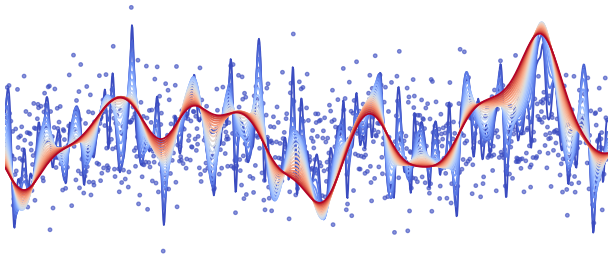
Generative Modelling for Time Series

From Linear Systems to Flow-Based Models

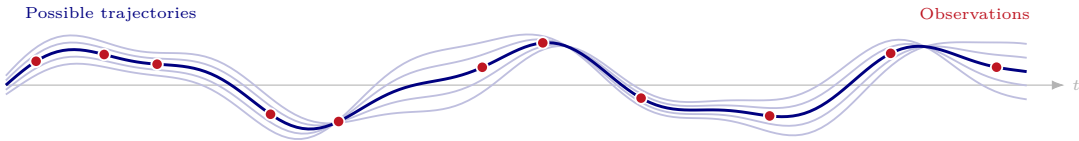
Felipe Tobar

Department of Mathematics
Imperial College London



Why model time series?

To understand how a **quantity of interest evolves in time**, usually from partial observations.



inverse problem formulation: partial observations explain several possible realisations

Application domains:

Life sciences, climate, audio, finance, health, astronomy, mechanical systems, and more.

Shared tasks:

unconditional generation, forecasting, denoising, imputation, compression, classification, and more.

Learning about the data's generative mechanism can help understanding its underlying causes.

What's the challenge?

Representation learning

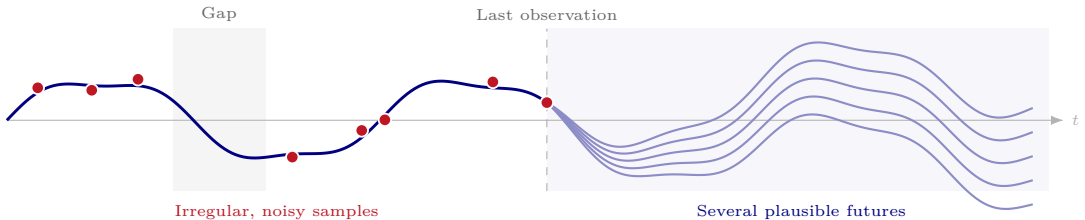
Combinatorial explosion: distributed (time) vs compact (spectral) models.

Multiple patterns

Short/long memory, oscillations, transients, changepoints, and roughness.

Partial & noisy observations

Irregular grids, missing values, noise, indirect measurements.



It is not clear where the information lies: Relevant features of the system's underlying state usually arise from interactions across lags, rather than pointwise values.

Historical overview

Patterns, randomness, spectrum, latent variables, learning and transport.

Early observations	**Records and recurring patterns** Astronomical observations; periodicity; fitting curves to data.
Early–mid 20th century	**Random processes and linear systems** Brownian motion; autoregression; spectra and optimal filtering.
Later 20th century	**Structured statistical models** ARIMA methodology; state-space inference; Bayesian process priors.
1990s–2010s	**Learned representations** Gaussian processes in machine learning; neural sequences and latent variables.
2010s–today	**Learned transport** Flows and diffusion; flow matching on grids and in function spaces.

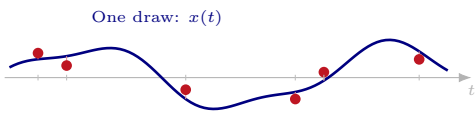
Setting and notation

i) In general[1], we will consider time series as **stochastic processes**:

$$X = \{X(t) : t \in \mathcal{T}\}, \quad X(t) \in \mathbb{R}^d, \qquad \mathcal{T} = \mathbb{Z} \text{ or } \mathcal{T} \subseteq \mathbb{R}.$$

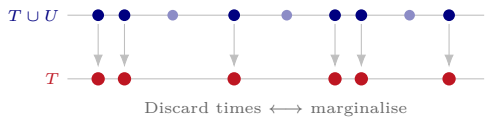
ii) On a finite grid, we observe a partial view of a **realisation** x.

$$T = \{t_1, \ldots, t_n\}, \qquad Y_i = X(t_i) + V_i.$$



The times can be irregular or incomplete; V_i describes measurement noise.

iii) Observations across grids should describe the same process.



$$p_T(\mathbf{x}_T) = \int p_{T \cup U}(\mathbf{x}_T, \mathbf{x}_U)\, \mathrm{d}\mathbf{x}_U.$$

Reordering the times must simply reorder the corresponding variables.

Compatible finite-dimensional laws define a process law: the idea of Kolmogorov's extension theorem.

[1] There will be interesting exceptions.

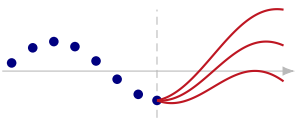
The generative-model approach

Learning a prior allows for analysis, conditional/unconditional generation and addressing inverse problems:

$$\mathbf{Y}_{\text{obs}} = H_{\text{obs}}\mathbf{X}_T + \mathbf{V}, \qquad \underbrace{p(\mathbf{x}_T \mid \mathbf{y}_{\text{obs}})}_{\text{posterior}} \propto \underbrace{p(\mathbf{y}_{\text{obs}} \mid \mathbf{x}_T)}_{\text{likelihood}} \underbrace{p(\mathbf{x}_T)}_{\text{prior}},$$

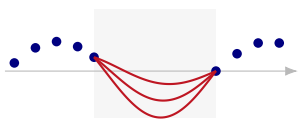
where the operator H_{obs} selects/transforms, and $\mathbf{V}$ models measurement noise. Different H_{obs} recover:

Forecasting



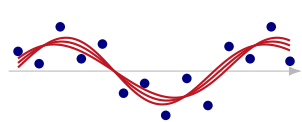
$p(\mathbf{x}_{\text{future}} \mid \mathbf{y}_{\text{past}})$

Imputation



$p(\mathbf{x}_{\text{missing}} \mid \mathbf{y}_{\text{obs}})$

Denoising



$p(\mathbf{x}_T \mid \mathbf{y}_{\text{obs}})$

• Measurements — Conditional trajectories

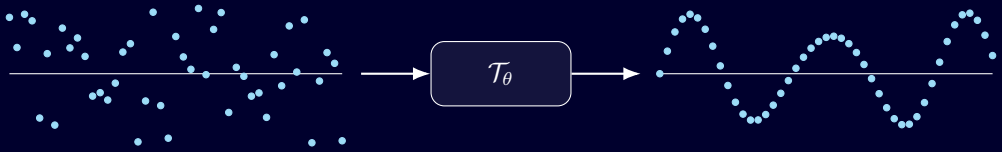
A general-purpose inference framework
(which is not always feasible)

Roadmap

Recurring idea: transform a *simple* random source into a *structured* signal.

I	**Classical methods**	Shape randomness with linear transformations Memory, state-space dynamics, and spectra
II	**Gaussian processes**	Move from grids to continuous domains Kernels, sampling, conditioning, and learning
III	**Flow-based models**	Learn nonlinear transformations Latent variables, alignment, and functional transports
IV	**Examples**	Formulate time series tasks as inverse problems Spectral estimation, superresolution, band-pass filtering

I - Classical Methods



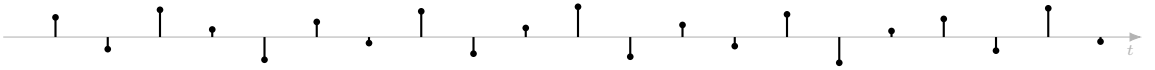
Learning outcomes

Turn unstructured noise into temporal dependence.

Study that structure in time and frequency.

White noise as driving input

How to choose a signal with a **maximal degree** of variability?



Definition (Entropy rate)

The average uncertainty per sample of a stationary process is

$$\bar{h}(Z) = \lim_{t\to\infty} \frac{h(Z_{1:t})}{t} = \lim_{t\to\infty} h(Z_t \mid Z_{1:t-1}),$$

where $h(Z) = -\mathbb{E}\log p(Z)$. The chain rule and stationarity give the conditional form.



$$\bar{h}(Z) = \lim_{t\to\infty} h(Z_t \mid Z_{1:t-1}) \;\le\; h(Z_t) \;\le\; \tfrac{1}{2}\log(2\pi e\sigma^2).$$

Independence: knowing the past does not reduce uncertainty.

Gaussianity: the largest entropy at fixed variance σ^2.

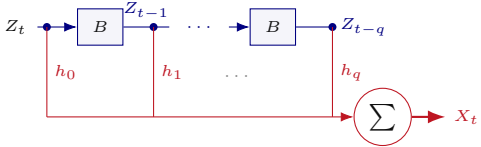
$\Longrightarrow$ **We choose independent Gaussian RVs:** $Z_1, Z_2, \ldots, Z_n \sim \mathcal{N}(0, \sigma^2 I)$ [aka, Gaussian white noise]

Inducing temporal structure

1) A finite linear filter.

Z_t: zero-mean white noise, $\mathrm{Var}(Z_t) = \sigma^2$.

$$X_t = \sum_{j=0}^{q} h_j Z_{t-j}.$$



2) Infinite memory is allowed:

$$X_t = \sum_{j=0}^{\infty} h_j Z_{t-j}, \qquad \sum_{j=0}^{\infty} h_j^2 < \infty.$$

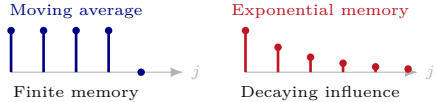
Finite variance is required.

3) Shared noise creates correlation.

$$\begin{aligned}\gamma_X(k) &= \mathrm{Cov}(X_t, X_{t-k}) \\ &= \sigma^2 \sum_{j=0}^{\infty} h_j h_{j+|k|}.\end{aligned}$$

Dependence comes from overlap between shifted copies of h. Weak stationarity.

4) Choose the memory through h_j.



Moving average: $h_j = 1/m$ for $0 \le j < m$; zero otherwise.

Exponential: $h_j = \phi^j$, $|\phi| < 1$, gives

$$X_t = \phi X_{t-1} + Z_t, \qquad \gamma_X(k) = \frac{\sigma^2 \phi^{|k|}}{1 - \phi^2}.$$

AR, MA, ARMA, and ARIMA families

$$\underbrace{X_t - \sum_{i=1}^{p} \phi_i X_{t-i}}_{\Phi} = \underbrace{Z_t + \sum_{j=1}^{q} \theta_j Z_{t-j}}_{\Theta}, \qquad Z_t \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2).$$

Different mechanisms

AR(p)	Feedback from past values; $q = 0$.
MA(q)	Finite noise memory; $p = 0$.
ARMA(p, q)	Both feedback and noise memory.
ARIMA(p, d, q)	ARMA for $\Delta^d X_t$, where $\Delta X_t = X_t - X_{t-1}$.

Properties

Stationarity (except ARIMA)
Time-invariant mean and lag covariance.

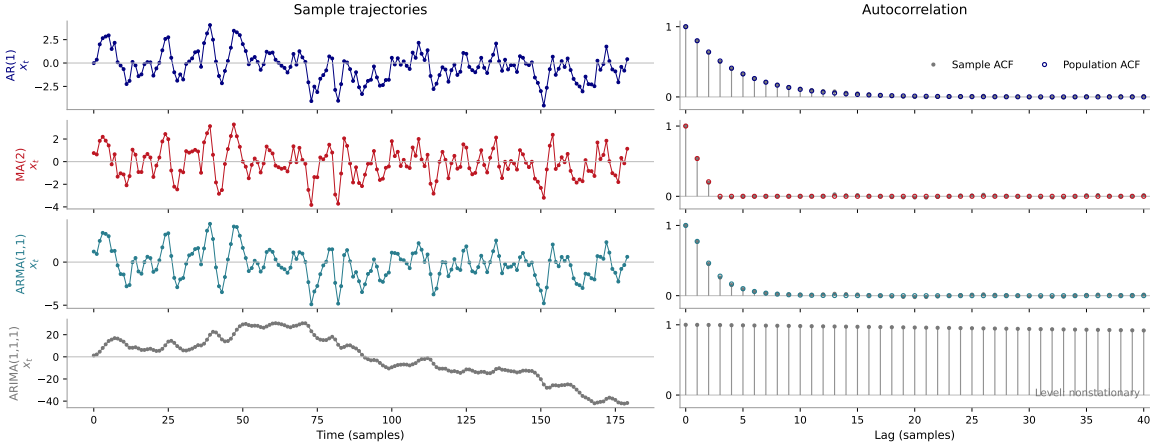
Causal stability
Φ controls summability of the response.

Invertibility
Θ controls recovery of Z_t from $X_t, X_{t-1}, \ldots$

Parameter learning: least squares for AR; numerical likelihood for MA/ARMA. Orders selected via crossvalidation, AIC, and residual autocorrelation.

Example: AR, MA, ARMA, and ARIMA in practice

Shared realisation of the driving **Gaussian white noise realisation** across models.

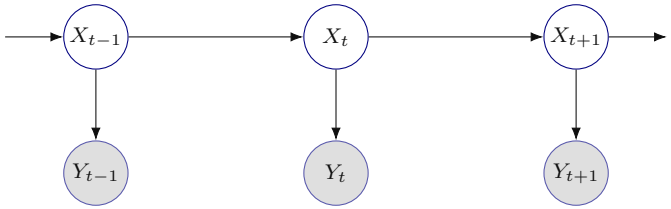


$\mathrm{Var}(Z_t) = 1$; $N = 8192$; first 180 samples shown. Stationary initialisation; no output normalisation.

ARIMA: the level is nonstationary; its first differences have the ARMA dependence shown in the third row.

State-space models

System's state not directly observable: Assume a link between evolving state and observations.



Example: Linear - Gaussian AR system

$$X_t = AX_{t-1} + Z_t, \quad Z_t \sim \mathcal{N}(0, Q),$$
$$Y_t = CX_t + V_t, \quad V_t \sim \mathcal{N}(0, R).$$

Gaussian X_0, independent of the mutually independent white-noise sequences Z_t and V_t.

Kalman filter: $X_t \mid Y_{1:t} = y_{1:t} \sim \mathcal{N}(m_t, P_t)$

$$a_t = Am_{t-1}, \qquad P_t^- = AP_{t-1}A^\top + Q,$$
$$K_t = P_t^- C^\top (CP_t^- C^\top + R)^{-1},$$
$$m_t = a_t + K_t(y_t - Ca_t), \qquad P_t = (I - K_t C)P_t^-.$$

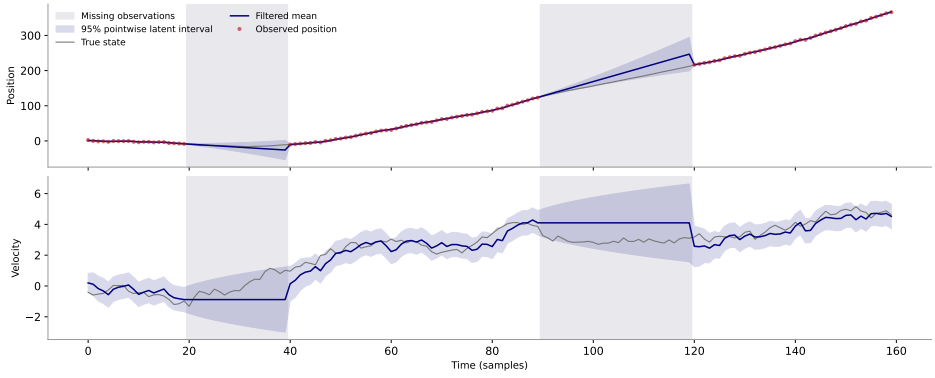
This works even w/out observations.

Optimality: The Kalman filter is optimal in the linear-Gaussian case. Beyond that, the optimal estimatior can be approximated via particle filtering.

Experiment: Tracking a 2D State Space model

Linear-Gaussian constant-velocity model ($\Delta t = 1$): observe noisy positions; infer hidden **position** R_t **and velocity** U_t.

$$\underbrace{\begin{pmatrix} R_t \\ U_t \end{pmatrix}}_{X_t} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} R_{t-1} \\ U_{t-1} \end{pmatrix} + Z_t, \quad Y_t = R_t + V_t, \quad Z_t \sim \mathcal{N}(0, 0.05 I_2), \quad V_t \sim \mathcal{N}(0, 1).$$



Missing data: predict without updating; uncertainty grows. Observations returning reduce uncertainty.

Power spectrum

Wide-sense stationarity (WSS): finite second moments, constant mean $\mathbb{E}[X_t] = \mu$, and covariance $\mathrm{Cov}(X_t, X_{t-k}) = \gamma_X(k)$ depending only on the lag.

Definition (Power spectral density, PSD)

For a WSS process with absolutely summable covariance,

$$S_X(\omega) = \sum_{k \in \mathbb{Z}} \gamma_X(k) e^{-i\omega k}, \qquad \gamma_X(k) = \frac{1}{2\pi} \int_{-\pi}^{\pi} S_X(\omega) e^{i\omega k} \, \mathrm{d}\omega.$$

From one path: assuming ergodicity, recover population moments from time averages. For a zero-mean series,

$$\widehat{\gamma}_N(k) = \frac{1}{N-k} \sum_{t=k+1}^{N} x_t x_{t-k} \longrightarrow \gamma_X(k).$$

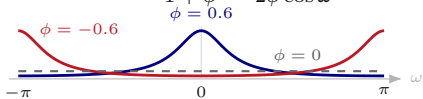
Fourier view of the path: its periodogram is

$$I_N(\omega) = \frac{1}{N} \left| \sum_{t=1}^{N} x_t e^{-i\omega t} \right|^2.$$

$\mathbb{E}[I_N(\omega)] \to S_X(\omega)$ for a zero-mean WSS process.

Example: stationary AR(1), $|\phi| < 1$.

$$S_X(\omega) = \frac{\sigma^2}{1 + \phi^2 - 2\phi \cos \omega}.$$

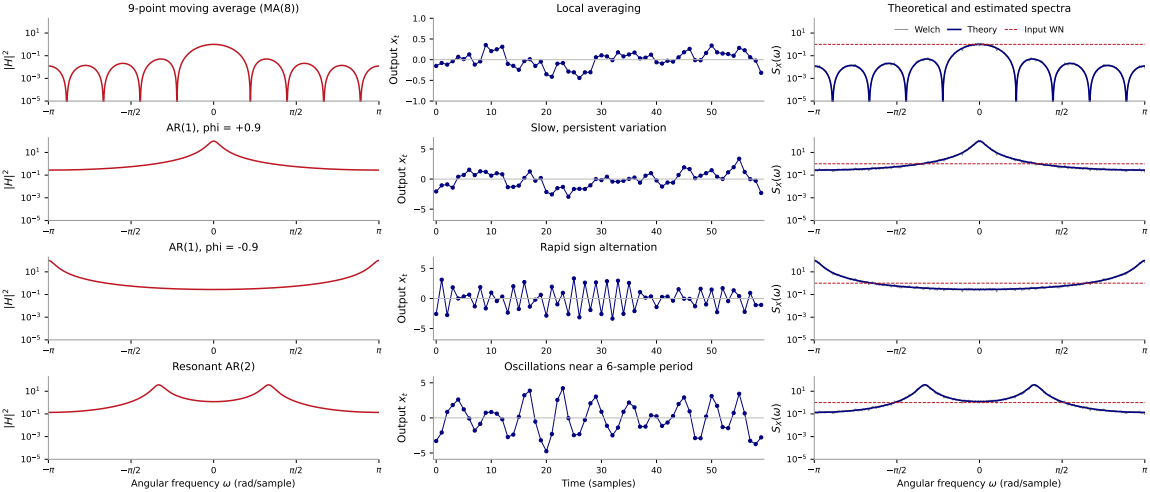


Positive feedback favours low frequencies; negative feedback favours high frequencies.

Idea: White noise has a flat spectrum. How does the filter shape the spectrum of X?

Insight: Filtering sculpts the power spectrum

The filter weights each frequency: $S_X(\omega) = |H(e^{i\omega})|^2 S_Z(\omega)$, with $S_Z(\omega) = 1$.



Same noise input; no output normalisation. Trajectories: 60 samples; Welch spectra: $N = 32768$. Connected dots mark discrete-time samples.

II - Gaussian processes



Discrete time: X_t, $t \in \mathbb{Z}$

Continuous time: $X(t)$, $t \in \mathbb{R}$

Learning outcomes

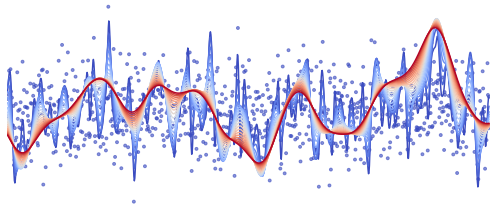
Construct consistent models of continuous-time trajectories.

Design via covariance or spectrum; learn from data.

The extension to continuous time

Motivation: Observations are only partial, in a grid beyond our control.

Desiderata: A model for the underlying continuous-time process, where observations can occur at **any finite set of times**.



Three requirements for dropping the grid

1. Source of noise

A continuous-time analogue of the white-noise source Z_t.

2. Filter restrictions

Replace the weighted sum by a well-defined stochastic integral.

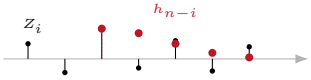
3. Grid consistency

Adding or removing times preserves the joint distribution at retained times.

Continuous-time white noise process

Step 0: A shifted filter at every noise stem.

$$X_n = \sum_{i=1}^{N} Z_i h_{n-i}.$$

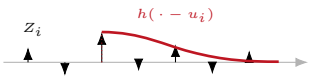


Scaled copies of the filter (i.i.d. weights) are added to obtain X_n.

Step 1: Maintain the impulses, but with a continuous filter.

$$Z_{\text{grid}}(u) = \sum_{i=1}^{N} Z_i \delta(u - u_i),$$

$$X_N(t) = \int h(t-u) Z_{\text{grid}}(u)\,\mathrm{d}u = \sum_{i=1}^{N} Z_i h(t-u_i).$$

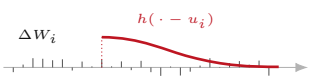


Continuous time, yet only N random coefficients.

Step 2: Refine the grid by scaling the weights accordingly.

$$\Delta W_i = \sqrt{\Delta u_i}\, Z_i, \qquad Z_i \overset{\text{iid}}{\sim} \mathcal{N}(0,1),$$

$$\sum_i h(t-u_i)\Delta W_i \;\longrightarrow\; X(t) = \int h(t-u)\,\mathrm{d}W(u).$$



Brownian motion W has continuous, non-differentiable, paths, $W(0) = 0$, and independent increments $\Delta W_i \sim \mathcal{N}(0, \Delta u_i)$.

The Gaussian process

Any finite collection of evaluations of the continuous-time convolution process defined above is **jointly Gaussian**.

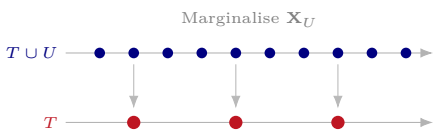
Definition (Gaussian process)

$X \sim \mathcal{GP}(m, k)$ if, for every finite set of times $T = \{t_1, \dots, t_n\}$,

$$\mathbf{X}_T = \big(X(t_1), \dots, X(t_n)\big)^\top \sim \mathcal{N}(\mathbf{m}_T, K_T),$$

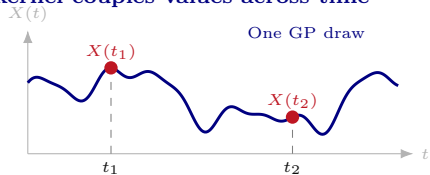
where $(\mathbf{m}_T)_i = m(t_i)$ and $(K_T)_{ij} = k(t_i, t_j)$.

Consistent across grids



Removing times preserves the joint law at retained times. The process is defined independently of the observation grid.

The kernel couples values across time



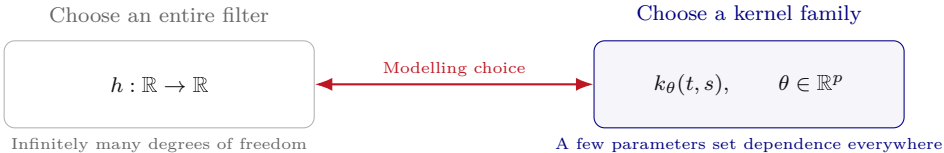
$k(t_1, t_2) = \mathrm{Cov}\big(X(t_1), X(t_2)\big)$. Across draws, the kernel controls how values at two times vary together.

Where is the kernel coming from?

The filter determines the covariance of $X(t) = \int h(t-u)\,\mathrm{d}W(u)$:

$$k(t,s) = \int_{\mathbb{R}} h(t-u)h(s-u)\,\mathrm{d}u, \qquad k(t,t) = \|h\|_2^2 < \infty \quad \text{for } h \in L^2.$$

Choose a positive-semidefinite kernel k_θ directly.



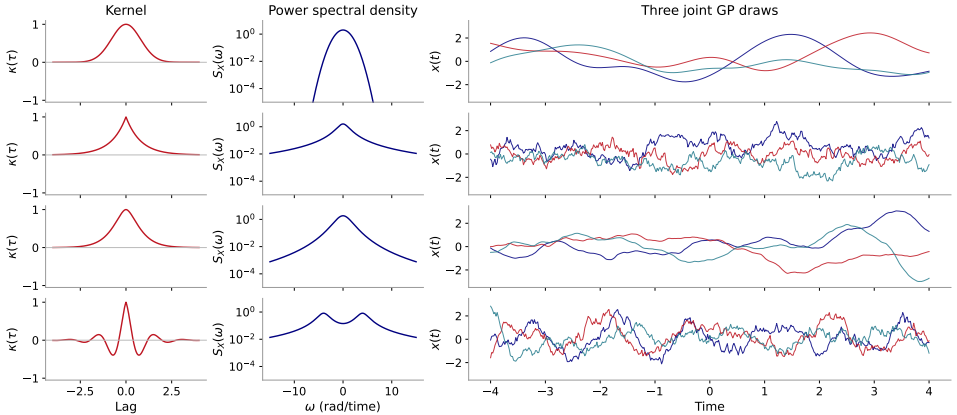
Strong inductive bias

The chosen (parametrised) kernel family specifies features such as correlation, regularity, and oscillations across the series. Parameters are learnt from observations.

Moore–Aronszajn gives a feature factorisation $k(t,s) = \langle \Phi(t), \Phi(s) \rangle_{\mathcal{H}}$ for every positive-semidefinite kernel; this need not be a convolution representation.

A library of kernels

For stationary GPs, $k(t,s) = \kappa(\tau)$, with $\tau = t - s$, $r = |\tau|/\ell$, and $\kappa(\tau) \overset{\mathcal{F}}{\longleftrightarrow} S_X(\omega)$.



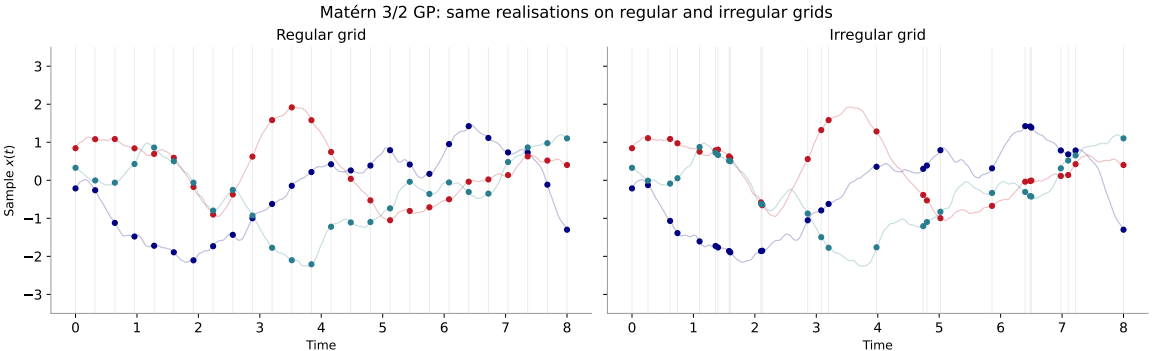
σ^2: variance; ℓ: length scale; ω_0: oscillation frequency. Here $\sigma^2 = 1$.

Build new kernels: sums and pointwise products preserve validity; nonnegative weights control their contributions.

Prior sampling: choose a grid, calculate covariance matrix, and draw

$$T = (t_1, \ldots, t_n) \longrightarrow K_{ij} = k(t_i, t_j) \longrightarrow K_T = LL^\top \longrightarrow \mathbf{X}_T = \mathbf{m}_T + L\mathbf{U}$$

$$\mathbf{U} \sim \mathcal{N}(0, I_n) \quad \Longrightarrow \quad \mathbf{X}_T \sim \mathcal{N}(\mathbf{m}_T, K_T).$$



Sample jointly: in both cases, values were selected from the same three realisations. Shared times = identical values.

Conditioning is a Bayesian update

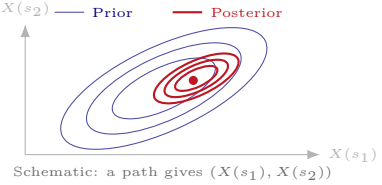
Observations: $\mathbf{t} = (t_1, \dots, t_n)^\top$, $\mathbf{y} = (y_1, \dots, y_n)^\top$, with $y_i = X(t_i) + v_i$, $v_i \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma_n^2)$.

Fix the hyperparameters. Write $\mathbf{m} = (m(t_i))_{i=1}^n$, $\mathbf{k}_t = (k(t, t_i))_{i=1}^n$, and $C_{ij} = k(t_i, t_j) + \sigma_n^2 \delta_{ij}$.

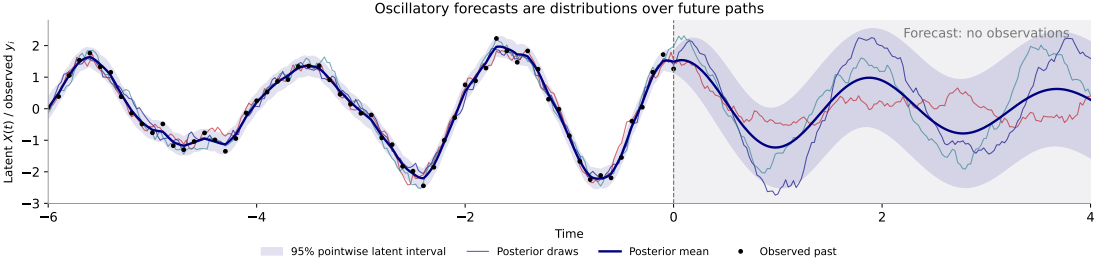
$$m_{\text{post}}(t) = m(t) + \mathbf{k}_t^\top C^{-1}(\mathbf{y} - \mathbf{m}),$$

$$k_{\text{post}}(t, s) = k(t, s) - \mathbf{k}_t^\top C^{-1} \mathbf{k}_s.$$

The nonparametric aspect: the posterior mean has one kernel term per observation; its representation grows with the data.



Example: damped-oscillatory kernel $k(t, s) = e^{-|t-s|/4} \cos\Big(2\pi(t-s)/1.8\Big)$, $m(t) = 0$, $\sigma_n^2 = 0.04$.



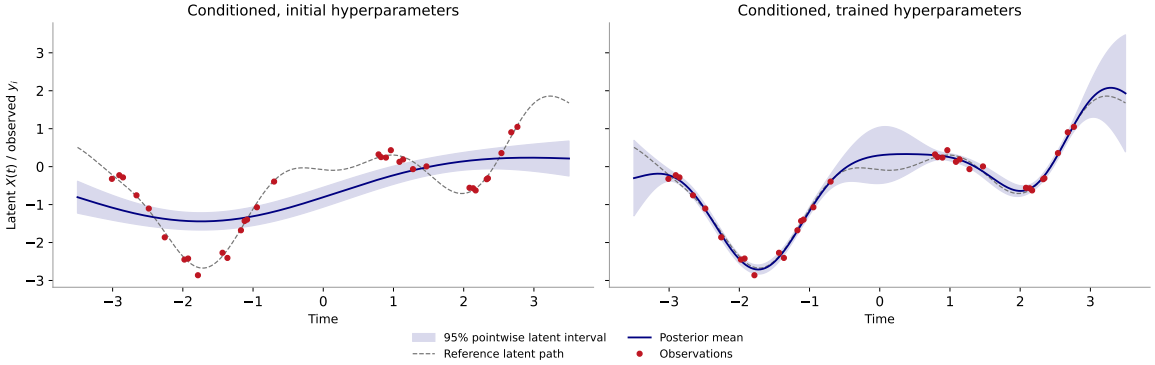
Thin coloured curves: **joint posterior draws**; thick curve: posterior mean. Band: 95% pointwise latent intervals.

Training: Choose the kernel (expert) and its parameters (optimisation)

Exact likelihood: $Y \sim \mathcal{N}(m_T, C_\theta)$, $C_\theta = K_\theta(T,T) + \sigma_n^2 I$.

$$\mathcal{L}(\theta) = \underbrace{\tfrac{1}{2} r^\top C_\theta^{-1} r}_{\text{data fit}} + \underbrace{\tfrac{1}{2} \log |C_\theta|}_{\text{covariance volume}} + \tfrac{n}{2} \log(2\pi), \qquad r = y - m_T.$$

Kernel inductive bias ⇒ only a few parameters specify an infinite-dimensional model (learning is unfeasible otherwise).



Both models condition on the same data. Training changes only the hyperparameters; bands omit parameter uncertainty.

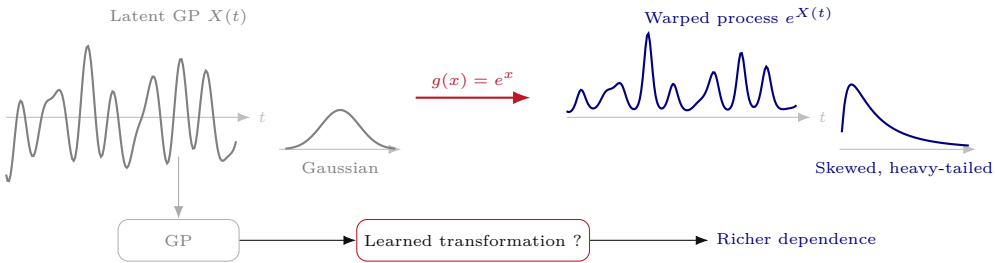
Gaussian processes: a structured base for richer models

Pros

- Consistent grids and interpretable covariance.
- Exact conditioning/training with Gaussian noise.

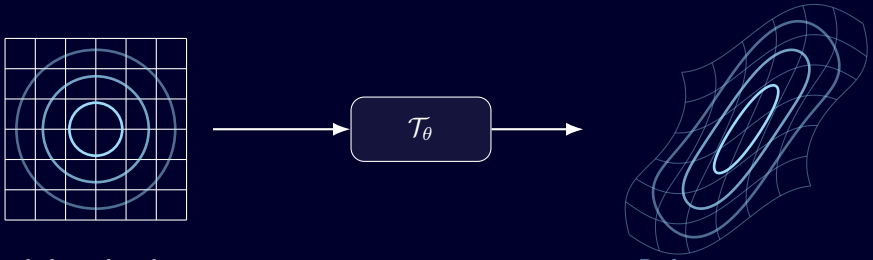
Cons

- Manual kernel choice, and "only Gaussian".
- Computations grow cubically with data.



Next: learn transformations for GPs with different levels of complexity.

III - Flow models



Simple base distribution

Learn the transformation

Richer joint structure

Learning outcomes

Generate time series through learned nonlinear transport.

Use representations and physics to shape signal structure.

GPs as latent variables: non-Gaussian likelihoods

Exact inference above assumed a **Gaussian likelihood**, but **we can change it** while keeping conditional independence.

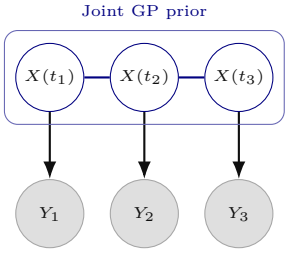
$$X \sim \mathcal{GP}(m, k), \qquad p(\mathbf{y} \mid \mathbf{X}_T) = \prod_i p(y_i \mid X(t_i)).$$

Regression: $Y_i \mid X(t_i) \sim \mathcal{N}(X(t_i), \sigma_n^2)$,

Classification: $Y_i \mid X(t_i) \sim \text{Bernoulli}(s(X(t_i)))$,
$s(a) = (1 + e^{-a})^{-1}$,

Counts: $Y_i \mid X(t_i) \sim \text{Poisson}\left(X(t_i)^2\right)$,

Positive values: $Y_i \mid X(t_i) \sim \text{Gamma}\left(\alpha, \alpha e^{-X(t_i)}\right)$.



GP temporal dependence **extends beyond real-valued signals.** Non-Gaussian likelihoods generally require **approximate posterior inference and marginal-likelihood training**:

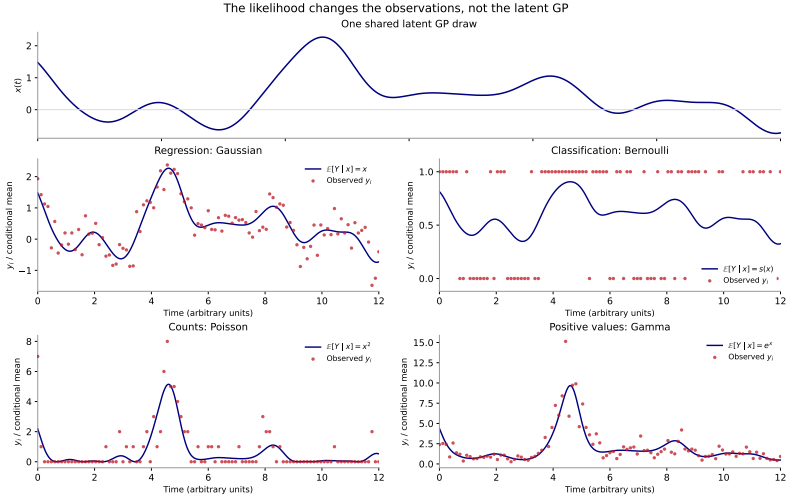
$$p(\mathbf{x}_T \mid \mathbf{y}) = \frac{p(\mathbf{y} \mid \mathbf{x}_T) p(\mathbf{x}_T)}{p(\mathbf{y})}, \qquad \underbrace{p(\mathbf{y}) = \int p(\mathbf{y} \mid \mathbf{u}) p(\mathbf{u}) \, \mathrm{d}\mathbf{u}}_{\text{generally intractable}}.$$

The integral normalises the posterior and defines the training objective. Use Laplace, variational, or Monte Carlo methods.

Rasmussen and Williams (2006), GPML, Chapter 3.

Example: Latent GP with different likelihoods

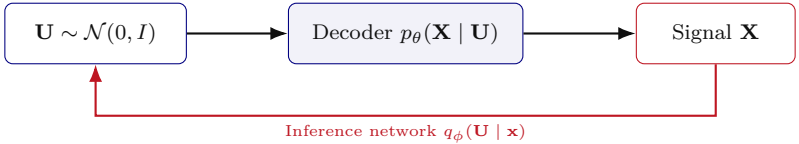
The same latent path drives all four cases; the likelihood determines the observation range and variability.



Blue: conditional mean; red: sampled observations. Smooth latent dynamics can produce noisy, binary, count or positive observations.

A more expressive transformation: The VAE route

[Back to finite time] Replace the linear filter by a **nonlinear decoder** receiving a Gaussian vector.



$$\log p_\theta(\mathbf{x}) \;\geq\; \underbrace{\mathbb{E}_{q_\phi(\mathbf{U}|\mathbf{x})}[\log p_\theta(\mathbf{x} \mid \mathbf{U})]}_{\text{reconstruction}} - \underbrace{\mathrm{KL}\Big(q_\phi(\mathbf{U} \mid \mathbf{x}) \| p(\mathbf{U})\Big)}_{\text{latent regularisation}} = \mathrm{ELBO}.$$

Gaussian example: $p(\mathbf{U}) = \mathcal{N}(0, I)$, $q_\phi(\mathbf{U} \mid \mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}_\phi, \mathrm{diag}(\boldsymbol{\sigma}_\phi^2))$, $p_\theta(\mathbf{x} \mid \mathbf{U}) = \mathcal{N}(f_\theta(\mathbf{U}), \sigma_{\text{obs}}^2 I)$. Here f_θ is a NN, σ_{obs}^2 is fixed, and $\boldsymbol{\mu}_\phi, \boldsymbol{\sigma}_\phi$ are encoder outputs at $\mathbf{x}$.

$$-\mathrm{ELBO} = \underbrace{\frac{1}{2\sigma_{\text{obs}}^2} \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(0,I)} \left\| \mathbf{x} - f_\theta(\boldsymbol{\mu}_\phi + \boldsymbol{\sigma}_\phi \odot \boldsymbol{\epsilon}) \right\|^2}_{\text{reconstruction error}} + \underbrace{\frac{1}{2} \sum_{j=1}^{d} (\mu_{\phi,j}^2 + \sigma_{\phi,j}^2 - 1 - \log \sigma_{\phi,j}^2)}_{\text{analytic Gaussian KL}} + \frac{n}{2} \log(2\pi\sigma_{\text{obs}}^2).$$

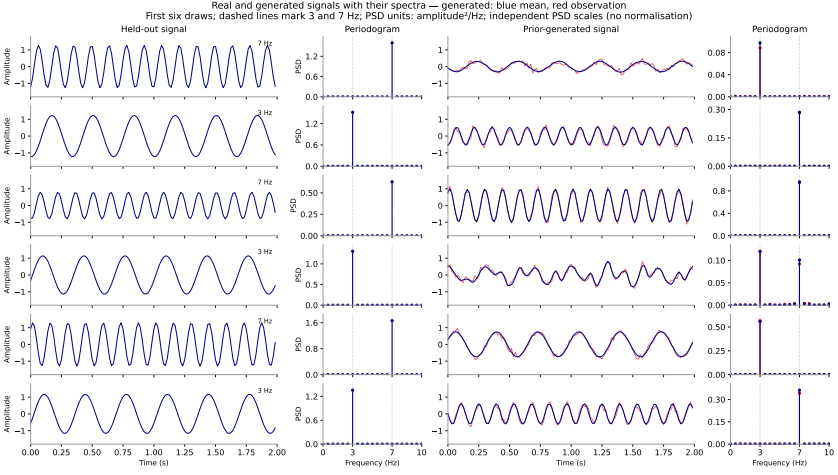
d: code dimension; n: signal dimension. Estimate the expectation using sampled codes.

Reconstruction does not enforce physics or temporal dynamics. If needed, they should be enforced via architectural or optimisation choices.

Kingma and Welling (2013), Auto-Encoding Variational Bayes.

VAE experiment: does generation preserve frequency?

Train on 3,000 sinusoids at 3 or 7 Hz, with random amplitude and phase (128 samples, 2 seconds). Convolutional VAE: 3D code, residual encoder and decoder.

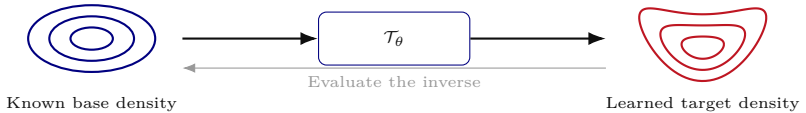


Held-out data versus fresh prior samples; spectra use independent vertical scales. Most generated signals recover a single frequency; amplitudes and occasional mode mixing remain imperfect.

Normalising flows: diffeomorphic maps ensure exact likelihood

It seemed like the VAE was too ambitious. Instead, we consider a differentiable bijection between objects of the **same dimension**:

$$\mathbf{X} = \mathcal{T}_\theta(\mathbf{X}_0), \qquad \mathbf{X}_0 \sim p_0, \qquad \det D\mathcal{T}_\theta \neq 0.$$



$$\boxed{p_\theta(\mathbf{x}) = p_0\left(\mathcal{T}_\theta^{-1}(\mathbf{x})\right)\left|\det D\mathcal{T}_\theta^{-1}(\mathbf{x})\right|}$$

For $\mathcal{T} = \mathcal{T}_L \circ \cdots \circ \mathcal{T}_1$, log determinants add:

$$\log p_\theta(\mathbf{x}_L) = \log p_0(\mathbf{x}_0) - \sum_{\ell=1}^{L} \log\left|\det D\mathcal{T}_\ell(\mathbf{x}_{\ell-1})\right|.$$

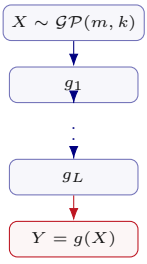
If the map's inverse and determinant are tractable: **train by exact maximum likelihood** (no integrals over latents).

Papamakarios et al. (2021), Normalizing Flows for Probabilistic Modeling and Inference.

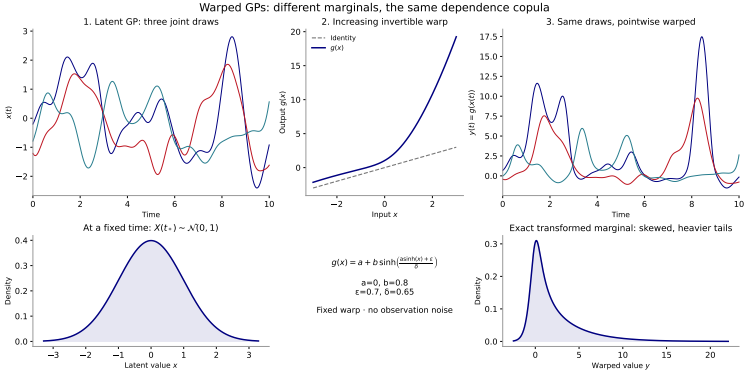
Example: Compositionally-warped GPs

The NF concept can be applied to GPs using **coordinate-wise transformations**. Consider $X \sim \mathcal{GP}(m, k)$, $Y(t) = g(X(t))$, with invertible pointwise $g = g_L \circ \cdots \circ g_1$. On a grid with nonsingular K_T:

$$p_Y(\mathbf{y}) = \mathcal{N}\left(g^{-1}(\mathbf{y}); \mathbf{m}_T, K_T\right) \prod_{i=1}^{n} \left|(g^{-1})'(y_i)\right| .$$



Same scalar maps at every time t



Non-Gaussian marginals, tractable density. Increasing warps preserve the Gaussian copula. This noiseless example uses one sinh–arcsinh warp; observation noise generally reintroduces an integral.

Rios and Tobar (2019), Compositionally-Warped Gaussian Processes.

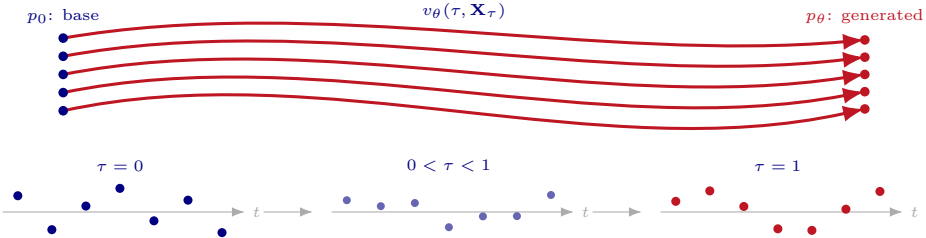
Continuous flows transport entire time series

On a fixed grid, a signal is a vector $\mathbf{X} = (X(t_1), \dots, X(t_n))^\top \in \mathbb{R}^n$.



Each layer maps $\mathbf{X}_k \mapsto \mathbf{X}_{k+1}$. Writing $\mathbf{X}_{k+1} - \mathbf{X}_k = \Delta\tau\, v_\theta(\tau_k, \mathbf{X}_k)$, the limit $\Delta\tau \to 0$ gives a **neural ODE**:

$$\frac{\mathrm{d}\mathbf{X}_\tau}{\mathrm{d}\tau} = v_\theta(\tau, \mathbf{X}_\tau), \qquad \mathbf{X}_0 \sim p_0, \qquad \mathbf{X}_1 = \mathcal{T}_\theta(\mathbf{X}_0).$$



Two clocks: t indexes samples; τ transports the signal, not its physical dynamics.

One-to-one flow: v_θ Lipschitz in the state, with solutions throughout $[0, 1]$; the network itself need not be invertible.

Chen et al. (2018), Neural Ordinary Differential Equations.

Conditional flow matching: learn the velocity

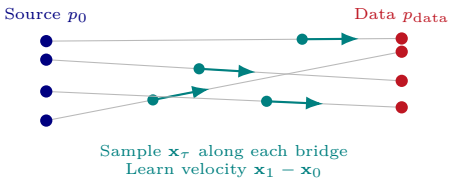
Draw independent endpoints $\mathbf{X}_0 \sim p_0$ and $\mathbf{X}_1 \sim p_{\text{data}}$, and $\tau \sim \mathcal{U}(0,1)$. The endpoint-conditioned linear bridge has a known velocity:

$$\mathbf{X}_\tau = (1-\tau)\mathbf{X}_0 + \tau \mathbf{X}_1, \qquad \dot{\mathbf{X}}_\tau = \mathbf{X}_1 - \mathbf{X}_0.$$

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}\left[\left\| v_\theta(\tau, \mathbf{X}_\tau) - (\mathbf{X}_1 - \mathbf{X}_0) \right\|^2 \right].$$

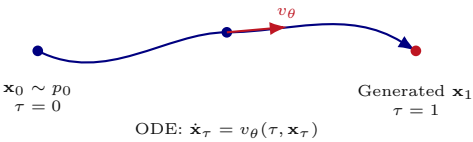
At the population optimum, $v^*(\tau, \mathbf{x}) = \mathbb{E}[\mathbf{X}_1 - \mathbf{X}_0 \mid \mathbf{X}_\tau = \mathbf{x}]$: averaging the bridge velocities transports their marginal distributions.

Train: supervised velocity regression



Points represent whole signals; arrows are regression targets. No ODE solve is needed during training.

Sample: integrate the learned field



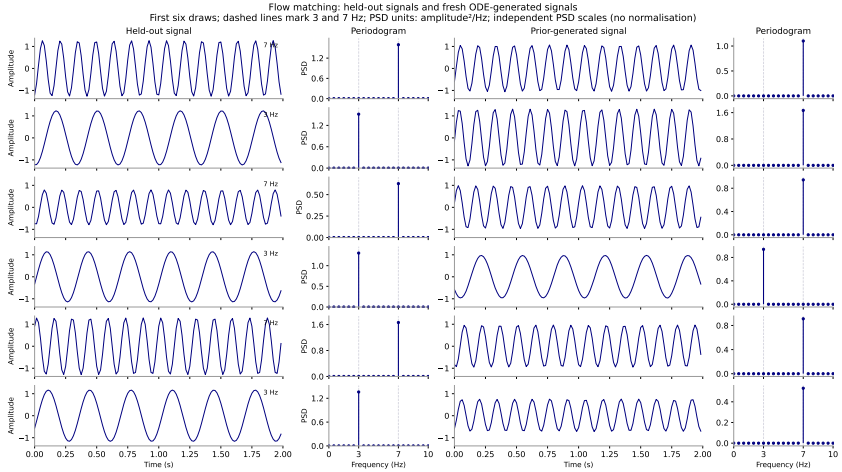
Start from fresh noise; integrate from 0 to 1. No data endpoint is supplied at generation time.

Here "conditional" refers to the training bridges, not to observed data. The transport statement assumes a suitably well-defined marginal flow.

Lipman et al. (2023), Flow Matching; Tong et al. (2024), generalised CFM.

Flow matching: generating the same two-frequency signals

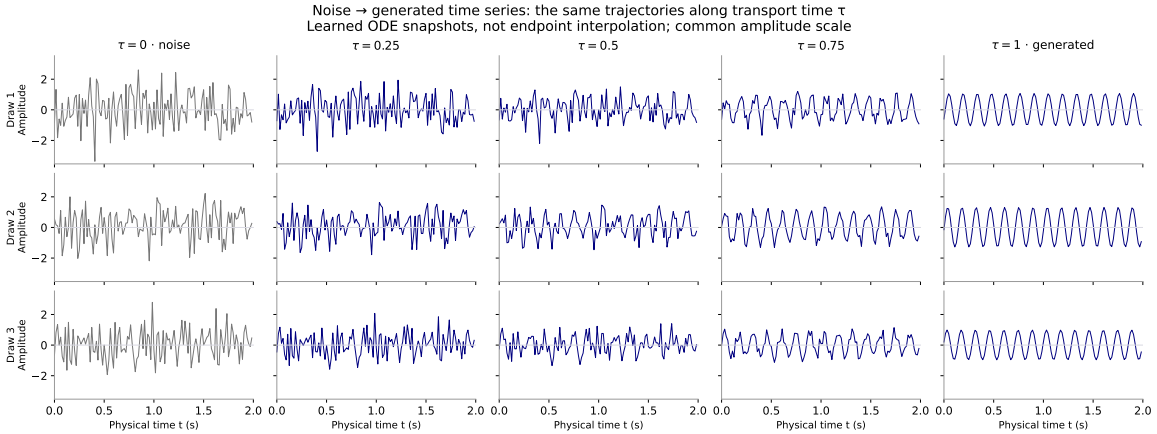
Same data as slide 27; a time-conditioned residual 1D U-Net learns the velocity. Generate from fresh Gaussian noise by solving the ODE.



99.6% of generated energy lies at 3 or 7 Hz. Frequency balance and amplitudes remain imperfect; spectra use independent vertical scales.

Flow matching: from noise to an oscillatory signal

Each row follows the same noise draw through the learned ODE; no target endpoint is supplied.



Columns advance transport time τ; each horizontal axis is physical time t. Actual solver snapshots, with a common amplitude scale and no smoothing or frequency projection.

Challenge: which properties do we want to preserve, or *align*?

A scientifically useful generator should reproduce relevant properties for the application, e.g., spectra, temporal dependence, constraints, or physical laws.

By construction
Support constraints, symmetries, structured decoders

Guarantees only what the construction enforces

In the objective
Spectral discrepancies, physical residuals

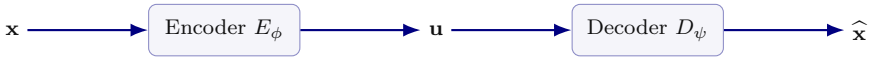
Soft penalties encourage but do not guarantee

The **alignment question** also appears in language models: does the learned system behave as intended? **In scientific AI**, physical and dynamical features require their own definitions and tests.

Latent flow matching: compress, transport, decode

Represent each signal $\mathbf{x} \in \mathbb{R}^{nd}$ by a smaller code $\mathbf{u} = E_\phi(\mathbf{x}) \in \mathbb{R}^r$, with $r \ll nd$.

1. Learn a representation: encode and reconstruct



2. Generate: transport noise to encoded data, then decode



Train the flow on codes $E_\phi(\mathbf{x})$ from the autoencoder.

Lower computational cost

Learn and integrate in r dimensions, rather than nd.

Control what the code preserves

Align the autoencoder with signal features, such as spectral structure, before training the flow.

Example: Spectrally aligned latent flow matching

Problem: latent compression can introduce spectral artefacts. **Idea:** align the autoencoder with signal features *before* training the flow.

1. Train autoencoder
Reconstruction + adversarial loss
→ 2. Align latent space
25 extra epochs
Transform-based loss
→ 3. Train latent FM
Gaussian noise → $E_\phi(\mathbf{x})$
→ 4. Generate
10-step Euler ODE + decode

Allocate reconstruction capacity to detail.

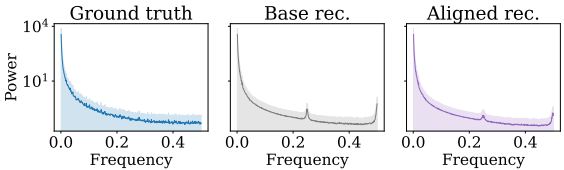
With $\widehat{\mathbf{x}} = D_\psi(E_\phi(\mathbf{x}))$, add

$$\mathcal{L}_T = \sum_k w_k^2 \left\| T_k(\mathbf{x}) - T_k(\widehat{\mathbf{x}}) \right\|^2.$$

Fourier: frequency bands. **Wavelets:** local multiscale detail. **Signatures:** higher-order path features.

Sobolev-inspired weights emphasise higher frequencies/detail; retain reconstruction and adversarial losses during alignment.

Evidence: reduced spectral artefacts.



Paper Fig. 1: Exchange-rate *reconstructions*; power spectra, mean ±1 SD (log scale). Alignment attenuates the spurious peak near 0.25.

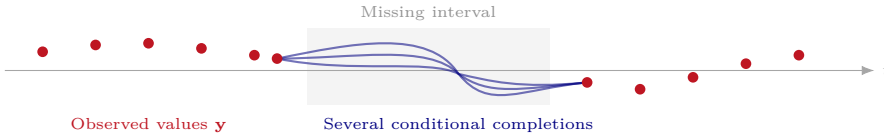
Generation results: HEPC discriminative score (lower is better): base 0.211 ± 0.103 → Fourier-aligned 0.018 ± 0.012 (Table 1). Tests use 1,000 generated and 1,000 held-out length-1,000 signals; reported generation is < 1 s per 1,000 samples on an RTX3090.

Takeaway: shape the representation to retain signal structure, then learn transport; empirical alignment is not an exact per-sample guarantee.

Carvajal Reyes and Tobar (2026), *Time series generation with spectrally aligned latent flow matching.* Fig. 1, Table 1 and Secs. 3–6.

Challenge: Conditional generation is intractable

Condition a full signal $\mathbf{x}$ on observations $\mathbf{y}$, with times, channels and mask supplied.



With an observation model $\mathbf{y} = H\mathbf{x} + \boldsymbol{\varepsilon}$:

$$p_\theta(\mathbf{x} \mid \mathbf{y}) = \frac{p_\theta(\mathbf{x}, \mathbf{y})}{p_\theta(\mathbf{y})} \propto \underbrace{p(\mathbf{y} \mid \mathbf{x})}_{\text{observation likelihood}} \underbrace{p_\theta(\mathbf{x})}_{\text{learned FM prior}} .$$

Two routes: learn a conditional sampler, or infer using the fixed prior.

Train with observed context.

Learn $v_\phi(\tau, \mathbf{x}_\tau, \mathbf{y})$ to approximate $p_\theta(\mathbf{x} \mid \mathbf{y})$. Construct training pairs by masking/splitting series (and adding measurement noise when appropriate). Generate with $\mathbf{y}$ fixed.

Prior-proposal MCMC: no retraining.

Draw a proposal $\mathbf{x}' \sim p_\theta$ using the trained FM model; accept with probability

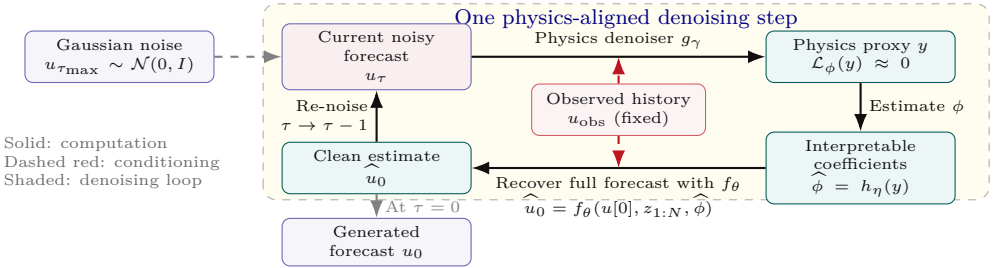
$$\min\left(1, \frac{p(\mathbf{y} \mid \mathbf{x}')}{p(\mathbf{y} \mid \mathbf{x})}\right).$$

Only likelihood evaluations are needed: the prior densities cancel.

Requires a regular observation likelihood; informative data can make prior proposals inefficient. Exact noiseless constraints need specialised inference.

Example: physics-aligned interpretable diffusion (PAID)

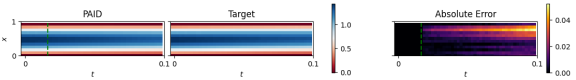
Aim: sample forecasts of $\mathcal{F}_\vartheta(u) + \mathcal{L}_\phi(u) = 0$: unknown $\mathcal{F}_\vartheta$, known $\mathcal{L}_\phi$ with coefficients ϕ to infer.



Example: diffusion–advection.

$$\underbrace{u_t - \beta u_{xx}}_{\text{known diffusion form}} + \underbrace{\alpha u_x}_{\text{unmodelled advection}} = 0.$$

Infer β; learn corrections for missing dynamics.
Standard convention: $\beta > 0$.



One example from draft Fig. 2: PAID forecast, target and absolute error. Dashed green: end of the observed history.

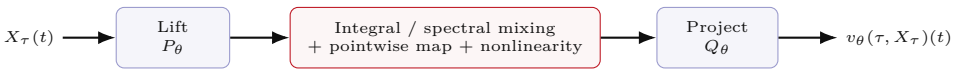
Skye Purchase and Felipe Tobar, *Physics-Aligned Interpretable Diffusion*, in preparation. Adapted from draft Figs. 1–2; PDE written in the standard sign convention.

Future directions: flow matching over functions

Beyond today's scope: learn transport of functions, rather than fixed-grid vectors. Functional flow matching extends the same velocity-regression idea:

$$X_\tau = (1-\tau)X_0 + \tau X_1, \qquad \mathcal{L}(\theta) = \mathbb{E}\left\| v_\theta(\tau, X_\tau) - (X_1 - X_0) \right\|_{\mathcal{H}}^2.$$

A **neural operator** represents the functional velocity $v_\theta(\tau, \cdot) : \mathcal{H} \to \mathcal{H}$:



Repeat mixing layers; include t and τ as inputs.

$$\partial_\tau X_\tau = v_\theta(\tau, X_\tau), \qquad X_0 \sim \text{a suitable GP base.}$$

Goal: generate an underlying signal and evaluate it on different grids, including new query times and resolutions.

We leave the functional analysis, operator architectures and numerical details for future work. Shared parameters alone do not guarantee consistency across grids.

Kerrigan, Migliorini and Smyth (2024), Functional Flow Matching. Kovachki et al. (2023), Neural Operator.

What transformations buy us, and what they do not

Construction	Random object	Learning / inference	Structure and its boundary
GP + likelihood	Latent function; observations	Marginal likelihood; often approximate posterior	Consistent GP prior; observation likelihood changes marginals, not latent Gaussianity.
VAE	Latent code and decoded signal	Variational bound	Flexible decoder; reconstruction does not ensure dynamics or cross-grid consistency.
Finite flow	Same-dimensional vectors	Change-of-variables likelihood	Invertible map and tractable density by design; no automatic scientific invariants.
CNF / CFM	Vector transport	Density evolution / velocity regression	ODE flow under regularity assumptions; physical time and transport time differ.
Latent flow	Code, then decoded signal	Autoencoding + latent transport	Decoder can impose constraints; compression can also discard relevant structure.
Functional flow	Random functions	Functional velocity regression	Function-space formulation; regularity and numerical convergence require care.

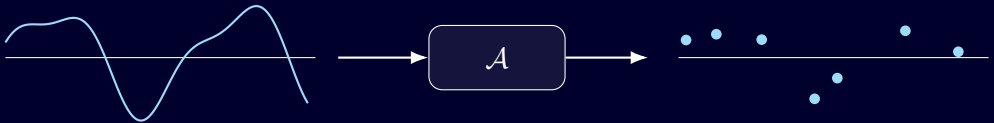
What we didn't see

Multioutput models
Joint channels and related series: cross-correlations and shared latent structure.

Distances among time series
Alignment, spectral features, or path geometry determine which differences matter.

Foundation models
Pretrain across datasets; adapt to new domains, contexts, and tasks.

IV - Examples: inverse problems



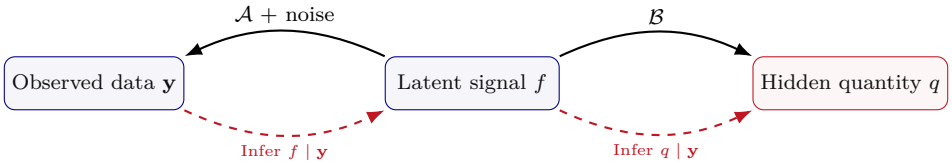
Learning outcomes

Solve inverse problems by conditioning a generative signal model.

Generative models: performing inversion via Bayesian updates

The forward operator $\mathcal{A}$ can filter, blur, mix or integrate the signal, then sample or mask it. We are interested in inverting the effect of $\mathcal{A}$ to determine the hidden signal or even a transformation of it:

$$f \sim \mathcal{GP}(0, k), \qquad \underbrace{\mathbf{y} = \mathcal{A}f + \boldsymbol{\varepsilon}}_{\text{observations}}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(0, R), \qquad \underbrace{q = \mathcal{B}f}_{\text{quantity to recover}} \quad .$$



Our focus: Linear operators $\mathcal{A}$, which preserve joint Gaussianity. Cross-covariances determine the posterior:

$$q \mid \mathbf{y} \sim \mathcal{N}\left(K_{qy}C_y^{-1}\mathbf{y},\ K_{qq} - K_{qy}C_y^{-1}K_{yq}\right), \qquad C_y = K_{\mathcal{A}f,\mathcal{A}f} + R.$$

Task	Measurements $\mathcal{A}f$	Target $\mathcal{B}f$
Spectral estimation	Point samples	Windowed Fourier coefficients
Band-pass filtering	Point samples	Selected frequency-band component
Deconvolution	Samples of a convolution	Original source signal

Noise is independent of f; q denotes finite real coordinates (stack real/imaginary Fourier parts). Conditioning is analytic for fixed parameters and evaluable transformed covariances.

Application 1: Bayesian nonparametric spectral estimation

Recover a spectrum from incomplete, noisy time samples. Use a stationary GP for the signal, then infer its Fourier representation:

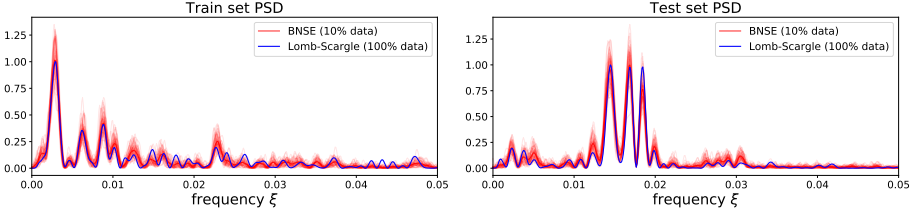
$$f \sim \mathcal{GP}(0, k), \qquad y_i = f(t_i) + \varepsilon_i, \qquad F_\alpha(\xi) = \int_{\mathbb{R}} e^{-\alpha t^2} f(t) e^{-2\pi i \xi t}\, \mathrm{d}t, \quad \alpha > 0.$$

The transform is linear.
Observations and the real/imaginary Fourier coefficients are jointly Gaussian. Condition to obtain their posterior means, covariances and draws.

Propagate uncertainty to spectral power.
For $S_\alpha(\xi) = |F_\alpha(\xi)|^2$,

$$\mathbb{E}[S_\alpha(\xi) \mid \mathbf{y}] = |m_F(\xi)|^2 + v_F(\xi).$$

m_F: posterior mean; v_F: posterior complex variance.



Paper Fig. 5: heart-rate spectra. BNSE uses 10% of the irregular, noisy data; the Lomb–Scargle reference uses the full noiseless series. The shaded posterior reflects uncertainty from limited observations.

Felipe Tobar (2018), *Bayesian Nonparametric Spectral Estimation*, NeurIPS. Gaussian window ensures a well-defined transform; analytic covariances for spectral-mixture kernels.

Application 2: Band-pass filtering with the sinc kernel

Recover a selected frequency component from noisy, irregular samples. Split the signal into independent GP components inside and outside a band B:

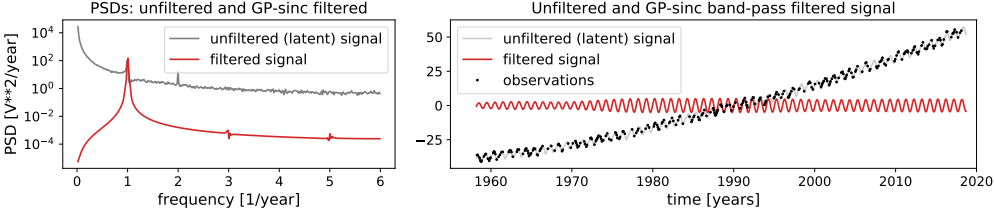
$$f = f_B + f_{\bar{B}}, \qquad y_i = f(t_i) + \varepsilon_i, \qquad k_B = \mathcal{F}^{-1}\left[S_f(\xi)\,\mathbf{1}_B(\xi)\right].$$

Design the prior in frequency.
A pair of rectangular spectral bands gives

$$k_{\text{sinc}}(\ell) = \sigma^2 \,\text{sinc}(\Delta\ell)\cos(2\pi\xi_0\ell).$$

Bandwidth Δ, centre ξ_0; $\text{sinc}(z) = \sin(\pi z)/(\pi z)$.

Filtering becomes posterior inference.
$\text{Cov}(f_B, f) = k_B$ supplies the cross-covariance for $p(f_B \mid \mathbf{y})$. The output is a distribution over continuous-time band-limited signals.

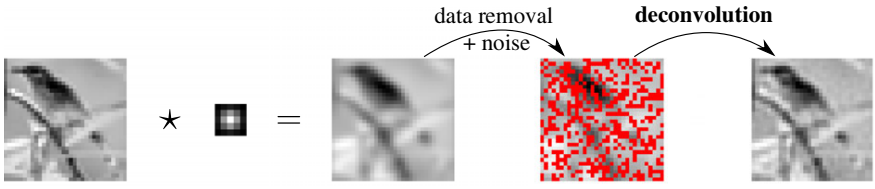


Paper Fig. 7: Mauna Loa CO_2, annual component recovered from 200 of 727 observations; $\xi_0 = 1$ year^{-1}, $\Delta = 0.1$ year^{-1}.
Felipe Tobar (2019), *Band-Limited Gaussian Processes: The Sinc Kernel*, NeurIPS. Gaussian conditioning is analytic; a general band-restricted kernel may need numerical evaluation.

Application 3: Gaussian process deconvolution

Recover a latent source through blur, noise and missing observations.

$$x \sim \mathcal{GP}(0, k_x), \qquad y_i = (h * x)(t_i) + \varepsilon_i = \int h(t_i - s)x(s)\,\mathrm{d}s + \varepsilon_i.$$



Paper Fig. 1: image illustration; the same construction applies to time series.

Apply the forward operator to the covariance.

$$k_{xy}(t, t_i) = \int h(t_i - s)k_x(t, s)\,\mathrm{d}s.$$

Condition the joint GP to recover $x \mid \mathbf{y}$, including uncertainty.

The filter limits what is recoverable.

Frequencies eliminated by h cannot be recovered just by adding observations. Known filters give Gaussian conditioning; unknown filters require learning and can be unidentifiable.

Tobar, Robert and Silva (2023), *Gaussian process deconvolution*, Proc. R. Soc. A 479:20220648.

The end

Generative Modelling for Time Series

AI use statement

OpenAI Codex was used to assist with LaTeX formatting, particularly TikZ figures; Python experiments; and translation of some sections from English into Spanish.