CS 6501: Constrained-Aware Generative AI

Lecture Notes 1
Course Overview and Taxonomy of Constraints

From plausibility to validity in generative modeling

Prof. Ferdinando Fioretto
Department of Computer Science, University of Virginia

Tuesday, January 13, 2026

Abstract

These notes introduce the central theme of the course: transforming deep generative models from systems that optimize plausibility to systems that can also deliver validity, safety, and controllability. We will use a unified probabilistic objective: sampling (or optimizing) with respect to a target distribution obtained by combining a base model with *soft constraint* potentials and *hard* feasibility sets. We then develop the theoretical pillars that make this unification meaningful reviewing (i) Energy-Based Models and the partition function problem, including what breaks when normalizing constants are intractable and which tools remain available; (ii) score matching and the role of score estimation in diffusion models; (iii) convex analysis primitives (projection and proximal operators) that enable hard constraint enforcement and differentiable optimization layers. In this introductory lecture these concepts will be presented at a high level, with an emphasis on the overarching abstractions and the taxonomy of constraints that will recur throughout the course.

1 Introduction

Deep generative models have become exceptionally good at producing samples that resemble real data. In many classical uses of generative modeling, this resemblance is the central objective: *a generated object should be statistically plausible under the training distribution.* For images, plausibility is often aligned with perceptual realism; for text, plausibility is aligned with fluency and semantic coherence; for statistical data, plausibility is aligned with matching observed statistics. This success is anchored in a common methodological thread: learn a flexible parametric distribution and sample from it, using architectures and training objectives that scale [Kingma and Welling, 2014, Goodfellow et al., 2014, Ho et al., 2020, Brown et al., 2020].

In scientific and engineering workflows, however, plausibility is necessary but rarely sufficient. The central difficulty is that downstream use typically demands *validity*: the output must satisfy external constraints that are not fully captured by the data distribution alone. A bridge design that looks like designs in the dataset may still violate stress limits; a molecule that resembles known compounds may still be chemically invalid or non-synthesizable; a robot plan that resembles expert demonstrations may still collide with obstacles or violate dynamics. Empirically, these validity requirements can carve out tiny feasible regions inside a high-dimensional space of plausible objects, making naive sampling ineffective and making post hoc filtering extremely expensive.

To separate these notions precisely, we will treat plausibility and validity as distinct criteria. Plausibility asks whether an output "belongs" to the learned data manifold. Validity asks whether the output satisfies explicit requirements coming from physics, logic, safety, or operational constraints.

Next, we consider several scientific and engineering domains where this plausibility-validity gap is especially pronounced, and where constrained-aware generation is essential.

1.1 Motivation through scientific and engineering applications

Protein design. Proteins are biological macromolecules that perform diverse functions, from catalysis to signaling to structural support. Designing novel proteins with specific structures and functions has transformative potential for therapeutics, diagnostics, and synthetic biology, and thus has been a central target for generative modeling [Huang et al., 2021, Dauparas et al., 2022, Anishchenko et al., 2021]. A typical protein design tasks centers around the generation of a "backbone structure", that is, the 3D coordinates of the protein's atoms. This is an extremely high-dimensional problem (typically thousands of atoms per protein) with complex constraints and has been a huge challenge for decades [Kuhlman and Baker, 2000].

A central shift in modern protein pipelines is the coupling of *structure prediction* with *generative design*. For example, AlphaFold demonstrated that learned geometric and evolutionary priors can yield highly accurate structure prediction in many regimes, substantially changing how constraints are represented and evaluated in protein modeling [Jumper et al., 2021]. On the generative side, RFdiffusion adapted RoseTTAFold-style showed how diffusion models can generate 3D protein backbones with motif- and target-conditioned design across monomers, binders, and symmetric oligomers [Watson et al., 2023]. Complementarily, ProteinMPNN showed strong empirical performance for sequence design conditioned on a backbone, and is now widely used as a sequence proposal component in generative design pipelines [Dauparas et al., 2022]. These tools are also being integrated into end-to-end design loops, where generated backbones are paired with sequence design and structure prediction to produce candidates for experimental validation [Ruffolo et al., 2023, Lindert et al., 2023].

Materials generation. Materials generation often targets the problem of discovering novel inorganic compounds or engineered microstructures with desired properties [Butler et al., 2018, Chen et al., 2023]. The process is extremely challenging given the vast combinatorial space of possible compositions and structures. Additionally, the validity requirements are stringent: valid materials must satisfy strict physical and chemical constraints (stoichiometry, symmetry, stability),

Diffusion-based generators for crystals are starting to shown increasing success in proposing novel structures while incorporating structural priors. For example, *MatterGen* is a diffusion-style generator for inorganic materials with explicit attention to stability-oriented steering [Zeni et al., 2024]. In reported evaluations, it generated a large set of novel crystal structure candidates with improved predicted thermodynamic stability relative to prior generative baselines, and the resulting candidates could be prioritized for downstream validation (for example via DFT screening) as part of an accelerated discovery pipeline. In these settings, the requirement of physical validity is often enforced through a combination of learned potentials (predictors of formation energy or stability) and hard symmetry constraints [Xie et al., 2021, Sun et al., 2023]. For engineered microstructures, that is, materials with complex mesoscale structures designed for target properties, diffusion-based inpainting and related generative approaches are being used to generate statistically consistent microstructures aligned with simulator outputs, supporting design beyond idealized crystals [Yang et al., 2023].

It is undoubted that GenAI could enable a revolution in materials discovery, as it has in protein design, given its potential to drastically accelerate the iteration cycles between proposal and validation. However, the validity gap is especially pronounced here: physical stability, synthesizability, and manufacturability are complex constraints that are not fully captured by data distributions alone. Indeed, in typical design cycles, high-fidelity density functional theory simulations or wet-lab synthesis are needed to validate candidates; at the same time, the absence of reliable surrogates models for these constraints makes purely data-driven generation risky.

Robotics and embodied decision making. Robotics generation targets the problem of producing action sequences and motion trajectories that achieve task objectives while satisfying stringent safety and feasibility constraints [Kober et al., 2013, Levine et al., 2020]. The process is extremely challenging given the combinatorial and continuous explosion of possible behaviors over time, together with partial observability and environment variability. Moreover, the validity requirements are stringent: valid trajectories must respect system dynamics, contact and friction constraints, collision avoidance, joint limits, and real-time latency budgets, and violations can be catastrophic.

Diffusion-based generative policies and planners are starting to show increasing success in proposing diverse behaviors while incorporating strong behavioral priors from data. For example, *Diffusion Policy* models visuomotor control as conditional diffusion over action sequences, demonstrating strong empirical performance on manipulation tasks and highlighting the benefit of diffusion models in capturing multi-modal action distributions in a stable training pipeline [Chi et al., 2023]. At the planning level, *Diffuser* reframes decision-making as trajectory generation, where iterative denoising plays the role of learned trajectory optimization and enables conditioning on goals and partial observations [Janner et al., 2022]. More recently, foundation-model approaches such as *RT-2* illustrate how large-scale vision-language pretraining can provide a semantic prior for control, enabling language-level task specifications to influence action selection and supporting broader generalization [Brohan et al., 2023], and multi-agent diffusion models are emerging as a way to capture complex interaction patterns in simulated environments [Liang et al., 2025].

It is increasingly plausible that GenAI will change robotics workflows. This is particularly valuable because robotics problems are naturally multi-solution: there are many feasible grasps, paths, and contact sequences for the same goal, and a generative policy can represent this multiplicity and re-sample when constraints change or disturbances occur. However, the validity gap is especially pronounced: sim-to-real distribution shift, unmodeled contacts, rare safety-critical edge cases, and compounding error over time all create regimes where purely learned priors can fail abruptly. As a result, progress hinges on explicit, independently validated constraint enforcement that couples generative proposals to verifiers and control-theoretic structure, rather than relying on learned scores alone [Amodei et al., 2016, Mayne, 2014].

1.2 Limitations for adoption in science and engineering and Course objective

While in all of these applications generative AI shows great promise, a common theme is that the main practical barrier to adoption is the *validity gap*: the difficulty of ensuring that generated outputs satisfy external constraints that are not fully captured by data distributions alone. In scientific and engineering settings, these constraints are not optional preferences but first principles that define correctness, safety, and feasibility. Proteins must fold into physically realizable structures while meeting binding and manufacturability requirements; crystals must satisfy symmetry, stoichiometry, and stability constraints; robot policies must respect dynamics and collision constraints under uncertainty; and engineered systems such as chips, power grids, and PDE-governed processes face hard design rules, conservation laws, and operational limits. If these constraints are violated, the output is not merely suboptimal, it is invalid, and the downstream pipeline fails, often expensively and sometimes dangerously.

The root cause is that standard generative models are trained to approximate a data distribution, not to satisfy a specification. Even when trained on "valid" datasets, learning such underlying distribution does not guarantee constraint satisfaction under distribution shift, novel designs, or adversarial prompts, precisely because the feasible region defined by domain rules can be a thin, structured subset of the ambient space.

These limitations are not artifacts of a specific architecture (Transformer versus diffusion) but structural: native token-level likelihood objectives and denoising objectives optimize average-case fit, not worst-case feasibility. When feasibility is rare, compositional, or global, the model can achieve excellent likelihood while still allocating nontrivial probability mass to invalid outputs. This is exactly the regime encountered in the applications above, where constraints are multi-scale and compositional, coupling local structure to global properties, and where small violations can render the final artifact unusable. Consequently, practical deployment requires methods that treat constraint satisfaction as a first-class component of the generative process

The course unifies these motivations under a single probabilistic objective: define a target distribution that combines a plausibility prior with constraint factors. The goal of the course is to bridge **Generative AI** (learning complex distributions) with **constrained optimization and constrained inference** (enforcing requirements). Concretely, we will return repeatedly to three organizing questions.

First, **Where** are constraints injected: during training, via architectural inductive bias, at inference time, or in post-processing. Second, **What** mathematical form the constraint takes: hard sets, soft penalties, logical formulas, expectation constraints, chance constraints, or hybrid combinations. Third, **How** we compute: via approximate sampling, variational approximations, or optimization.

2 The fundamental equation of constrained-aware generation

To tackle the validity gap, this course adopts a unified probabilistic framework for *constrained-aware generation.* The key idea is to define a *target* distribution that combines a base generative model with explicit constraint terms that encode validity requirements. Informally, we begin from a pretrained generator that acts as a strong *proposal* distribution over plausible objects, and then *tilt* this proposal toward validity by reweighting outputs that violate requirements and by excluding outputs that are outright infeasible.

Formally, let $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d$ denote the object to be generated (an image, a trajectory vector, a continuous embedding, or a relaxation of a discrete object) and let $\mathbf{c}$ denote context (a prompt, conditioning information, target properties, partial observations). We define a target conditional distribution $\pi(\mathbf{x} \mid \mathbf{c})$ as follows:

The Fundamental Equation of Constrained Generation

$$\pi(\mathbf{x} \mid \mathbf{c}) = \frac{1}{Z(\mathbf{c})} \quad \underbrace{p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{c})}_{\text{Based model (plausibility)}} \quad \underbrace{\exp(-\lambda\, \phi(\mathbf{x}, \mathbf{c}))}_{\text{Soft constraints}} \; \underbrace{\mathbf{1}\{\mathbf{x} \in \mathcal{C}(\mathbf{c})\}}_{\text{Hard constraints}} \,. \tag{1}$$

The base model $p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{c})$ is typically trained to approximate a data distribution or a conditional distribution induced by a dataset, and it provides a strong prior over *plausible* objects. The term $\phi(\mathbf{x}, \mathbf{c})$ is a real-valued *potential* that penalizes undesired properties, encodes preferences, or measures surrogate constraint violations. It is often (but not always) differentiable, which makes it amenable to gradient-based guidance and energy-based reweighting. Typically, the parameter $\lambda \geq 0$ is introduced to control the strength of this soft constraint term, trading off plausibility and validity. Finally, the hard constraint set $\mathcal{C}(\mathbf{c})$ encodes feasibility and is represented by an indicator; it therefore introduces non-smoothness and can induce discontinuities in $\pi(\mathbf{x} \mid \mathbf{c})$ because infeasible points receive zero probability mass.

During the course we will study multiple choices for each component and we will see how different algorithmic strategies arise from different modeling decisions. Before developing the technical pillars that support this framework, we unpack the two central aspects that govern both theory and practice: the partition function and the constraint representation.

1. Normalization and the partition function. The term $Z(\mathbf{c})$ is the *partition function* (or normalizing constant),

$$Z(\mathbf{c}) = \int_{\mathcal{X}} p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{c}) \, \exp(-\lambda\phi(\mathbf{x}, \mathbf{c})) \, \mathbf{1}\{\mathbf{x} \in \mathcal{C}(\mathbf{c})\} \, d\mathbf{x}, \tag{2}$$

with the integral replaced by a sum for discrete $\mathcal{X}$. Conceptually, $Z(\mathbf{c})$ rescales the unnormalized product so that $\pi(\cdot \mid \mathbf{c})$ integrates to one. Practically, however, $Z(\mathbf{c})$ is intractable in almost all settings where the framework is interesting: $\mathcal{X}$ is high-dimensional, ϕ may be nonconvex and learned, and the feasible region $\mathcal{C}(\mathbf{c})$ can carve out a thin subset of $\mathcal{X}$.

This intractability has three important consequences that recur throughout the course. First, we generally cannot evaluate likelihoods under π because $\log \pi(\mathbf{x} \mid \mathbf{c}) = \log p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{c}) - \lambda\phi(\mathbf{x}, \mathbf{c}) + \log \mathbf{1}\{\mathbf{x} \in \mathcal{C}(\mathbf{c})\} - \log Z(\mathbf{c})$ depends on $\log Z(\mathbf{c})$. Second, naïve *rejection sampling* from $p_{\boldsymbol{\theta}}$ becomes ineffective when constraints are strict, since the acceptance probability is essentially $Z(\mathbf{c})$ relative to the mass of $p_{\boldsymbol{\theta}}$ and can be exponentially small when feasibility is rare. Third, if one attempted to *learn* an unnormalized model of the form $\pi(\mathbf{x} \mid \mathbf{c}) \propto \exp(-E_{\boldsymbol{\theta}}(\mathbf{x}; \mathbf{c}))$, where E is an energy term (See Section 3 for more details), then $\nabla_{\boldsymbol{\theta}} \log Z(\mathbf{c})$ introduces a "negative phase" expectation under $\pi(\cdot \mid \mathbf{c})$, which typically requires sampling in the inner loop and is a primary source of computational cost in energy-based learning.

The first major conceptual pivot of the course is therefore that constrained-generation methods should be understood as *ways to sample from, or optimize with respect to, Equation (1) without computing Equation (2)*. This is why we will emphasizes tools such as Langevin-type dynamics and score-based transitions that depend on gradients of $\log \pi$ rather than its normalization, variational approximations that trade off expected energy and entropy without requiring exact partition function evaluation, and MCMC methods whose acceptance ratios cancel $Z(\mathbf{c})$.

2. Base model as a prior, constraints as a likelihood factor. A useful way to interpret Equation (1) is as a "Bayes-like" factorization. The base model $p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{c})$ plays the role of a *prior* over plausible objects, while the constraint terms act as a *likelihood* that favors validity. In particular, the soft factor $\exp(-\lambda\phi(\mathbf{x}, \mathbf{c}))$ can be read as a Gibbs likelihood that rewards low-cost configurations, and the hard indicator can be read as an infinite-penalty likelihood that rules out infeasibility. This viewpoint clarifies why constrained generation is an inference problem: we are effectively computing a posterior over designs given an external specification. It also clarifies why constraints cannot be treated as an afterthought in high-stakes settings: the posterior may place negligible mass on feasible designs unless constraint injection is explicit and strong.

Hard versus soft constraints. Hard constraints define membership in $\mathcal{C}(\mathbf{c})$ and are typically checkable by a verifier, for example compilation success, grammar membership, satisfaction of a set of convex inequalities, collision-free feasibility under geometric checks, or satisfaction of a conservation law under a simulator. Soft constraints are encoded by $\phi(\mathbf{x}, \mathbf{c})$ and are more flexible: they can represent preferences, costs, and surrogate measurements, such as toxicity predictors, property predictors, reward models, or relaxed penalties for constraint violations.

From an algorithmic standpoint, the key distinction is that soft constraints can often be injected through gradients (guidance) and energy reweighting, whereas hard constraints require discrete restriction, projection/repair, or a verifier-in-the-loop mechanism. From a reliability standpoint, the key distinction is that hard constraints can offer *certifiability* when verification is correct, while soft constraints are only as reliable as the proxy ϕ. This proxy dependence creates two recurring risks. The first is *misspecification*: ϕ may fail to capture the true constraint, particularly under distribution shift. The second is *proxy hacking*: optimization or sampling under ϕ can exploit weaknesses of the surrogate and produce artifacts that score well under ϕ but fail under true validation.

A recurring design choice is therefore whether to encode a requirement as hard or soft, or to combine the two. Hard constraints are appropriate when feasibility is verifiable, but they may be computationally expensive to enforce and may lead to low acceptance rates if enforced by rejection. Soft constraints are computationally convenient and can be smoothly traded off with plausibility via λ, but they require careful calibration and independent verification to ensure they correlate with true validity.

3 Energy-Based Models and the partition function problem

This section briefly discusses what an Energy-Based Model (EBM) is, why EBMs appear naturally in constrained generation, and what the partition function prevents you from doing.

3.1 Energy Based Models: Definition

An EBM parameterizes a probability distribution through an energy function E, with probability mass concentrated on low-energy configurations [LeCun et al., 2006, Wainwright and Jordan, 2008]. The energy function plays the role of a *global score*: it aggregates multiple desiderata into a single scalar. As reviewed in Equation (1), in constrained generation, the energy naturally decomposes into a plausibility term (coming from the base model), a soft penalty term (coming from ϕ), and a hard feasibility term (coming from the indicator of $\mathcal{C}$). This decomposition is exactly what we will exploit throughout the course: different methods implement different approximations to sampling or optimization under the implied EBM. However, to turn energies into probabilities we are required to compute a normalization constant (the partition function Z); the intractability of this function is the central computational obstacle in energy-based learning.

The following definition formalizes the notion of a conditional EBM.

Definition 1 (Conditional Energy-Based Model)**.** *Given an energy function $E(\mathbf{x};\mathbf{c}) \in \mathbb{R}\cup\{+\infty\}$, an EBM defines a conditional distribution*

$$\pi(\mathbf{x} \mid \mathbf{c}) = \frac{\exp(-E(\mathbf{x};\mathbf{c}))}{Z(\mathbf{c})}, \qquad Z(\mathbf{c}) = \int_{\mathcal{X}} \exp(-E(\mathbf{x};\mathbf{c}))\, d\mathbf{x}, \tag{3}$$

provided $Z(\mathbf{c}) < \infty$.

Note that Equation (1) is an EBM with energy:

$$E(\mathbf{x};\mathbf{c}) = -\log p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{c}) + \lambda\phi(\mathbf{x},\mathbf{c}) + I_{\mathcal{C}(\mathbf{c})}(\mathbf{x}), \tag{4}$$

where $I_{\mathcal{C}(\mathbf{c})}(\mathbf{x}) = 0$ if $\mathbf{x} \in \mathcal{C}(\mathbf{c})$ and $+\infty$ otherwise. This is the standard way to encode hard constraints in energy-based inference: feasibility is represented as an infinite barrier, which removes infeasible configurations from the support [LeCun et al., 2006, Wainwright and Jordan, 2008].

Remark 1 (Continuous vs. discrete domains)**.** *For continuous $\mathcal{X} \subseteq \mathbb{R}^d$, the partition function in Equation (3) is an integral. For discrete domains (language, graphs), it is a sum over exponentially many configurations. In both cases, $Z(\mathbf{c})$ is typically intractable, which is why we emphasize methods that do not require evaluating it.*

3.2 The partition function problem: what it blocks and why

The partition function $Z(\mathbf{c})$ is the cost of converting a relative preference score (the energy) into a normalized probability distribution. Unfortunately, $Z(\mathbf{c})$ depends on *all* configurations in $\mathcal{X}$, not just those we observe. This dependence has two immediate consequences. First, even if we can compute the energy of a given $\mathbf{x}$, we usually cannot compute its normalized probability. Second, if we try to learn the energy model by maximum likelihood, the gradient contains an expectation under the model itself, which requires sampling during training. This "sampling-inside-learning" loop is the classic computational bottleneck of EBMs.

Thus, the partition function $Z(\mathbf{c})$ blocks two operations that would otherwise be straightforward:

(1) **Likelihood evaluation is unavailable.** Even when $E(\mathbf{x};\mathbf{c})$ is computable, the log-likelihood decomposes as

$$\log \pi(\mathbf{x} \mid \mathbf{c}) = -E(\mathbf{x};\mathbf{c}) - \log Z(\mathbf{c}), \tag{5}$$

and $\log Z(\mathbf{c})$ is generally unknown. This makes direct likelihood-based evaluation and calibration difficult for EBMs.

(2) **Maximum likelihood learning introduces the negative phase.** Suppose we attempt to learn an unnormalized conditional model

$$\pi_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{c}) \;\propto\; \exp(-E_{\boldsymbol{\theta}}(\mathbf{x};\mathbf{c}))\,.$$

Then, differentiating the log-likelihood yields

$$\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{c}) = -\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\mathbf{x};\mathbf{c}) \;-\; \nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}}(\mathbf{c}). \tag{6}$$

The first term depends only on the observed pair $(\mathbf{x},\mathbf{c})$ and is often called the *positive phase*, because it pushes down the energy of data configurations. The second term can be written as a model expectation:

$$\nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}}(\mathbf{c}) = \frac{1}{Z_{\boldsymbol{\theta}}(\mathbf{c})} \nabla_{\boldsymbol{\theta}} \int \exp(-E_{\boldsymbol{\theta}}(\mathbf{x};\mathbf{c}))\, d\mathbf{x} = -\mathbb{E}_{\mathbf{x}\sim\pi_{\boldsymbol{\theta}}(\cdot\mid\mathbf{c})}\big[\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\mathbf{x};\mathbf{c})\big]. \tag{7}$$

Substituting Equation (7) into Equation (6) shows that maximum likelihood learning requires estimating an expectation under the current model distribution $\pi_{\boldsymbol{\theta}}(\cdot \mid \mathbf{c})$. This term is called the *negative phase* because it pushes up the energy of samples drawn from the model, preventing the model from collapsing probability mass onto a small subset of configurations. Computing it typically requires MCMC (or related sampling) in the inner loop, which is the primary source of computational cost in classical energy-based learning [LeCun et al., 2006, Hinton, 2002].

3.3 Inference without $Z(\mathbf{c})$

Although $Z(\mathbf{c})$ obstructs likelihood evaluation and makes maximum likelihood learning expensive, it does *not* prevent us from sampling or performing approximate inference. Many inference algorithms depend only on *energy differences* or *energy gradients*. This is why EBMs are viable as targets for constrained generation even when they are difficult to train from scratch.

Metropolis-Hastings. Metropolis-Hastings constructs a Markov chain with stationary distribution $\pi(\cdot \mid \mathbf{c})$. Its acceptance probability involves the ratio

$$\frac{\pi(\mathbf{x}' \mid \mathbf{c})}{\pi(\mathbf{x} \mid \mathbf{c})} = \frac{\exp(-E(\mathbf{x}';\mathbf{c}))/Z(\mathbf{c})}{\exp(-E(\mathbf{x};\mathbf{c}))/Z(\mathbf{c})} = \exp\big(-(E(\mathbf{x}';\mathbf{c}) - E(\mathbf{x};\mathbf{c}))\big),$$

so the partition function cancels. This cancellation is the basic reason MCMC is the canonical inference tool for EBMs [Robert and Casella, 2004].

Langevin dynamics and score-based transitions. If $E(\mathbf{x};\mathbf{c})$ is differentiable (or differentiable almost everywhere), Langevin dynamics defines a diffusion process that targets π as its stationary distribution under standard conditions. In continuous space, the Langevin SDE is

$$d\mathbf{x}_t = -\nabla_{\mathbf{x}} E(\mathbf{x}_t;\mathbf{c})\, dt + \sqrt{2}\, d\mathbf{w}_t, \tag{8}$$

where $\mathbf{w}_t$ is standard Brownian motion. A simple Euler discretization yields the unadjusted Langevin algorithm (ULA),

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta \nabla_{\mathbf{x}} E(\mathbf{x}_k;\mathbf{c}) + \sqrt{2\eta}\, \epsilon_k, \qquad \epsilon_k \sim \mathcal{N}(0, I). \tag{9}$$

The key point is that $\nabla_{\mathbf{x}} E$ does not depend on $Z(\mathbf{c})$. When the energy includes a hard indicator $I_{\mathcal{C}(\mathbf{c})}$, classical gradients fail at the boundary, motivating *projected* or *reflected* Langevin variants that replace the nonsmooth term by a projection or proximal step. This connects directly to the convex-optimization primitives developed later and to inference-time constraint enforcement methods such as projected diffusion [Christopher et al., 2024].

Noise-contrastive estimation and score matching. Finally, although maximum likelihood learning of EBMs is hindered by the negative phase, alternative estimation principles avoid explicit normalization. Noise-contrastive estimation reduces density estimation to classification between data and known noise [Gutmann and Hyvärinen, 2010]. Score matching estimates the score $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ without estimating $p(\mathbf{x})$ itself [Hyvärinen, 2005]. These ideas are conceptually central to diffusion models and will be revisited when we discuss score-based generative modeling and modular constraint injection.

4 Convex analysis primitives: projection and proximal operators

As discussed in Section 2, hard constraints are represented by an indicator function that is zero on the feasible set and $+\infty$ outside it. Even when the set is smooth, the indicator potential is discontinuous. As a result, enforcing hard constraints frequently becomes an optimization subproblem embedded in a generative procedure: a generative procedure produces an intermediate candidate $\mathbf{y}$ that is plausible under the base model, and then we apply an operator that returns the "smallest" modification that restores feasibility. In the convex regime, this operator has a precise mathematical definition and enjoys strong stability properties; this is the point at which convex analysis enters the constrained-generation story [Boyd and Vandenberghe, 2004, Bubeck, 2015].

4.1 Projection

The projection operator formalizes the intuition of *minimal repair*: it maps an invalid point to the nearest valid point, with nearness measured by a chosen norm. When inserted repeatedly, projection becomes a mechanism for maintaining validity throughout a generative trajectory, rather than repairing only at the end.

Definition 2 (Euclidean projection). *Let $\mathcal{C} \subseteq \mathbb{R}^d$ be nonempty and closed. The Euclidean projection is*

$$\mathrm{Proj}_{\mathcal{C}}(\mathbf{y}) \triangleq \arg\min_{\mathbf{x}\in\mathcal{C}} \frac{1}{2}\|\mathbf{x}-\mathbf{y}\|_2^2. \tag{10}$$

When $\mathcal{C}$ is convex, the minimizer exists and is unique [Bubeck, 2015, Boyd and Vandenberghe, 2004]. This is not a minor technicality. Uniqueness implies that projection defines a well-posed operator $\mathrm{Proj}_{\mathcal{C}} : \mathbb{R}^d \to \mathcal{C}$ that can be inserted into iterative algorithms (for sampling, optimization, or decoding) as a deterministic feasibility correction. Moreover, projection is *stable*: small changes in $\mathbf{y}$ cannot lead to arbitrarily large changes in $\mathrm{Proj}_{\mathcal{C}}(\mathbf{y})$, a property that will later underpin differentiability and end-to-end learning through optimization layers.

4.2 Proximal operators

Projection is a special case of a more general operator that will recur across the course in both optimization layers and constrained sampling methods.

Definition 3 (Proximal operator)**.** *For a proper, closed, convex function* $f : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$*, define*

$$\operatorname{prox}_f(\mathbf{y}) \triangleq \operatorname*{arg\,min}_{\mathbf{x}\in\mathbb{R}^d} \left\{ f(\mathbf{x}) + \frac{1}{2}\|\mathbf{x}-\mathbf{y}\|_2^2 \right\}. \tag{11}$$

If $f = I_{\mathcal{C}}$ is the indicator of a convex set, then $\operatorname{prox}_{I_{\mathcal{C}}} = \operatorname{Proj}_{\mathcal{C}}$. Proximal operators are the building blocks of splitting methods such as proximal gradient and ADMM [Parikh and Boyd, 2014, Boyd et al., 2011]. Later in the course, they will also serve as the cleanest language for discussing projected diffusion models [Christopher et al., 2024], because those methods alternate a generative update with a projection-like correction.

4.3 Differentiable optimization layers

So far, projection and proximal steps serve as inference-time correction operators. A separate, but closely related, idea is to embed constraint satisfaction directly into a learning pipeline by placing an optimization problem inside the model. In that setting, the forward pass *solves* a constrained problem, and the backward pass differentiates through its solution map. This is a principled way to couple prediction and constrained decision-making, and it is one of the main routes by which convex optimization enters modern end-to-end learning [Amos and Kolter, 2017] [Agrawal et al., 2019].

In later lectures we will formalize when these solution maps are differentiable, how to compute Jacobian-vector products efficiently, and how these ideas interface with constrained generation objectives.

5 A taxonomy of constraints

The word "constraint" is overloaded in the generative AI literature. It may refer to a strict feasibility requirement (a molecule must satisfy valence), a soft preference (increase aesthetic score), a downstream objective (maximize binding affinity), or a safety policy (avoid disallowed content). It's important to realize that the mathematical form of the constraint is often the single most important predictor of which algorithmic tools are appropriate and which guarantees are even meaningful. Thus, before diving into methods, we pause to classify constraints along three key axes. First, *mathematical structure*, which governs tractability and the availability of stable operators (projection, proximal maps, dual certificates). Second, *domain type*, which determines whether constraints can be enforced by continuous optimization, discrete search, or hybrid methods. Third, *verifiability*, which determines whether constraint satisfaction can be certified by an external oracle or only approximated by learned proxies.

5.1 Axis 1: mathematical structure

We classify constraints based on their mathematical representation in Equation (1), as hard set constraints, soft penalty/energy terms, or distributional requirements.

- **Hard constraints:** A hard constraint specifies a feasible set $\mathcal{C}(\mathbf{c}) \subseteq \mathcal{X}$ and requires $\mathbf{x} \in \mathcal{C}(\mathbf{c})$. In the unified target of Equation (1), this appears via the indicator $\mathbf{1}\{\mathbf{x} \in \mathcal{C}(\mathbf{c})\}$, or equivalently via the barrier $I_{\mathcal{C}(\mathbf{c})}(\mathbf{x})$ in the energy representation of Equation (4). The geometry of $\mathcal{C}(\mathbf{c})$ drives algorithm selection. If $\mathcal{C}(\mathbf{c})$ is convex, feasibility restoration can be implemented by the projection operator in Equation (10) and analyzed using convex-analytic tools [Boyd and Vandenberghe, 2004, Bubeck, 2015]. If $\mathcal{C}(\mathbf{c})$ is nonconvex, projection may be ill-posed or multi-valued, and algorithms typically rely on relaxations, local methods, or problem-specific heuristics.

- **Penalty and energy constraints:** Soft constraints specify a real-valued potential $\phi(\mathbf{x}, \mathbf{c})$ that encodes preferences, costs, or surrogate constraint violations. In Equation (1), ϕ enters multiplicatively as $\exp(-\lambda\phi)$, and in the EBM view it contributes additively to the energy. When ϕ is differentiable with respect to $\mathbf{x}$, it admits gradient-based injection. This will allow us to impose guidance through Langevin-type updates and score-based methods, as we will study later in this course. Note however that the ϕ may be only a proxy for the true requirement; in that case, "reducing" ϕ does not imply "increasing" validity, and the resulting system can exhibit proxy exploitation unless additional verification is present.
- **Distributional constraints:** Finally. some requirements are naturally expressed at the distribution level, for example

$$\mathbb{E}_{\mathbf{x}\sim\pi(\cdot|\mathbf{c})}[g(\mathbf{x}, \mathbf{c})] \le \tau, \qquad \text{or} \qquad \mathbb{E}_{\mathbf{x}\sim\pi}[g(\mathbf{x})] = m,$$

rather than pointwise constraints on individual samples. Such constraints connect to maximum entropy and exponential-family modeling, where Lagrange multipliers introduce exponential tilting terms that resemble the soft-constraint factor in Equation (1) [Wainwright and Jordan, 2008]. Practically, these constraints often require estimating expectations (hence sampling or variational approximations) and raise questions about generalization under distribution shift.

5.2 Axis 2: domain type

Next we classify constraints based on the nature of the output space $\mathcal{X}$. In particular, we distinguish continuous, discrete, and hybrid domains, because the domain type determines which algorithmic tools are available.

- **Continuous domains:** $\mathcal{X} \subseteq \mathbb{R}^d$, e.g., images, trajectories, geometric designs, physical fields. This setting makes gradients and SDE/ODE-based samplers available, and it makes convex constraints particularly tractable through projections and proximal maps [Boyd and Vandenberghe, 2004, Bubeck, 2015]. Even when constraints are nonconvex, continuous relaxations can still yield useful local correction operators.
- **Discrete domains:** $\mathcal{X}$ is combinatorial, e.g., language, code, graphs. Hard constraints often correspond to grammars, automata, type systems, or logical theories, and enforcement frequently reduces to constrained decoding, search, or verification-guided generation. Continuous relaxations can be used as approximations, but the fundamental difficulty is that small continuous changes do not necessarily correspond to valid discrete edits, which complicates the direct application of gradient-based constraint mechanisms.
- **Hybrid domains:** $\mathcal{X}$ has both continuous and discrete components. This is common in many scientific problems such as molecular design where the graph topology is discrete but geometric conformations are continuous.

5.3 Axis 3: verifiability

Finally, we classify constraints based on whether they can be *verified* by an external oracle or only approximated by learned proxies.

- **Oracle-verifiable constraints:** Some constraints admit an external verifier that can certify satisfaction, such as compilation success, exact conservation laws, collision checking with a trusted geometric model, or membership in a convex set described by known inequalities. When such an oracle exists, constraint satisfaction can be made certifiable, either by post-processing (projection, repair) or by verifier-in-the-loop generation. The remaining challenge is typically computational: oracle calls may be expensive, and naive rejection can be inefficient in high dimensions.

- **Learned proxies and surrogate constraints:** Many practical constraints are enforced by learned predictors: toxicity classifiers, preference models, property predictors in chemistry and materials, and surrogate stability estimators. These proxies are often essential for scaling, but they introduce model risk and can be exploited by the generator when used as sole objectives. A recurring methodological theme in constrained generative AI is therefore how to combine proxy guidance with independent verification, and how to evaluate robustness under prompt shift, distribution shift, and adversarial inputs. In the language of Equation (1), this often means treating ϕ as a steering signal while reserving $\mathcal{C}$ (or additional verifiers) for requirements that must be satisfied with high confidence.

6 Where do constraints enter?

This course is structured around the constraint injection stage because it often determines what is feasible computationally and what guarantees are realistic. In particular, we distinguish four main modes of constraint injection, each with its own algorithmic tools and theoretical underpinnings.

1. Training-time injection Training-time approaches seek to learn $p_{\boldsymbol{\theta}}$ so that it places most mass on feasible and desirable regions. One approach is to add penalties to the training objective, another is to curate data so that constraints are reflected in the training distribution, and a third is to train end-to-end through a downstream solver.

Decision-focused learning is a canonical solver-in-the-loop training paradigm: one trains a predictive model so that, after a downstream optimization step, decisions are good, rather than merely predictions [Mandi et al., 2023]. The core technical enabler is differentiating through argmin operators, typically in convex settings where the solution map is well-behaved [Amos and Kolter, 2017, Agrawal et al., 2019].

2. Architectural injection Architectural methods constrain the range of the model, for example through symmetry constraints (equivariance) or structured decoders. In scientific applications, geometric deep learning is often the architectural vehicle for enforcing invariances, such as SE(3) equivariance in molecular and protein settings. These architectural choices can encode constraints "by construction" but are harder to retrofit into large pretrained models.

3. Inference-time injection Inference-time methods treat $p_{\boldsymbol{\theta}}$ as a prior and enforce constraints during generation. For diffusion models, guidance adds gradient terms to the reverse dynamics [Dhariwal and Nichol, 2021]; for autoregressive LMs, constrained decoding restricts token choices by a constraint automaton or a verifier. For hard continuous constraints, projected diffusion models insert projection steps into the sampling trajectory to maintain feasibility throughout generation [Christopher et al., 2024]. This course will emphasize inference-time approaches because they often provide the most flexible response to new constraints without retraining.

4. Post-processing and repair Post-processing applies a repair map R after generation, for example projection, solver-based repair, or local refinement. Post-processing can guarantee feasibility when the repair method is correct, but it can distort the distribution significantly if repairs are large. A recurring principle is therefore to distribute repair across generation steps, making each repair small. This is one way to interpret projection-at-each-step methods in diffusion-based constrained synthesis [Christopher et al., 2024].

7 Summary and outlook

Lecture 1 establishes the course's unifying abstraction: constrained-aware generation is sampling or optimization under a target distribution obtained by combining a base model with soft potentials and hard feasibility sets, as formalized by Equation (1). The main theoretical obstacle is the partition function in Equation (2), which pushes us toward methods that do not require normalization, notably MCMC and Langevin-type dynamics, as well as learning principles such as score matching. Hard constraints introduce nonsmoothness, motivating projection and proximal operators and, later, differentiable optimization layers that bring convex solvers into learning and inference.

In Lecture 2 we will build the base-model side more explicitly (maximum likelihood, latent-variable models, and the meaning of conditional modeling), so that subsequent lectures can clearly separate what is inherited from $p_{\boldsymbol{\theta}}$ and what is introduced by constraint injection.

References

A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter. Differentiable Convex Optimization Layers. *NeurIPS*, 2019.

B. Amos and J. Z. Kolter. OptNet: Differentiable Optimization as a Layer in Neural Networks. *ICML*, 2017.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends in Machine Learning*, 2011.

K. Bi et al. Accurate medium-range global weather forecasting with 3D neural networks. *Nature*, 2023.

T. Brown et al. Language Models are Few-Shot Learners.

S. Bubeck. Convex Optimization: Algorithms and Complexity. *Foundations and Trends in Machine Learning*, 2015.

K. T. Butler, D. W. Davies, H. Cartwright, O. Isayev, and A. Walsh. Machine learning for molecular and materials science. *Nature*, 559:547–555, 2018.

(Placeholder citation) W. Chen and *others*. Generative modeling for materials discovery and design: a review. *arXiv preprint*, 2023.

J. K. Christopher, S. Baek, and F. Fioretto. Constrained Synthesis with Projected Diffusion Models. *NeurIPS*, 2024. arXiv:2402.03559. Constrained Synthesis with Projected Diffusion Models. *NeurIPS*, 2024. arXiv:2402.03559.

H. Chi et al. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. *Robotics: Science and Systems (RSS)*, 2023.

J. Dauparas et al. Robust deep learning–based protein sequence design using ProteinMPNN. *Science*, 2022.

P. Dhariwal and A. Nichol. Diffusion Models Beat GANs on Image Synthesis. *NeurIPS*, 2021.

M. Janner, Y. Duan, A. Gupta, and S. Levine. Diffuser: Diffusion Models for Offline Reinforcement Learning. *ICLR*, 2022.

I. Goodfellow et al. Generative Adversarial Nets. *NeurIPS*, 2014.

M. Gutmann and A. Hyvärinen. Noise-Contrastive Estimation: A New Estimation Principle for Unnormalized Statistical Models. *AISTATS*, 2010.

R. Lam et al. GraphCast: Learning skillful medium-range global weather forecasting. *Science*, 2023.

G. Hinton. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, 2002.

J. Ho, A. Jain, and P. Abbeel. Denoising Diffusion Probabilistic Models. *NeurIPS*, 2020.

G. Huang et al. A review of deep learning for protein design. *arXiv preprint*, 2021.

I. Anishchenko et al. De novo protein design by deep network hallucination. *Nature*, 600:547–552, 2021.

A. Hyvärinen. Estimation of Non-Normalized Statistical Models by Score Matching. *JMLR*, 2005.

D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. *ICLR*, 2014.

B. Kuhlman and D. Baker. Native protein sequences are close to optimal for their structures. *Proceedings of the National Academy of Sciences*, 97(19):10383–10388, 2000.

Y. LeCun, S. Chopra, and R. Hadsell. A Tutorial on Energy-Based Learning. In *Predicting Structured Data*. MIT Press, 2006.

A. Mirhoseini et al. A graph placement methodology for fast chip design. *Nature*, 2021.

J. Mandi et al. Decision-Focused Learning: Foundations, State of the Art, Benchmark and Future Opportunities. *JAIR*, 2024. arXiv:2307.13565.

N. Parikh and S. Boyd. Proximal Algorithms. *Foundations and Trends in Optimization*, 2014.

J. Pathak et al. FourCastNet: A global data-driven high-resolution weather model using adaptive Fourier neural operators. *Proceedings of Machine Learning and Systems (PMLR)*, 2022.

J. Noh, J. Kim, H. S. Stein, and Y. Jung. Inverse Design of Inorganic Crystals Using Generative Adversarial Networks. *npj Computational Materials*, 2019.

D. Rezende, S. Mohamed, and D. Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *ICML*, 2014.

C. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer, 2004.

Y. Song and S. Ermon. Generative Modeling by Estimating Gradients of the Data Distribution. *NeurIPS*, 2019.

B. Sánchez-Lengeling and A. Aspuru-Guzik. Inverse Molecular Design Using Machine Learning: Generative Models for Matter Engineering. *Science*, 2018.

A. Vaswani et al. Attention is All You Need. *NeurIPS*, 2017.

J. L. Watson et al. De novo design of protein structure and function with RFdiffusion. *Nature*, 2023.

M. Wainwright and M. Jordan. Graphical Models, Exponential Families, and Variational Inference. *Foundations and Trends in Machine Learning*, 2008.

J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, *et al.* Highly accurate protein structure prediction with AlphaFold. *Nature*, 596:583–589, 2021.

J. L. Watson, D. Juergens, N. B. Bennett, B. L. Trippe, J. Yim, H. E. Eisenstein, M. K. Ahern, P. Xie, and D. Baker. RFdiffusion: A generative model for protein backbones. *bioRxiv*, 2023. Preprint.

J. Dauparas, I. Anishchenko, S. Bennett, H. Bai, R. R. H. Ragotte, L. F. Milles, B. I. Wicky, A. M. Courbet, R. J. de Haas, N. P. Bethel, P. J. Le Poulain, A. J. R. Pellock, D. Tischer, A. H. C. Chan, B. Koepnick, H. Nguyen, A. K. Kang, B. J. Sankaran, S. E. Bick, D. Baker, and S. Ovchinnikov. Robust deep learning–based protein sequence design using ProteinMPNN. *Science*, 378(6615):49–56, 2022.

J. A. Ruffolo, J. J. Gray, and J. K. Leman. Antibody design with generative models and epitope constraints. *arXiv preprint*, 2023.

(Placeholder citation) S. Lindert and *others*. End-to-end protein design loops combining backbone generation, sequence design, and structure prediction. *arXiv preprint*, 2023.

X. Lu, Y. Zhou, and *others*. Diffusion models for antibody design with epitope targeting. *arXiv preprint*, 2023.

A. Merchant, S. M. Batzner, S. S. Schoenholz, E. Aykol, G. Simm, and E. D. Cubuk. Scaling deep learning for materials discovery. *Nature*, 624:80–85, 2023.

T. Xie, J. Fu, S. Gao, and *others*. CDVAE: Crystal diffusion variational autoencoder for periodic materials. *arXiv preprint*, 2021.

W. Sun, C. Li, and *others*. DiffCSP: A diffusion model for crystal structure prediction and generation. *arXiv preprint*, 2023.

J. Köhler, A. Klein, and F. Noé. Flow matching: Generative modeling by matching flows. *arXiv preprint*, 2023.

(Placeholder citation) J. Gu and *others*. FlowMM: Flow matching on manifolds for materials generation. *arXiv preprint*, 2024.

(Placeholder citation) Y. Yang and *others*. Diffusion-based generative modeling for microstructure synthesis and design. *arXiv preprint*, 2023.

C. Zeni, R. Pinsler, D. Zügner, A. Fowler, M. Horton, X. Fu, S. Shysheya, J. Crabbé, L. Sun, J. Smith, B. Nguyen, H. Schulz, S. Lewis, C.-W. Huang, Z. Lu, Y. Zhou, H. Yang, H. Hao, J. Li, R. Tomioka, and T. Xie. MatterGen: a generative model for inorganic materials design. *arXiv preprint arXiv:2312.03687*, 2024.

C. Chi, Z. Xu, S. Feng, E. B. Cousineau, Y. Zhao, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems (RSS)*, 2023.

M. Janner, Y. Li, and S. Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning (ICML)*, 2022.

A. Brohan, N. Brown, J. Carvalho, and *others*. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint*, 2023.

J. Liang, J. K. Christopher, S. Koenig, and F. Fioretto. Simultaneous Multi-Robot Motion Planning with Projected Diffusion Models. *International Conference on Machine Learning (ICML)*, 2025. arXiv:2502.03607.

R. Lam, A. Sanchez-Gonzalez, A. Willson, P. Wirnsberger, A. Fortunato, D. Alet, and *others*. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023.

I. Price, A. Sanchez-Gonzalez, and *others*. GenCast: Probabilistic weather forecasting with generative models. *arXiv preprint*, 2023.

J. Pathak, S. Subramanian, J. Harrington, A. R. Raja, and *others*. FourCastNet: A global data-driven high-resolution weather model using adaptive Fourier neural operators. *arXiv preprint*, 2022.

A. Mirhoseini, A. Goldie, M. Yazgan, J. Jiang, E. Songhori, S. Wang, Y. Lee, E. Johnson, and *others*. A graph placement methodology for fast chip design. *Nature*, 594:207–212, 2021.

Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

(Placeholder citation) M. Khalil and *others*. Diffusion models for topology optimization and generative design. *arXiv preprint*, 2023.

P. Donti, B. V. Roy, S. Kolter, and *others*. Machine learning for decision making in power systems. *Proceedings of the IEEE*, 109(9):1651–1676, 2021.

J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané. Concrete Problems in AI Safety. *arXiv preprint arXiv:1606.06565*, 2016.

D. Q. Mayne. Model predictive control: Recent developments and future promise. *Automatica*, 50(12):2967–2986, 2014.

P. Pinson. On generating scenarios of wind power production. *Renewable Energy*, 55:68–74, 2013.