Figure 3). The data originates from regions across the globe, with a higher concentration of recordings in North America and Europe.
Additional metadata, such as magnitude, recording station information and sampling rate, was used to estimate the corner frequency values f_c required by the correlation-based model. This was done by fitting a Brune-type source model [3] to the Fourier amplitude spectrum of each signal. The estimation process was carried out using the Python SourceSpec library [22]. This tool allows for automated and robust retrieval of source parameters, including f_c, using nonlinear curve fitting techniques. These frequency-dependent estimates are then used to normalize the spectral ordinates and compute the interfrequency correlation functions as described above.

The resulting dataset consists of two different components, a tensor of shape [60150, 3, 6000], where the last dimension represents the time evolution of the seismic signals across the three spatial components, and a tensor of [60150,1] containing the corresponding estimated corner frequency for each signal. To be able to process such dataset, some modifications were needed in the section of the code processing the input data, namely the `AugmentedDatasetSTEAD` class.

3.2 Cost function

As previously introduced in Subsection 2.3.1, the original cost function used during training, a trivial mean squared error (MSE) term measuring the "distance" between the model's predicted

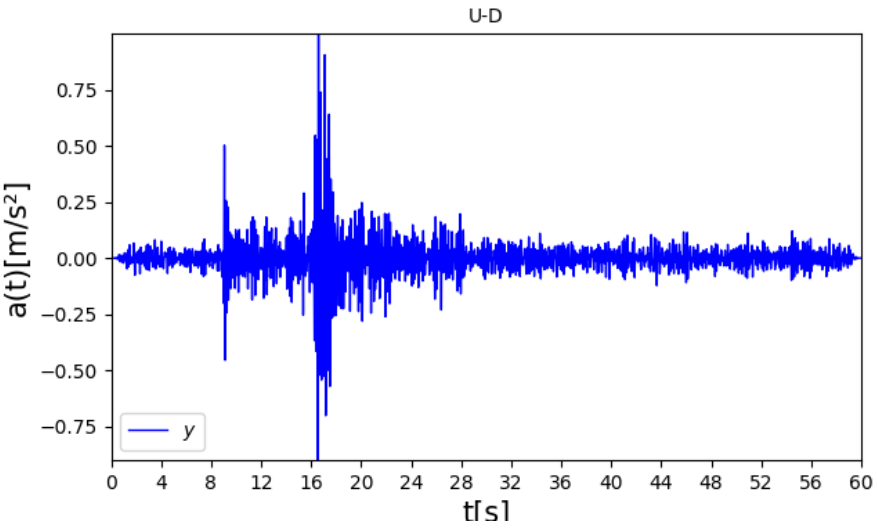


Fig. 3. 1D component of the acceleration of a seismic signal over time

noise (as it functions as a denoiser) and the true noise, was augmented with an additional term incorporating the correlation model.

The additional cost term takes as input the predicted noise $\mathbf{v}_\theta(\mathbf{y}_t, t)$, the original input corrupted with noise $\mathbf{y}_t$, the two parameters[2] α_t and σ_t used in the noising step, and the corner frequency value f_c associated with that signal.
Using the `torch.fft.fft` function along the time dimension, the signal is transformed into the frequency domain. A subset of frequency components, selected on a logarithmic scale, is considered based on the assumption in [26] that these components are approximately normally distributed. These are used to compute both:

- a normalized covariance matrix of the log-scaled frequency components,
- a correlation matrix derived from the interfrequency model, parameterized by the corner frequency (whose descriptions is in subsection 2.3).

The loss is finally computed as the mean squared error between these two matrices, encouraging the reconstructed signal to exhibit a realistic frequency correlation pattern.

```
def Loss_Covariant(v, alphas, sigmas, noised_reals, fc, device=None):

    v = v.to(device)

    # reconstruct from v the predicted clean image x0
    x0 = alphas * noised_reals - sigmas * v

    delta_f_T = 0.01   # frequency spacing
    T = x0.shape[-1]
    print(T)
    fs = T * delta_f_T  # sampling frequency

    dati_frequenza = torch.fft.fft(x0, dim=-1).real
```

[2] they correspond to $\alpha_t \longleftrightarrow \sqrt{\bar{\alpha}_t}$ and $\sigma_t \longleftrightarrow \sqrt{1-\bar{\alpha}_t}$ and are computed with the function `def get_alphas_sigmas(t)` that can be found in `diffusion/diff.py`

```
    frequenze = torch.fft.fftfreq(T, d=1/fs).to(device)

    positivi = frequenze > 0
    frequenze = frequenze[positivi]

    n = len(frequenze)
    log_frequencies = torch.logspace(
        torch.log10(frequenze[1]).item(),
        torch.log10(frequenze[n // 2]).item(),
        steps=n // 2,
        base=10.0,
        device=device
    )

    indici_log_frequencies = torch.searchsorted(frequenze, log_frequencies)
    dati_corrispondenti = dati_frequenza[:, :, indici_log_frequencies]

    X = dati_corrispondenti.flatten(0, 1)
    cov_matrix = torch.cov(X.T)
    std_dev = torch.sqrt(torch.diag(cov_matrix))
    correlation_matrix = cov_matrix / torch.outer(std_dev, std_dev)

    correlation_empirical = correlation(
        log_frequencies.unsqueeze(1), log_frequencies.unsqueeze(0), fc)

    return torch.mean((correlation_matrix - correlation_empirical) ** 2)
```

To integrate this new term with the standard mean squared error loss, a hyperparameter λ_{corr} was introduced to scale the correlation-based loss appropriately. This scaling ensures that the two loss components are on comparable magnitudes and that the correlation loss is properly weighted during training.

$$\mathscr{L} = \mathscr{L}_{\mathbf{v}} + \lambda_{\text{corr}} \mathscr{L}_{\text{corr}}$$

3.3 Architecture and training

The U-Net architecture is shown in figure 4.

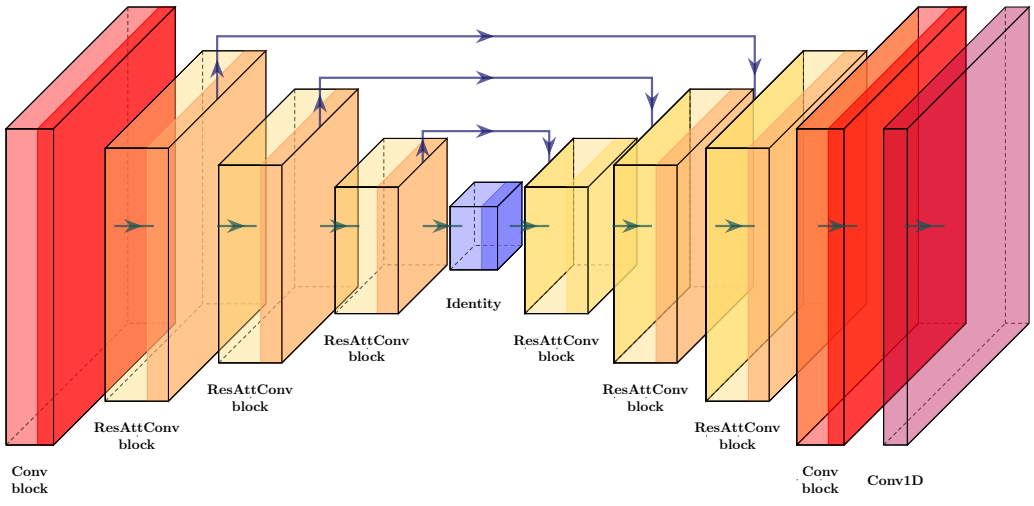


Fig. 4. Illustration of the U-Net architecture with a depth of 4.

The model, denoted as `DiffusionAttnUnet1DCond`, takes as input a noisy signal (a signal contaminated with gaussian noise), a diffusion timestep and an optional conditioning variable, and outputs an estimate of the noise, as previously explained.

The network consists of a deep encoder-decoder structure with symmetric skip connections (see Appendix A.1), following the U-Net structure as described in [19]. Each resolution level (`ResAttConv`) incorporates multiple successions of residually connected convolutional blocks (`ResConvBlock`), with optional self-attention modules (`SelfAttention1d`) that allow the model to capture long-range dependencies in the signal. The level at which attention is applied is configurable, and by default, attention is introduced only in the deepest layers to reduce computational cost. The implementation is recursive, progressively building up the U-Net structure from the deepest layer to the input layer.
The timestep information, used to progressively remove noise, is embedded using Fourier features and broadcast across the spatial dimension before being concatenated with the input signal. When present, the conditioning variable is also concatenated channel-wise, allowing the network to generate outputs guided by external information.

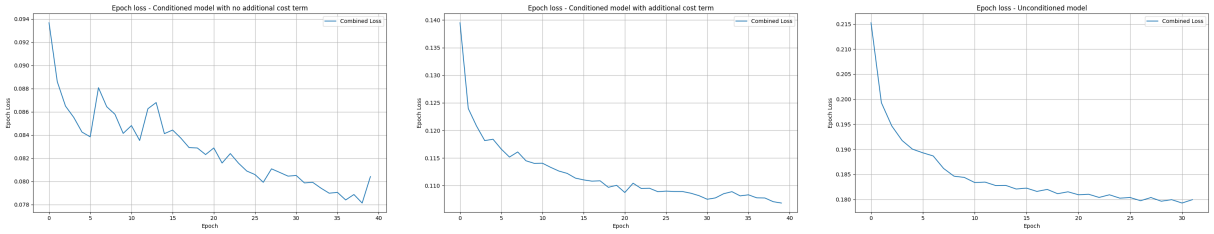
Three different models were trained to evaluate the impact of the new loss function, described above, and of the conditioning at the low frequencies on the generation of seismic signals. In all cases, the training was initialized from the same previously-trained model, which had been conditioned on lower-frequency components (three additional input channels embedding information from the lower frequencies, each computed dynamically at training time).

1. The first training consisted in fine-tuning the model previously trained with low-frequency conditioning, adapting it to the new dataset we created, as the original training was performed on a different dataset.

2. The second training reused the original conditioning mechanism and introduced the additional loss function designed to encode interfrequency correlation. The checkpoint could be loaded directly without requiring any architectural changes.

3. The third training aimed to isolate and evaluate the contribution of the loss function itself, independently from any conditioning, to obtain a model capable of generating data without requiring such conditioning or auxiliary input. To achieve this, it was necessary to exclude the three low-frequency conditioning channels from the input. Since the pretrained model expected 22 input channels (including the conditioning), a script was developed to adapt the checkpoint accordingly. Specifically, the code selectively trims the pretrained model's weights to match the new architecture with only 19 input channels and 3 output channels (instead of 6). The script loads the full state dictionary from the original checkpoint and modifies the weight tensors of the first and last convolutional blocks to remove the unused channels. The updated checkpoint, once saved, enables fine-tuning of the model without conditioning while preserving the valuable initialization from the previous training. This step was crucial to fairly assess the effect of the new loss function in isolation.

Regarding the three separate training runs, the necessary preliminary steps involved identifying an appropriate batch size and tuning the hyperparameters to minimize the cost function across epochs. This process proved particularly challenging due to memory limitations on the local architectures used for training, which constrained the maximum batch size to only 16 samples. As a result, each training cycle was computationally expensive, and hyperparameter tuning, especially of the learning rate and λ_{corr}, was slow and resource-intensive.
The optimizer employed during training was AdamW, a variant of Adam that decouples weight decay from the gradient update. This improves regularization and training stability, and it is



(a) Conditioned model, no additional cost term

(b) Conditioned model, additional cost term

(c) Unconditioned model, additional cost term

Fig. 5. Loss across epochs for the three trained models.

widely adopted in modern deep learning architectures.
Both the conditioned model variants were trained for 40 epochs, after which the mean epoch loss and the weight updates exhibited no significant variation. The unconditioned model with the additional cost term required only 32 epochs, as the training loss reached a plateau more quickly. This stability is evident in Figure 5, where the absolute values show only minimal changes across the epochs.

We briefly mention that another architecture was developed but never further explored within this scope. Nonetheless, we think it could be a promising idea and leave further details in Appendix A.2.

4 Results

4.1 Evaluation metrics

To evaluate the quality of the generated signals, in addition to visual inspection for assessing deviations from the original input, relying solely on the Mean Squared Error (MSE) is often insufficient, as it does not account for temporal dependencies or structural patterns that influence the perceptual quality of the signal.

Therefore, a commonly adopted metric in the field of deep learning is the Signal-to-Noise Ratio (SNR) [30]. This metric quantifies the fidelity of a reconstructed signal by comparing the power of the original (clean) signal to the power of the noise, defined as the difference between the target and the predicted signal. It is computed as follows:

$$\mathrm{SNR}(\text{target, pred}) = 10 \cdot \log_{10} \left(\frac{\sum_{i=0}^{T} \text{target}i^2}{\sum i = 0^{T} (\text{target}_i - \text{pred}_i)^2} \right)$$

A higher SNR value indicates that the predicted signal closely matches the original target, whereas lower values, including negative ones, reflect higher reconstruction noise.

Another metric employed to evaluate the synthetic signals is the Structural Similarity Index Measure (SSIM) [16], which quantifies the similarity between two signals by considering structural information, rather than just point-wise differences (like the MSE). Given the original clean signal `target` and the model's predicted signal `pred`, the SSIM is defined as:

$$\mathrm{SSIM}(\text{target}, \text{pred}) = \frac{(2\mu_{\text{target}}\mu_{\text{pred}} + C_1)(2\sigma_{\text{target,pred}} + C_2)}{(\mu_{\text{target}}^2 + \mu_{\text{pred}}^2 + C_1)(\sigma_{\text{target}}^2 + \sigma_{\text{pred}}^2 + C_2)}$$

where:

- μ_{target}, μ_{pred} are the local means of the target and predicted signals, respectively;
- σ^2_{target}, σ^2_{pred} are the local variances;
- $\sigma_{\text{target,pred}}$ is the local covariance between `target` and `pred`;
- C_1 and C_2 are constants introduced to stabilize the division in case of weak denominator terms. They are defined as $C_1 = (K_1 L)^2$, $C_2 = (K_2 L)^2$, where L is the dynamic range of the signal ($L = 1$ if the signals are normalized), and $K_1 = 0.01$, $K_2 = 0.03$ by default.

The SSIM value ranges from -1 to 1, where:

- SSIM $= 1$ indicates perfect structural similarity;
- SSIM $= 0$ indicates no structural correlation;
- negative values may occur in case of strong structural dissimilarity.

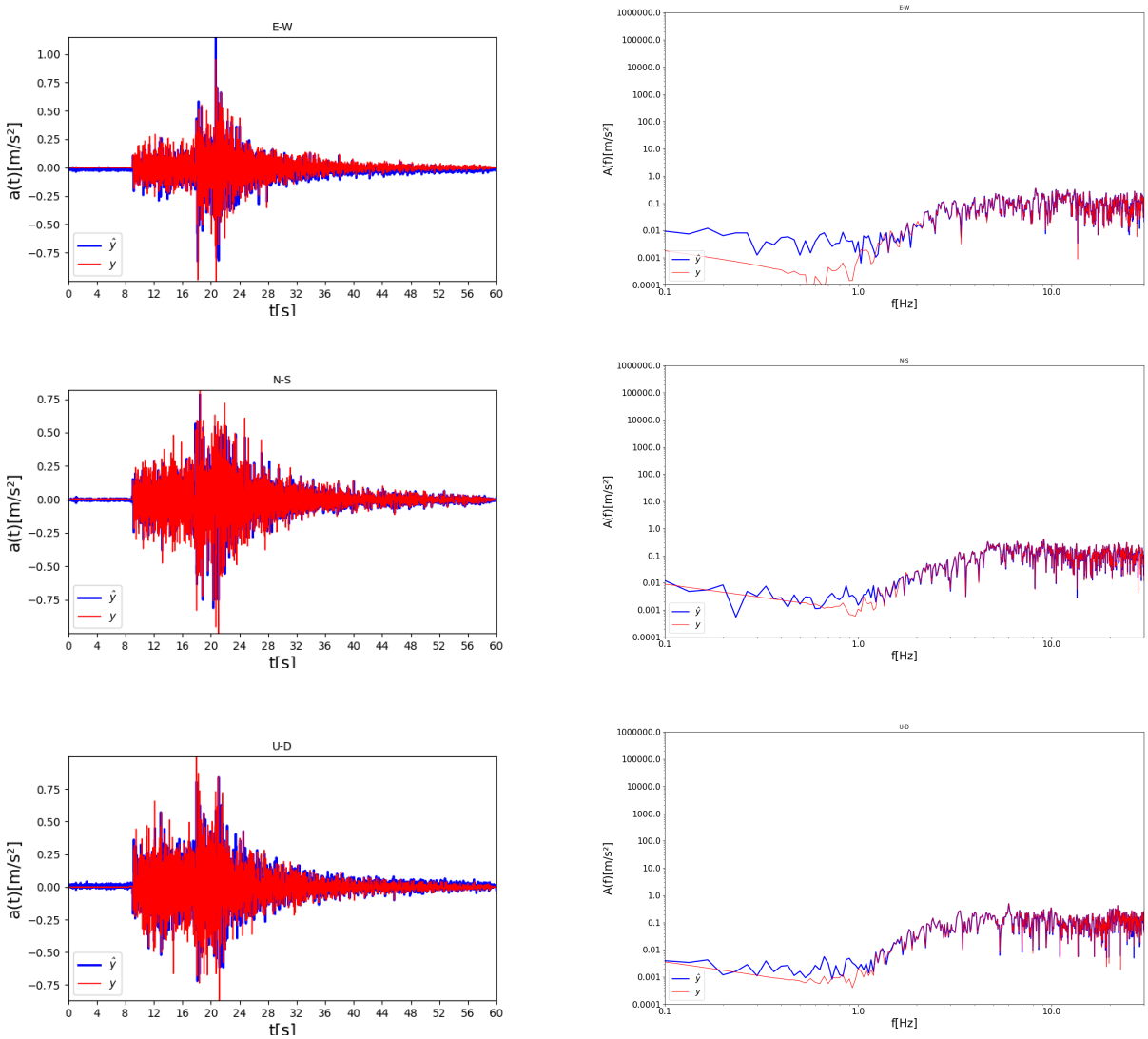


Fig. 6. Generated signals $\hat{y}$ and corresponding inputs y by the conditioned model with additional correlation-based cost term, in time and frequency domain, for the three spatial directions (E-W, N-S, U-D). SNR: 4.7332, SSIM: 0.7404

In the context of seismic accelerograms denoising, SNR quantifies the ratio of useful signal to background noise in seismic signals; it measures how effectively a model preserves the original signal while reducing noise. A higher SNR indicates less noise interference and clearer seismic datas. SSIM, on the other hand, evaluates the similarity in structure and waveform shape between original and reconstructed seismic signals; it assesses alignment in amplitude, timing of peaks, troughs, and frequency content. A higher SSIM value indicates better preservation of structural fidelity and waveform integrity. Together they provide complementary insights into the performance of seismic signal reconstruction models.

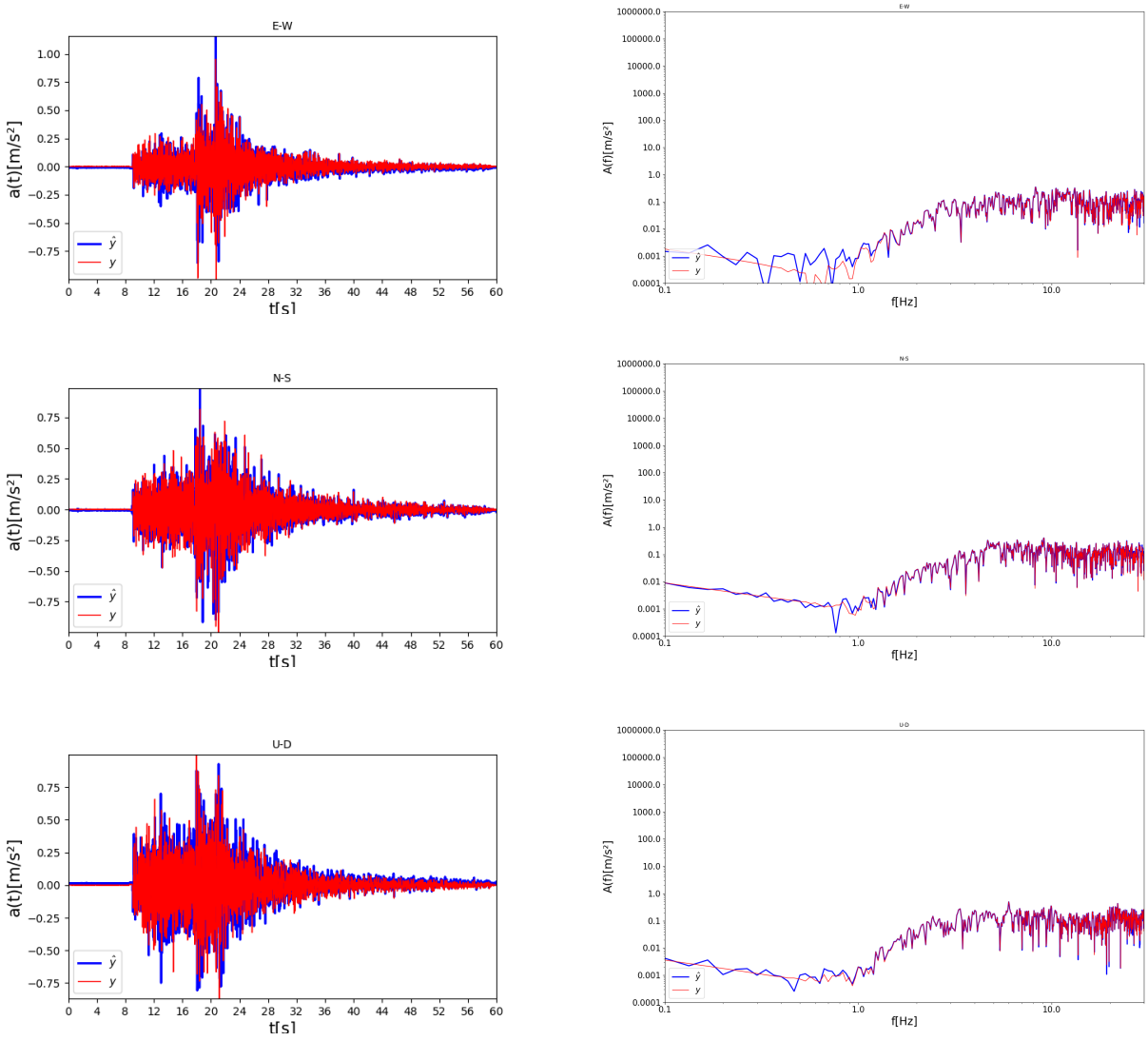


Fig. 7. Generated signals $\hat{y}$ and corresponding inputs y by the conditioned model with no additional cost term, in time and frequency domain, for the three spatial directions (E-W, N-S, U-D). SNR: 4.1204, SSIM: 0.7490

4.2 Comparison of generated signals

The following subsection presents examples of signals generated by the three trained models, with the aim of assessing the quality of the outputs and quantifying the influence of low-frequency conditioning and the additional cost term on the generation process. All signals were produced

using Algorithm 4 with a schedule of 100 diffusion steps and $\eta = 0$; the code for both conditional and unconditional generation can be found in `diffusion/generate_diffusion.py`.
The figures show the generations of signals from the same selected 3D seismic input, split into its three directional components, using the three trained models. Additional examples can be generated by following the instructions provided in the README file of the GitHub repository; moreover, various generated signals, along with corresponding metrics, are available in the Google Drive folder.

We begin our testing deploying the model conditioned at low frequencies and with the additional cost term introduced in Section 2.3.1. The signals in Figure 6 exhibit reasonable adherence with respect to the input, both in the time domain and in the frequency one. The overall structure of the signal is consistent with the original one, and the SNR value indicates a fair level of agreement between the predicted and reference signals in terms of amplitude over time.
A similar behavior is shown by the model that presents only conditioning with no additional frequency correlation cross term, as one can deduce from Figure 7. The inclusion of the additional correlation-based cost term leads to a slight overall improvement in signal quality, as shown by the marginal increase in both SNR and SSIM. However, the effect remains subtle, suggesting that while the cost term contributes to regularization, further tuning or alternative formulations may be needed to achieve more substantial gains.

To have a better understanding of the differences between the two models, the visual inspection is supported by the quantitative results reported in Table 1, where average metrics were computed over a set of 600 generations.

	Mean SNR	**Mean SSIM**
Conditioned model with no additional cost term	4.1345	0.6815
Conditioned model with the additional cost term	4.1359	0.6271

Table 1. Mean SNR and SSIM metrics for 600 signals generated by the two conditional models.

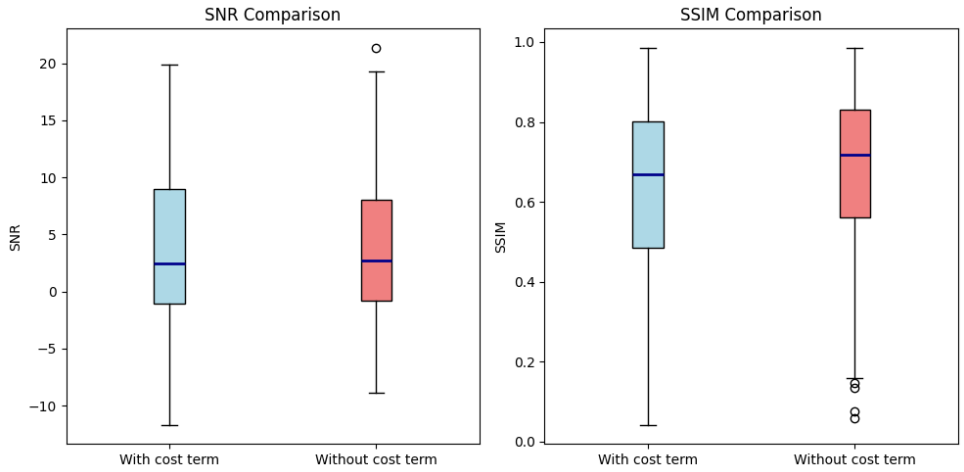


Fig. 8. SNR and SSIM distribution across 600 generated test signals.

Under equal number of training epochs, the conditional model with the additional cost term exhibits lower structural similarity (SSIM) compared to the baseline model, although noise reduction performance (SNR) is slightly higher. The drop in SSIM may be attributed to the shift in the training objective induced by the additional loss term, and could potentially be miti-

gated by extended training or alternative loss weighting strategies. The model incorporating the correlation-based term demonstrates strong performance across various scenarios, with SNR values often exceeding 10 and SSIM scores approaching 1. It is worth noting, however, that both models occasionally yield poor results on less frequent or more challenging signals, which contributes to the lower average metrics. These observations support the effectiveness of the proposed regularization strategy, while also highlighting the need for further tuning in rare-case scenarios.

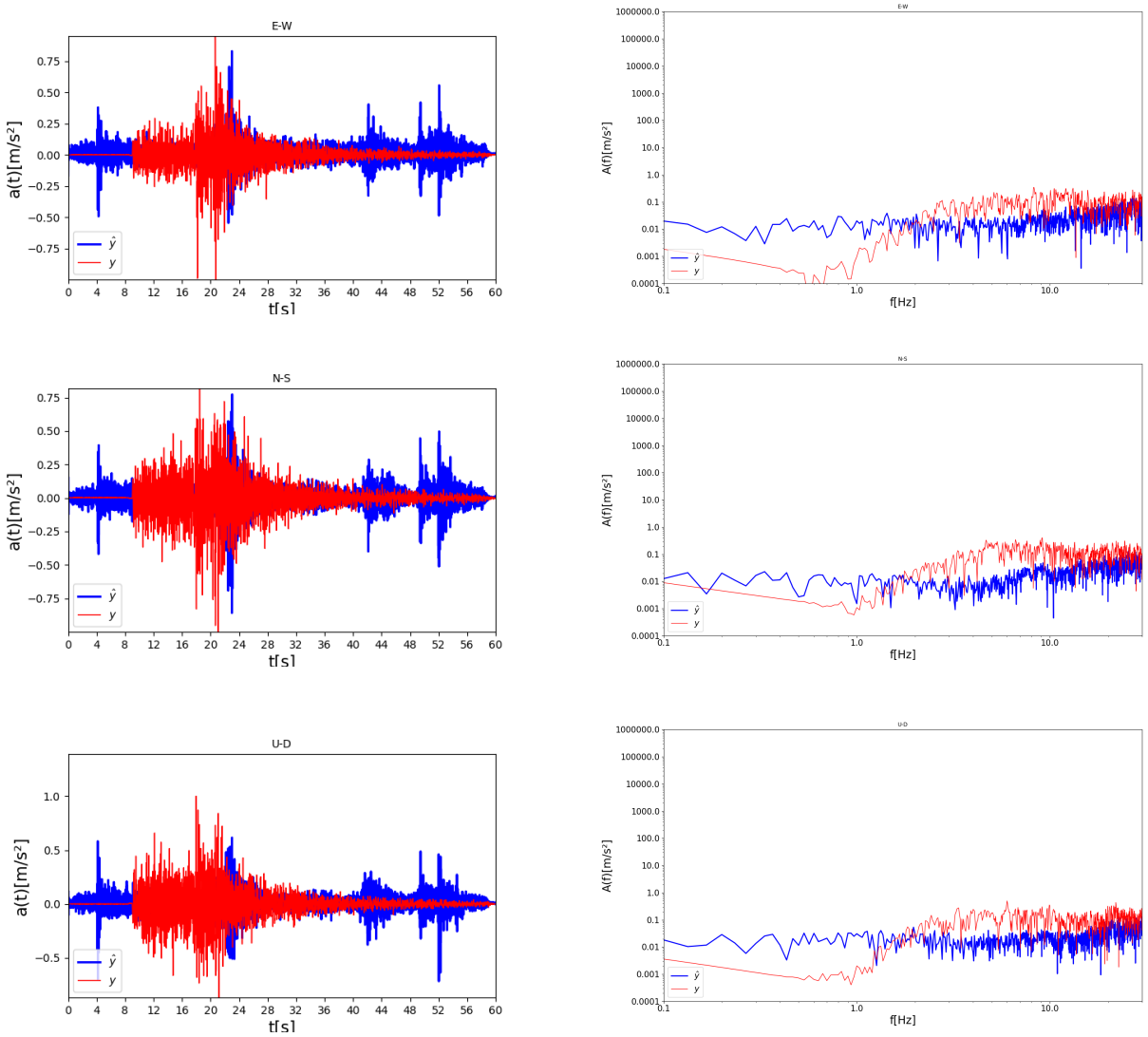


Fig. 9. Generated signals $\hat{y}$ and corresponding inputs y by the unconditioned model with only the additional correlation-based cost term, in time and frequency domain, for the three spatial directions (E-W, N-S, U-D). SNR: -4.6034, SSIM: 0.2402

The boxplots in Figure 8 provide an additional comparison of the SNR and SSIM distributions obtained from the two diffusion model variants across 600 tested signals. Regarding SNR, both models exhibit comparable median values and similar distributions, suggesting that the inclusion of the cost term does not significantly impact the overall signal-to-noise ratio. The model without the cost term shows a single high-value outlier, indicating occasional performance spikes, whereas the model with the cost term demonstrates more consistent behavior. For SSIM, the model without the cost term achieves a slightly higher median value, which suggests better preservation of

structural features in most cases. However, it also presents a larger number of low-value outliers, pointing to a reduced robustness and higher variability in quality across different signals. On the other hand, the model with the cost term shows a more compact distribution with fewer outliers, indicating more stable and reliable performance, especially in worst-case scenarios.
Overall, these results highlight a trade-off between average performance and consistency: while the model without the cost term may perform better on average in terms of SSIM, the inclusion of the cost term leads to greater robustness and fewer failure cases.

Finally, we investigate the impact of low-frequency conditioning on the quality of the generated signals. In fact, as previously discussed, the ideal scenario would be to set aside the external low-frequency information and rely solely on the architecture's ability to coherently generate seismic acceleration data. Figure 9 displays the outputs produced by a model trained without conditioning, using only the standard MSE loss combined with the additional structural cost term. The results highlight a poor generative capability of the model when it is not guided by the additional conditional information. In particular, the negative SNR values indicate that the noise component dominates over the generated signal, suggesting a significant mismatch with respect to the target. These findings seem to suggest that incorporating only the correlation cost term is not sufficient to ensure the generation of structurally coherent signals, at least within the current U-Net architecture.

	Mean SNR	**Mean SSIM**
Conditioned model with only the additional cost term	-5.8543	0.3928

Table 2. Mean SNR and SSIM metrics for 125 signals generated by the model that presents only the interfrequency correlation loss and no low frequency conditioning.

5 Conclusions and Future Developments

This work presented a pipeline for generating synthetic seismic acceleration signals that are consistent with the statistical and structural patterns observed in real data. The primary motivation for employing a deep learning-based approach lies in the limitations of traditional physics-based simulations. Specifically, the numerical solution of the elastodynamic equation often suffers from inaccuracies due to inaccurate representations of the Earth's structure and materials and challenging boundary condition specifications. Furthermore, embedding such constraints into a neural architecture like a U-Net is non-trivial and inherits the same modelling challenges.
To address these limitations, we proposed a novel regularization strategy that introduces an additional term in the loss function, designed to capture the interfrequency correlation structure of the logarithmic Fourier spectral components of seismic signals. This strategy guides the model to generate outputs that are not only statistically coherent but also more plausible from a physical standpoint, without explicitly enforcing the underlying physical equations.

The already available U-Net architecture was trained in three distinct settings: fine-tuning a pretrained model with low-frequency conditioning on our processed dataset; training with both low-frequency conditioning and the new correlation-based loss term; training without conditioning, using only the additional loss. These configurations allowed us to assess the individual and combined contributions of the conditioning and the structural loss term.
The experimental results demonstrate that low-frequency conditioning is essential to enable the model to generate realistic signals, especially in the time domain. The inclusion of the correlation-based loss term, although not sufficient on its own, provides a slight improvement in the spectral coherence of the generated signals. Quantitative metrics support the added value of combining conditioning with the proposed regularization strategy, particularly in enhancing the model's robustness across frequencies.

Future extensions of this work could explore the integration of the correlation-based cost term to more advanced generative architectures, such as transformer-based diffusion models [11], which may offer better scalability and expressiveness. Additionally, the availability of greater computational resources (multi-GPU setups and larger memory) would allow for more extensive hyperparameter tuning and training on larger datasets, ultimately improving the quality and reliability of the synthetic signals produced.

References

[1] K. Aki and P. G. Richards. *Quantitative seismology*. 2002.

[2] J.-H. Bastek, W. Sun, and D. M. Kochmann. Physics-informed diffusion models. *arXiv preprint arXiv:2403.14404*, 2024.

[3] J. N. Brune. Tectonic stress and the spectra of seismic shear waves from earthquakes. *Journal of geophysical research*, 75(26):4997–5009, 1970.

[4] H. T. Erdinc, R. Orozco, and F. J. Herrmann. Generative geostatistical modeling from incomplete well and imaged seismic observations with diffusion models. *arXiv preprint arXiv:2406.05136*, 2024.

[5] M. A. Florez, M. Caporale, P. Buabthong, Z. E. Ross, D. Asimaki, and M.-A. Meier. Data-driven synthesis of broadband earthquake ground motions using artificial intelligence. *Bulletin of the Seismological Society of America*, 112(4):1979–1996, 2022.

[6] H. Gabrielidis, F. Gatti, and S. Vialle. Physics-based super-resolved simulation of 3d elastic wave propagation adopting scalable diffusion transformer. *arXiv preprint arXiv:2504.17308*, 2025.

[7] F. Gatti, S. Touhami, F. Lopez-Caballero, R. Paolucci, D. Clouteau, V. A. Fernandes, M. Kham, and F. Voldoire. Broad-band 3-d earthquake simulation at nuclear site by an all-embracing source-to-structure approach. *Soil Dynamics and Earthquake Engineering*, 115:263–280, 2018.

[8] C. F. Higham, D. J. Higham, and P. Grindrod. Diffusion models for generative artificial intelligence: An introduction for applied mathematicians. *arXiv preprint arXiv:2312.14977*, 2023.

[9] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[10] J. Ho, T. Salimans, A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022.

[11] G. Hugo, F. Gatti, and V. Stephane. Physics-based super-resolved simulation of 3d elastic wave propagation adopting scalable diffusion transformer. *Available at SSRN 5228055*, 2025.

[12] Q. Li and L. Shen. Dual-branch interactive cross-frequency attention network for deep feature learning. *Expert Systems with Applications*, 254:124406, 2024.

[13] X. Li, J. Thickstun, I. Gulrajani, P. S. Liang, and T. B. Hashimoto. Diffusion-lm improves controllable text generation. *Advances in neural information processing systems*, 35:4328–4343, 2022.

[14] I. Mazzieri, M. Stupazzini, R. Guidotti, and C. Smerzini. Speed: Spectral elements in elastodynamics with discontinuous galerkin: A non-conforming approach for 3d multi-scale problems. *International Journal for Numerical Methods in Engineering*, 95(12):991–1010, 2013.

[15] S. Mousavi, W. L. Ellsworth, W. Zhu, and G. C. Beroza. STEAD: A global dataset of seismic signals for ai. `https://github.com/smousavi05/STEAD`, 2020.

[16] P. Ndajah, H. Kikuchi, M. Yukawa, H. Watanabe, and S. Muramatsu. Ssim image quality metric for denoised images. In *Proc. 3rd WSEAS Int. Conf. on Visualization, Imaging and Simulation*, pages 53–58, 2010.

[17] R. Paolucci, F. Gatti, M. Infantino, C. Smerzini, A. G. Özcebe, and M. Stupazzini. Broadband ground motions from 3d physics-based numerical simulations using artificial neural networks. *Bulletin of the Seismological Society of America*, 108(3A):1272–1286, 2018.

[18] R. Paolucci, C. Smerzini, and M. Vanini. Bb-speedset: A validated dataset of broadband near-source earthquake ground motions from 3d physics-based numerical simulations. *Bulletin of the Seismological Society of America*, 111(5):2527–2545, 2021.

[19] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[20] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35: 36479–36494, 2022.

[21] T. Salimans and J. Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

[22] C. Satriano. SourceSpec – earthquake source parameters from p- or s-wave displacement spectra (v1.6), 2022. URL `https://doi.org/10.5281/zenodo.6954238`.

[23] P. M. Shearer. *Introduction to seismology*. Cambridge university press, 2019.

[24] D. Shu, Z. Li, and A. B. Farimani. A physics-informed diffusion model for high-fidelity flow field reconstruction. *Journal of Computational Physics*, 478:111972, 2023.

[25] J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[26] P. J. Stafford. Interfrequency correlations among fourier spectral ordinates and implications for stochastic ground-motion simulation. *Bulletin of the Seismological Society of America*, 107(6):2774–2791, 2017.

[27] F. Wang, X. Huang, and T. Alkhalifah. Controllable seismic velocity synthesis using generative diffusion models. *Journal of Geophysical Research: Machine Learning and Computation*, 1(3):e2024JH000153, 2024.

[28] W. Yin, D. Zhou, and R. Nie. Di-unet: dual-branch interactive u-net for skin cancer image segmentation. *Journal of Cancer Research and Clinical Oncology*, 149(17):15511–15524, 2023.

[29] M. Z. Yousif, L. Yu, S. Hoyas, R. Vinuesa, and H. Lim. A deep-learning approach for reconstructing 3d turbulent flows from 2d observation data. *Scientific Reports*, 13(1):2529, 2023.

[30] T. Yuan, W. Deng, J. Tang, Y. Tang, and B. Chen. Signal-to-noise ratio: A robust distance metric for deep metric learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4815–4824, 2019.

A Appendix

A.1 DiffusionAttnUnet1DCond

Internal structure of the model:

```
class DiffusionAttnUnet1DCond(nn.Module):
    def __init__(
        self,
        io_channels=3,  # if you plan on using conditional input, set this to 6
        latent_dim=0,
        depth=4,        # max depth of 5 with [batch, channels, 6000] data
        n_attn_layers=3,
        c_mults=[128, 128, 256, 256] + [512] * 10

    ):
        super().__init__()
        self.timestep_embed = FourierFeatures(1, 16)
        self.latent_dim = latent_dim
        attn_layer = depth - n_attn_layers - 1
        block = nn.Identity()
        conv_block = ResConvBlock

        for i in range(depth, 0, -1):
            c = c_mults[i - 1]
            if i > 1:
                c_prev = c_mults[i - 2]
                add_attn = i >= attn_layer and n_attn_layers > 0
                block = SkipBlock(
                    Downsample1d("cubic"),
                    conv_block(c_prev, c, c),
                    SelfAttention1d(
                        c, c // 32) if add_attn else nn.Identity(),
                    conv_block(c, c, c),
                    SelfAttention1d(
                        c, c // 32) if add_attn else nn.Identity(),
                    conv_block(c, c, c),
                    SelfAttention1d(
                        c, c // 32) if add_attn else nn.Identity(),
                    block,
                    conv_block(c * 2 if i != depth else c, c, c),
                    SelfAttention1d(
                        c, c // 32) if add_attn else nn.Identity(),
                    conv_block(c, c, c),
                    SelfAttention1d(
                        c, c // 32) if add_attn else nn.Identity(),
                    conv_block(c, c, c_prev),
                    SelfAttention1d(c_prev, c_prev //
                                    32) if add_attn else nn.Identity(),
                    Upsample1d(kernel="cubic")
                )
            else:
                block = nn.Sequential(
                    conv_block(io_channels + 16 + self.latent_dim, c, c),
                    conv_block(c, c, c),
                    conv_block(c, c, c),
                    block,
```

```
                    conv_block(c * 2, c, c),
                    conv_block(c, c, c),
                    conv_block(c, c, io_channels, is_last=True),
                )
        self.net = block
        self.lastconv = nn.Conv1d(io_channels, 3, 3, padding="same")

        with torch.no_grad():
            for param in self.net.parameters():
                param *= 0.5

    def forward(self, input, t, cond=None):
        timestep_embed = expand_to_planes(
            self.timestep_embed(t[:, None]), input.shape)
        inputs = [input, timestep_embed]
        if cond is not None:
            inputs.append(cond)
        out = self.net(torch.cat(inputs, dim=1))
        out = self.lastconv(out)
        return out
```

A.2 Dual Branch Model

Taking inspiration from [12, 28] we developed an alternative model: a dual encoding branches - single decoding branch UNet with cross attention layers. The main idea was to enhance the performance at low frequencies ($\leq 1\ Hz$) by feeding the two encoding branches two separate inputs which will then attend to each other in the encoding phase through multi-head cross attention layers (MHCA). These two inputs could be formulated into two flavors:

- unconditional: split $\mathbf{y}_t$ using high and low pass filtering feeding the results to the respective branches
- conditional: feed $\mathbf{y}_t$ to one branch and a low frequency condition $\mathbf{x}_0$ to the other

This idea was not fully explored within the scope of the present work, as the model's increased complexity and less linear gradient propagation led to significantly longer training times. Nevertheless, we believe that, particularly in its conditioned version, it has the potential to further enhance the performance of the current approach. The full implementation can be found in the `extra` folder of the GitHub repository.

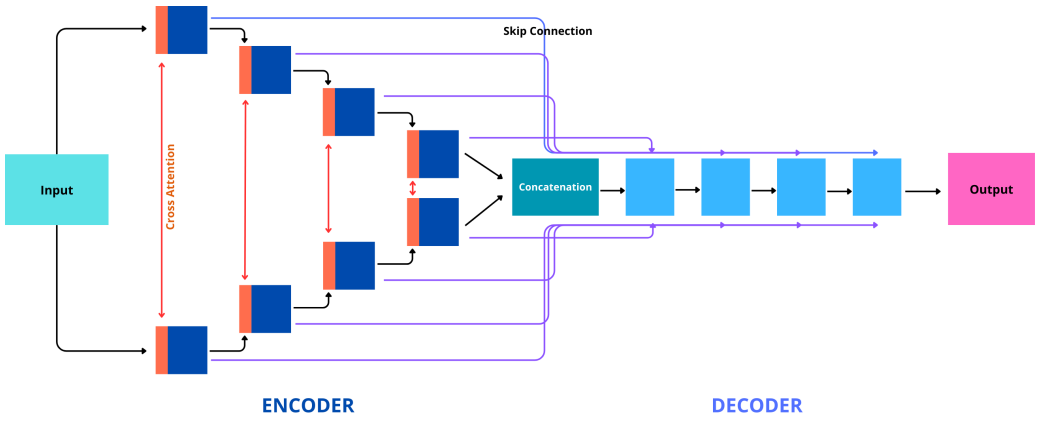


Fig. 10. Dual-branch model scheme.