Generative AI with Stochastic Differential Equations

An introduction to flow and diffusion models

MIT IAP 2025 | Jan 21, 2025

Peter Holderrieth and Ezra Erives

Sponsor: Tommi Jaakkola

Welcome to class 6.S184/6.S975!

Course Instructors

Peter

Ezra

Sponsor

Tommi

Generative AI - A new generation of AI systems

Artistic Images

Realistic Videos

ChatGPT

Draft Texts

These systems are “creative”: they generate new objects.

This class teaches you algorithms to generate objects.

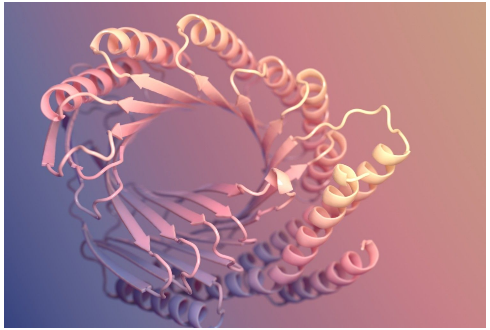
Flow and Diffusion Models: state-of-the-art models for generating images, videos, proteins!

Stable Diffusion

DALEE

OpenAI Sora

Meta MovieGen

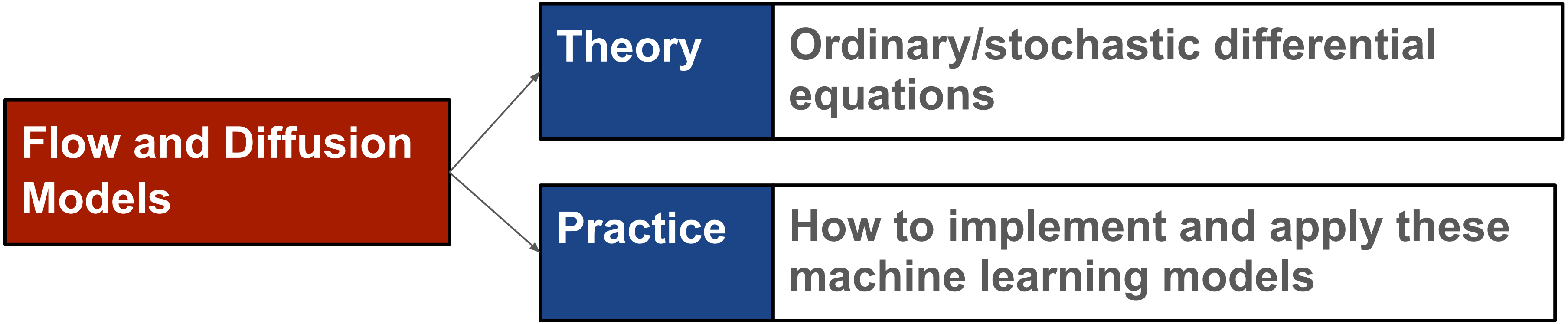


AlphaFold3

RFDiffusion

Most SOTA generative AI models for images/videos/proteins/robotics: Diffusion and Flow Models

This Class: Theory and Practice of Flow/Diffusion Models



Flow and Diffusion Models

Theory Ordinary/stochastic differential equations

Practice How to implement and apply these machine learning models

The goal of this class to teach you:

1. Flow and diffusion models from *first principles.*
2. The *minimal* but necessary amount of mathematics for 1.
3. How to implement and apply these algorithms.

Class Overview

- **Lecture 1 (today):**
    - From Generation to Sampling: Formalize “generating an image/etc.”
    - Construct Flow and Diffusion Models
- **Lecture 2 - Marginal Vector Field and Score:** Derive training targets.
- **Lecture 3 - Flow Matching and Score Matching:** Define training algorithm.
- **Lecture 4 - Build Image Generators:** Network architectures + conditioning
- **Lecture 5 - Advanced Topics:** Alignment, complex data types, distillation
- **Lecture 6 - Guest lecture:**
    - Jason Yim (MIT): **Protein design**
    - Benjamin Burchfiel (Toyota Research Institute): **Robotics**

Section 1:

From Generation to Sampling

Goal: Formalize what it means to “generate” something.

We represent images/videos/protein as vectors

Images:

- Height H and Width W
- 3 color channels (RBG)

$$z \in \mathbb{R}^{H \times W \times 3}$$

Videos:

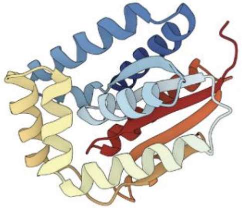
- T time frames
- Each frame is image

$$z \in \mathbb{R}^{T \times H \times W \times 3}$$

Molecular structures:

- N atoms
- Each atom has 3 coordinate

$$z \in \mathbb{R}^{N \times 3}$$

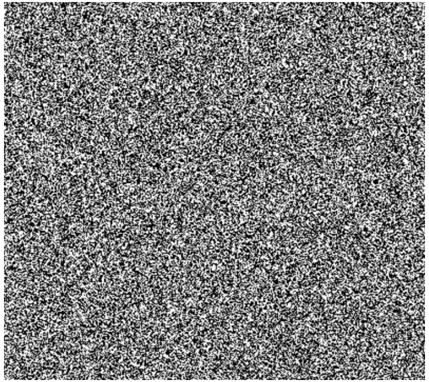


We represent the objects we want to generate as vectors:

$$z \in \mathbb{R}^d$$

What does it mean to successfully generate something?

Prompt: “A picture of a dog”

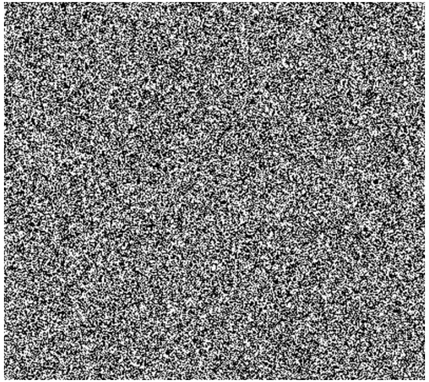



Useless < Bad < Wrong animal < Great!

These are subjective statements - Can we formalize this?

Data Distribution: How “likely” are we to find this picture in the internet?

Prompt: “A picture of a dog”



Impossible < Rare < Unlikely < Very likely

How good an image is ~= How likely it is under the data distribution

Generation as sampling from the data distribution

Data distribution: Distribution of objects that we want to generate: $\boxed{p_{\text{data}}}$

Probability density:

$$\boxed{\begin{aligned} p_{\text{data}} : \mathbb{R}^d &\to \mathbb{R}_{\geq 0}, \\ z &\mapsto p_{\text{data}}(z) \end{aligned}}$$

Note: We don't know the probability density!

Generation means sampling the data distribution:

$$\boxed{z \sim p_{\text{data}}} \Rightarrow z =$$

A Dataset consists of samples from the data distribution

Training requires datasets: To train our algorithms, we need a **dataset**.

Examples:

- Images: Publicly available images from the internet
- Videos: YouTube
- Protein structures: Scientific data (e.g. Protein Data Bank)

A dataset consists of a finite number of samples from the data distribution:

$$z_1, \dots, z_N \sim p_{\text{data}}$$

Conditional Generation allows us to condition on prompts

Data distribution $\boxed{p_{\text{data}}}$

Fixed prompt

"Dog"

Conditional data distribution $\boxed{p_{\text{data}}(\cdot|y)}$

Condition variable: **y**

y="Dog"

y="Cat"

y="Landscape"

Conditional generation means sampling the conditional data distribution:

$$z \sim p_{\text{data}}(\cdot|y)$$

We will first focus on unconditional generation and then learn how to translate an unconditional model to a conditional one.

Generative Models generate samples from data distribution

Initial distribution: $\boxed{p_{\text{init}}}$ ***Default:*** $\boxed{p_{\text{init}} = \mathcal{N}(0, I_d)}$

A generative model converts samples from a initial distribution (e.g. Gaussian) into samples from the data distribution:

$\boxed{x \sim p_{\text{init}}}$ ⇨ Generative Model ⇨ $\boxed{z \sim p_{\text{data}}}$

Summary

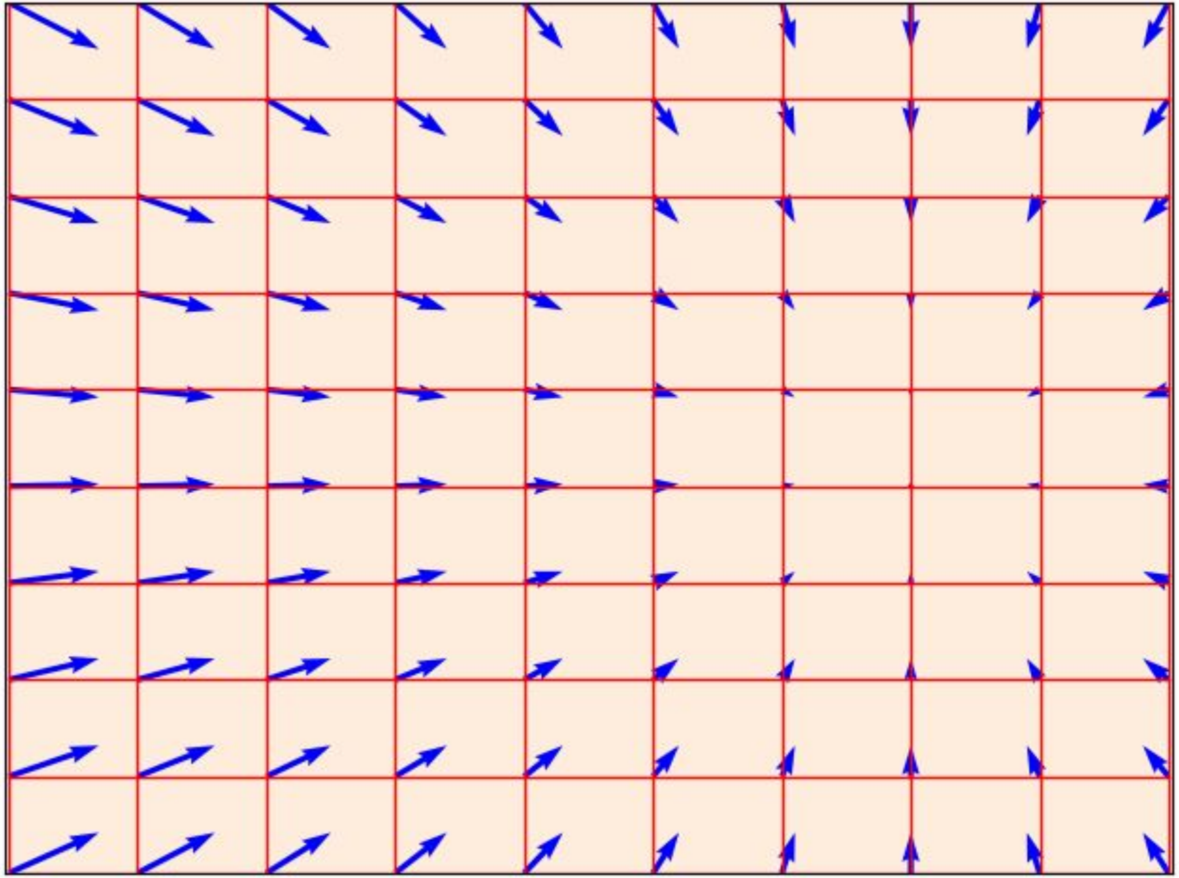
- **Objects to Generate:** We focus on vectors z representing data objects (e.g., images, videos)
- **Data distribution:** Distribution that places higher probability to objects that we consider “good”.
- **Generation as sampling:** generate an object = sampling from the data distribution
- **Dataset**: Finite number of samples from the data distribution used for training
- **Conditional Generation:** Condition on label y and sample from the conditional data distribution
- **Generative Model:** Train a model to transform samples from a simple (e.g., Gaussian) distribution into the data distribution.

Section 2:

Flow and Diffusion Models

Goal: Understand differential equations and how we can build generative models with them.

Flow - Example



Existence and Uniqueness Theorem ODEs

Theorem (Picard–Lindelöf theorem): If the vector field $u_t(x)$ is continuously differentiable with bounded derivatives, then a *unique* solution to the ODE

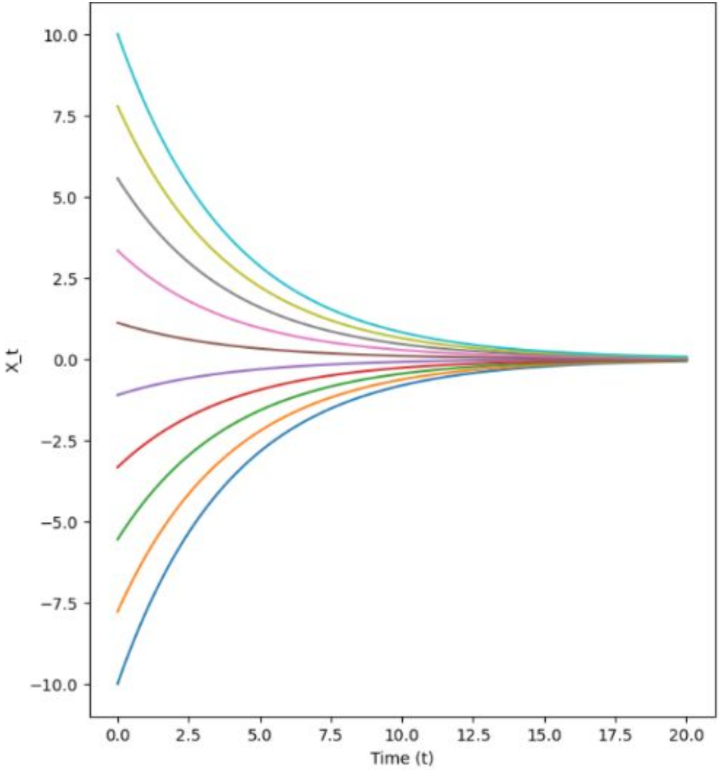
$$X_0 = x_0, \qquad \frac{\mathrm{d}}{\mathrm{d}t} X_t = u_t(X_t)$$

exists. In other words, a flow map exists. More generally, this is true if the vector field is **Lipschitz**.

Key takeaway: In the cases of practical interest for machine learning, **unique solutions to ODE/flows exist**.

Math class: Construct solutions via Picard-Iteration

Example: Linear ODE



Simple vector field:

$$u_t(x) = -\theta x \qquad (\theta > 0)$$

Claim: Flow is given by

$$\psi_t(x_0) = \exp\left(-\theta t\right) x_0$$

Proof:

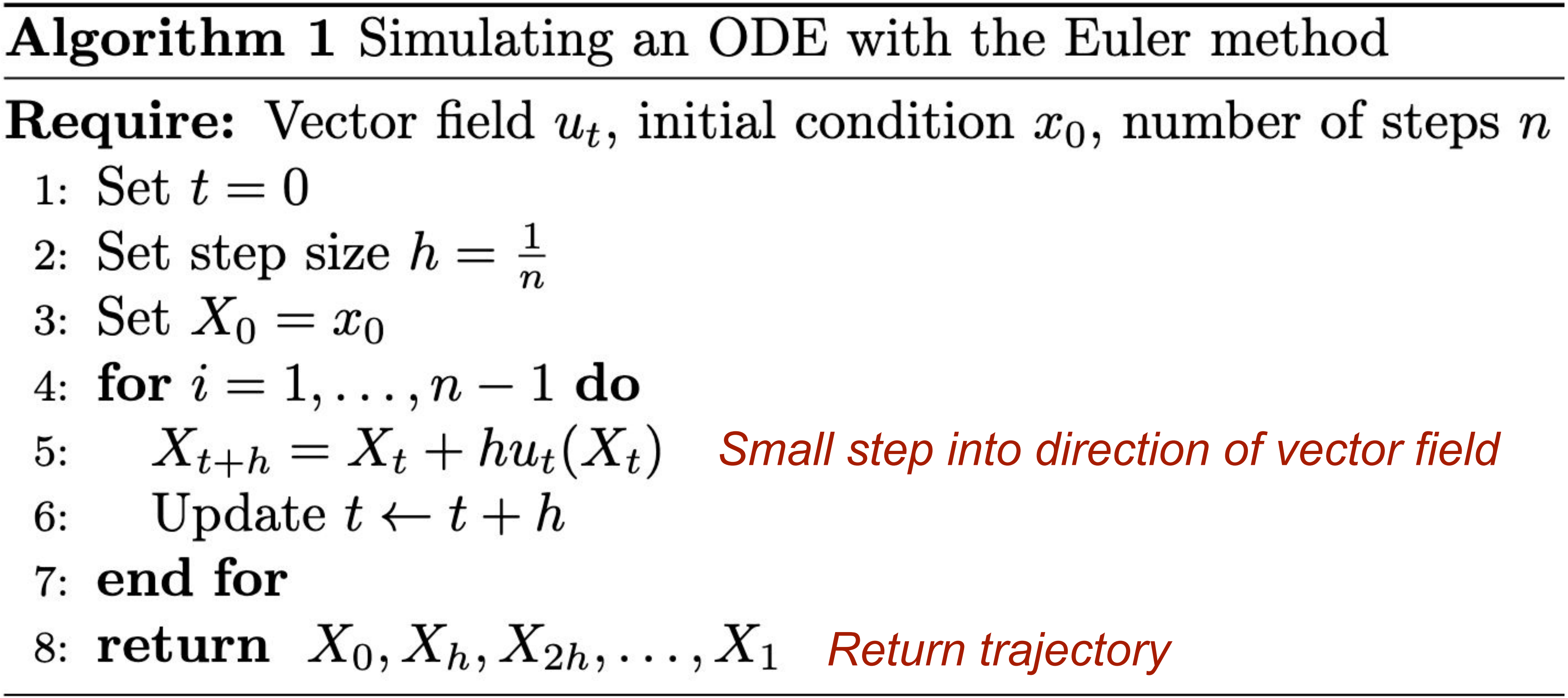
1. Initial condition:

$$\psi_t(x_0) = \exp(0) x_0 = x_0$$

2. ODE:

$$\frac{\mathrm{d}}{\mathrm{d}t}\psi_t(x_0) = \frac{\mathrm{d}}{\mathrm{d}t}\left(\exp\left(-\theta t\right) x_0\right) = -\theta \exp\left(-\theta t\right) x_0 = -\theta \psi_t(x_0) = u_t(\psi_t(x_0))$$

Numerical ODE simulation - Euler method



Algorithm 1 Simulating an ODE with the Euler method

```
Require: Vector field $u_t$, initial condition $x_0$, number of steps $n$
Set $t = 0$
Set step size $h = \frac{1}{n}$
Set $X_0 = x_0$
for $i = 1, \ldots, n-1$ do
    $X_{t+h} = X_t + h u_t(X_t)$
    Update $t \leftarrow t + h$
end for
return $X_0, X_h, X_{2h}, \ldots, X_1$
```

Small step into direction of vector field

Return trajectory

Toy example

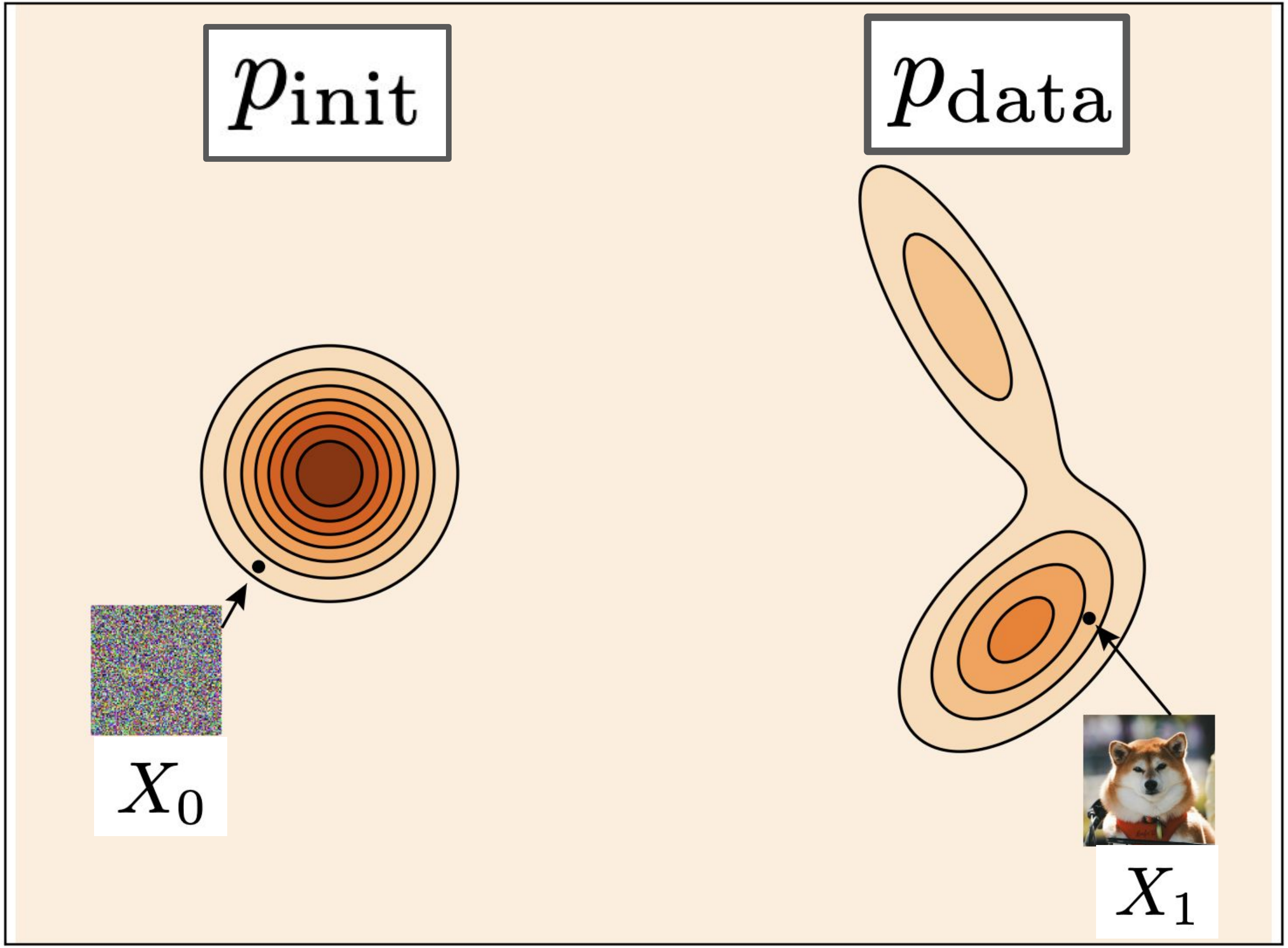


Figure credit: Yaron Lipman

Toy Flow Model

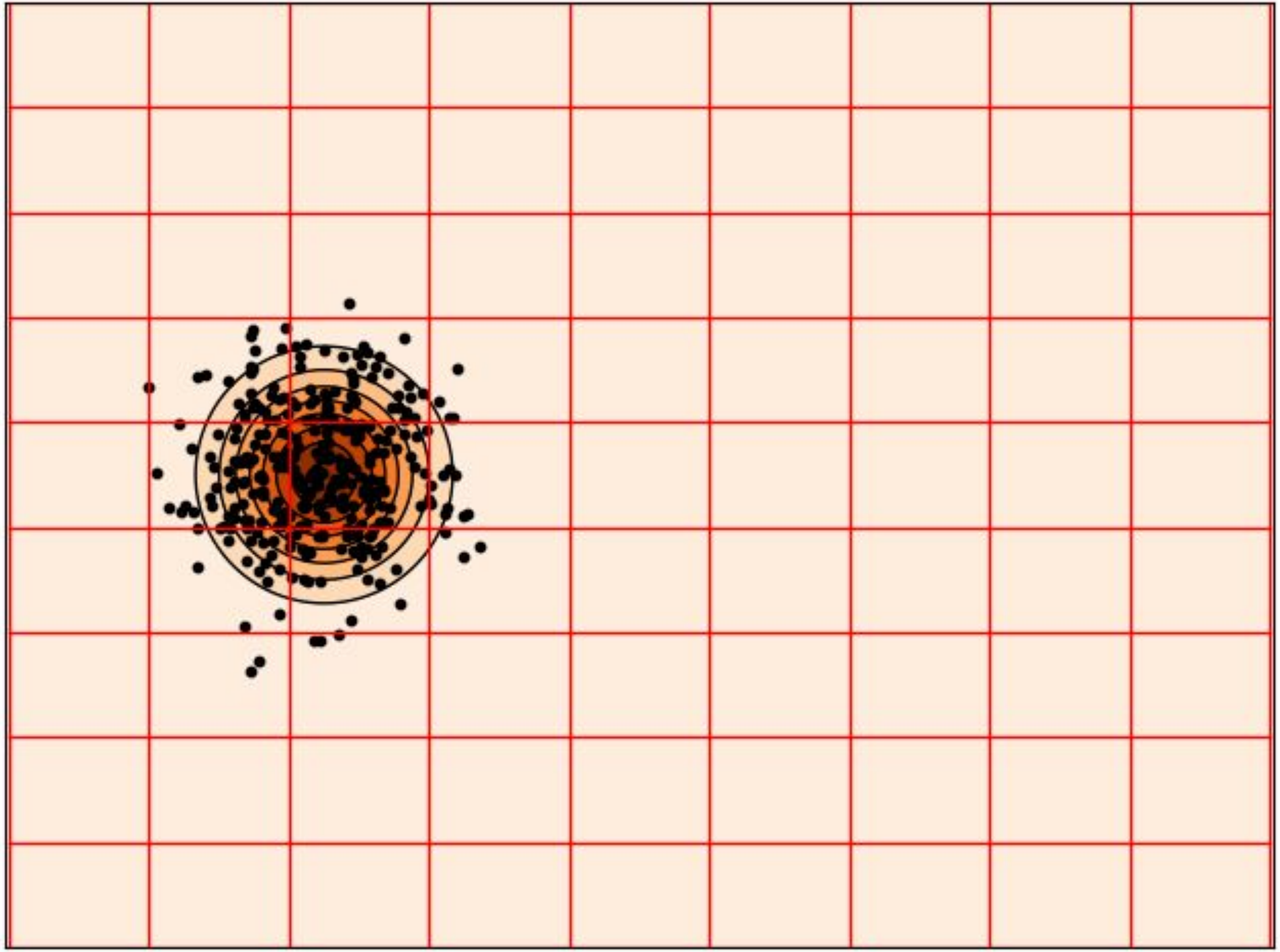
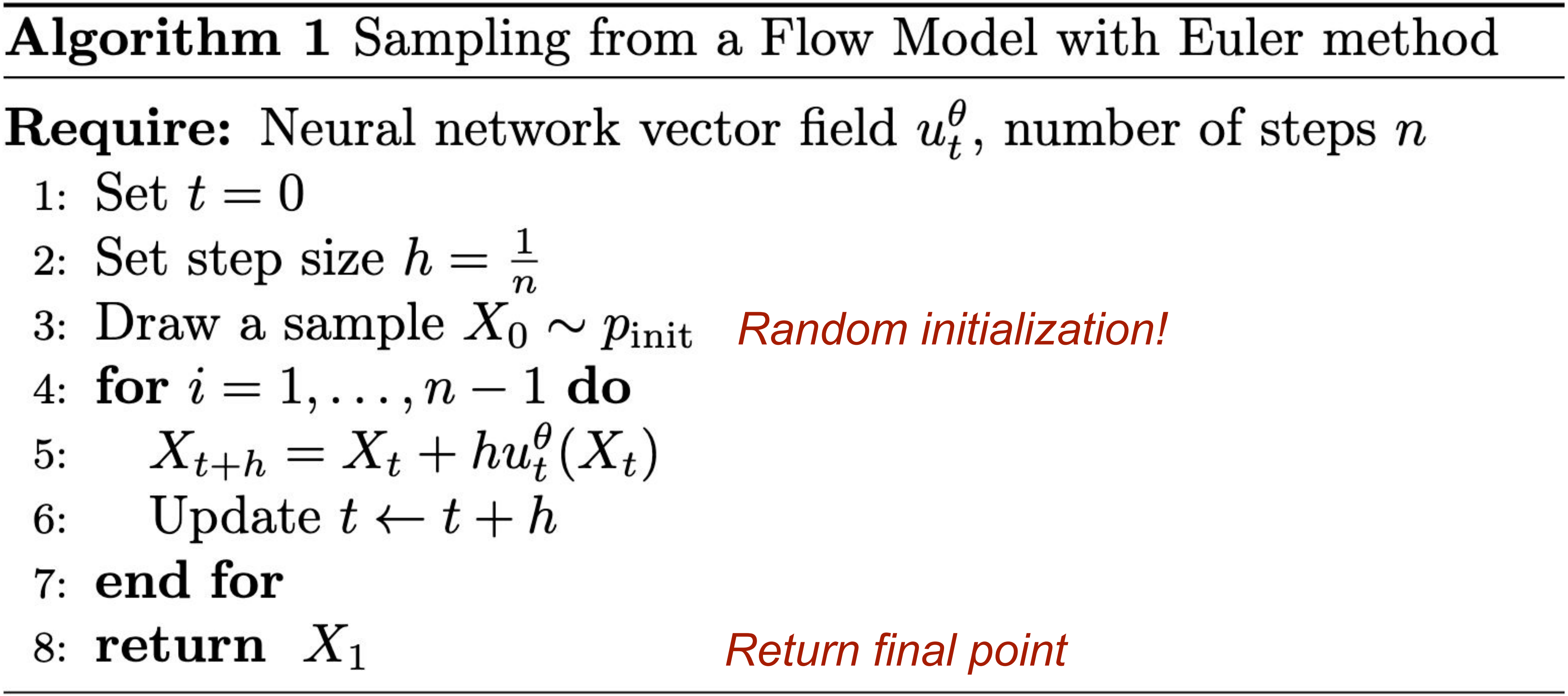


Figure credit: Yaron Lipman

How to generate objects with a Flow Model



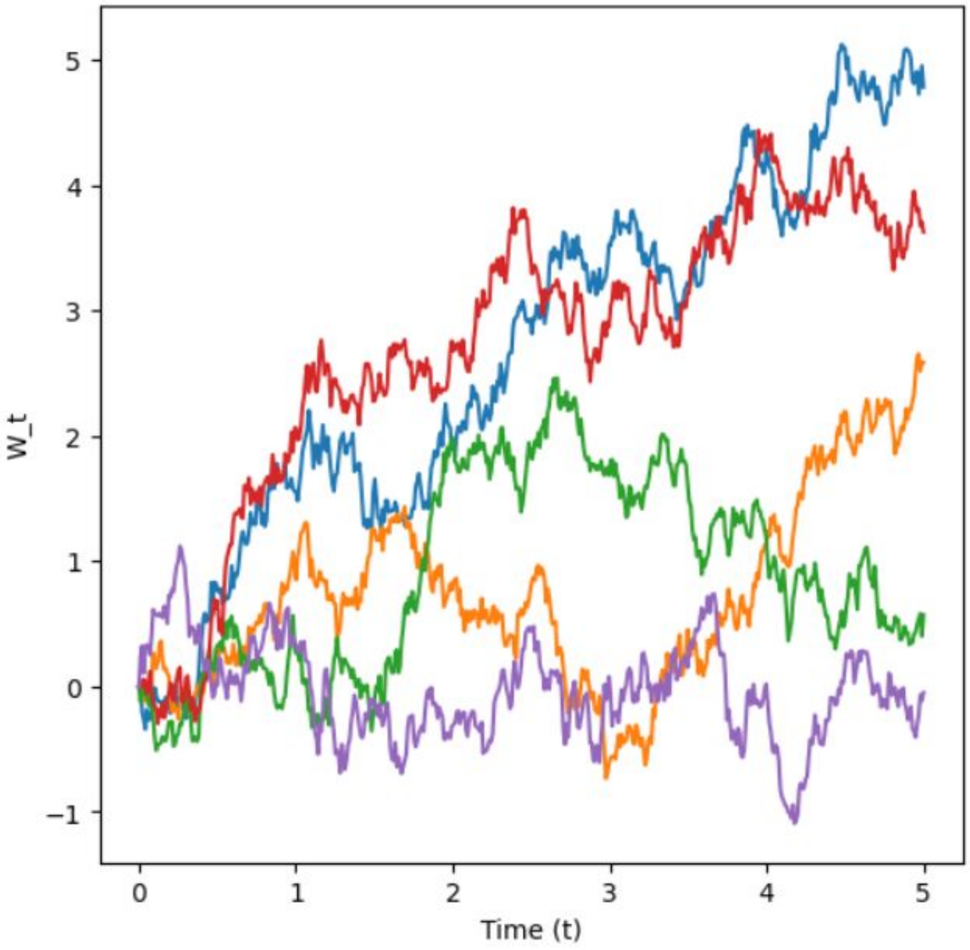
Algorithm 1 Sampling from a Flow Model with Euler method

```
Require: Neural network vector field $u_t^\theta$, number of steps $n$
Set $t = 0$
Set step size $h = \frac{1}{n}$
Draw a sample $X_0 \sim p_{\text{init}}$
for $i = 1, \dots, n-1$ do
    $X_{t+h} = X_t + h u_t^\theta(X_t)$
    Update $t \leftarrow t + h$
end for
return $X_1$
```

Random initialization!

Return final point

Brownian Motion



Existence and Uniqueness Theorem SDEs

Theorem: If the vector field $u_t(x)$ is continuously differentiable with bounded derivatives and the diffusion coeff. is continuous, then a *unique* solution to the SDE

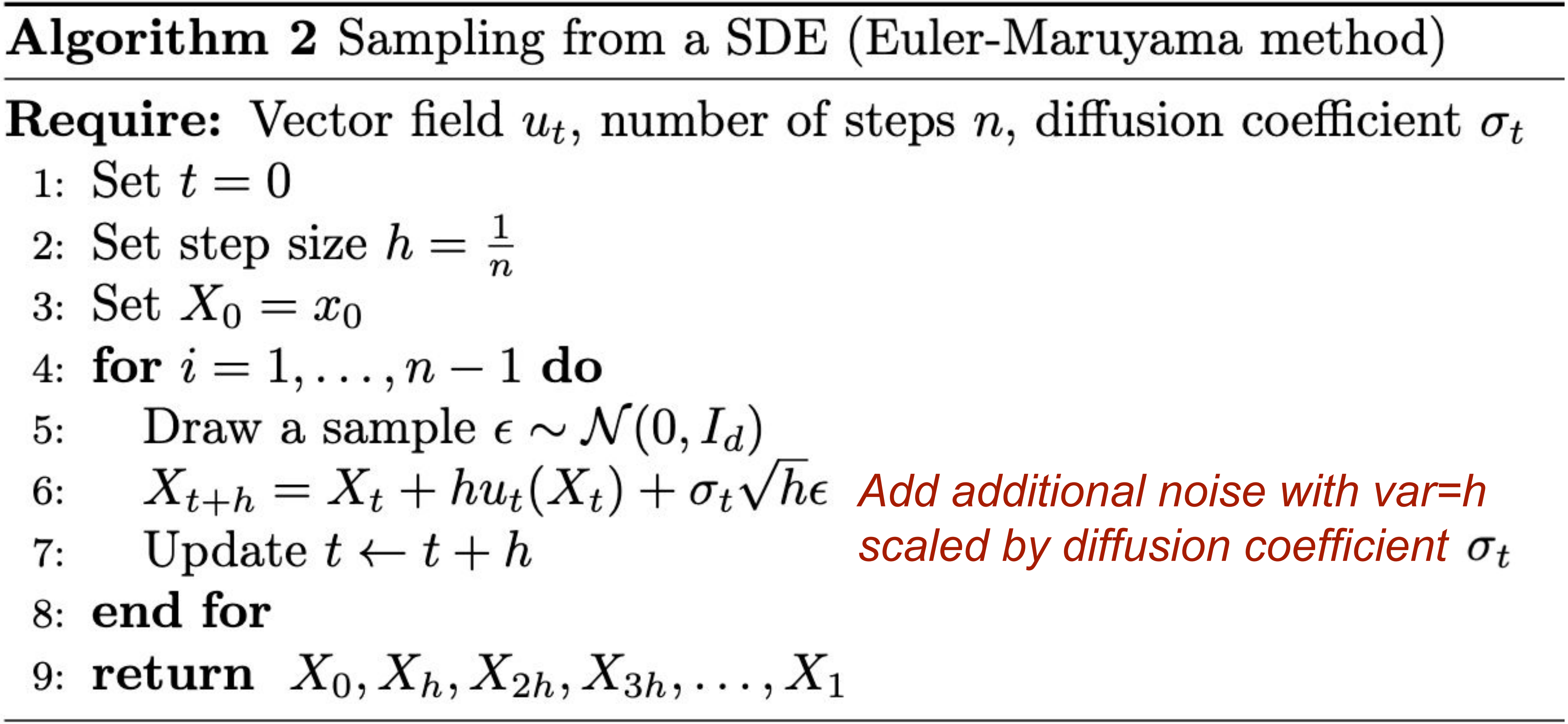
$$X_0 = x_0, \quad \mathrm{d}X_t = u_t(X_t)\mathrm{d}t + \sigma_t \mathrm{d}W_t$$

exists. More generally, this is true if the vector field is **Lipschitz**.

Key takeaway: In the cases of practical interest for machine learning, **unique solutions to SDE**.

Stochastic calculus class: Construct solutions via stochastic integrals and Ito-Riemann sums

Numerical SDE simulation (Euler-Maruyama method)



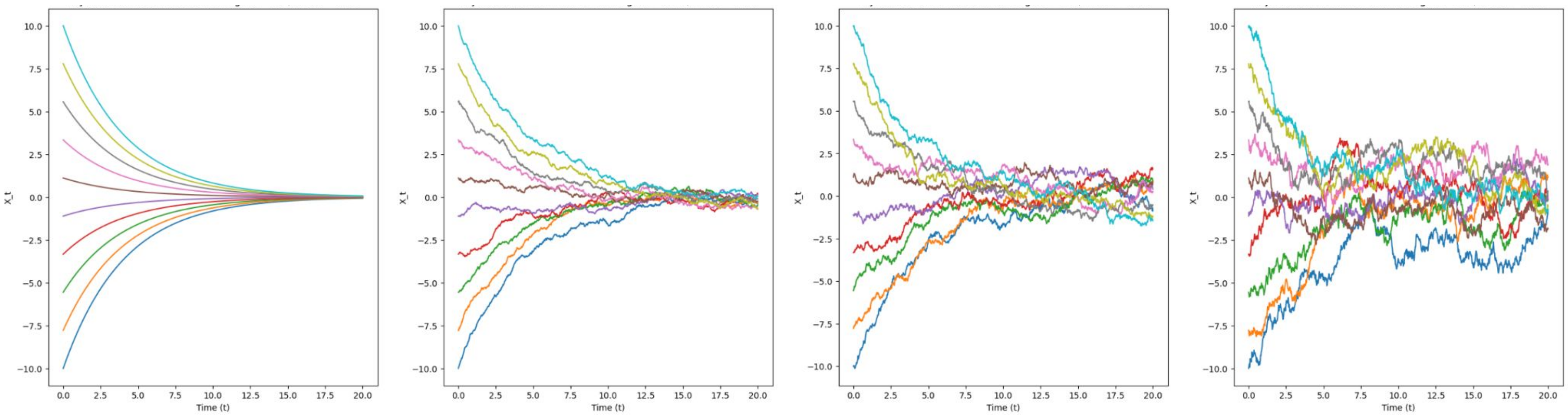
Algorithm 2 Sampling from a SDE (Euler-Maruyama method)

```
Require: Vector field $u_t$, number of steps $n$, diffusion coefficient $\sigma_t$
Set $t = 0$
Set step size $h = \frac{1}{n}$
Set $X_0 = x_0$
for $i = 1, \dots, n - 1$ do
    Draw a sample $\epsilon \sim \mathcal{N}(0, I_d)$
    $X_{t+h} = X_t + hu_t(X_t) + \sigma_t \sqrt{h}\epsilon$
    Update $t \leftarrow t + h$
end for
return $X_0, X_h, X_{2h}, X_{3h}, \dots, X_1$
```

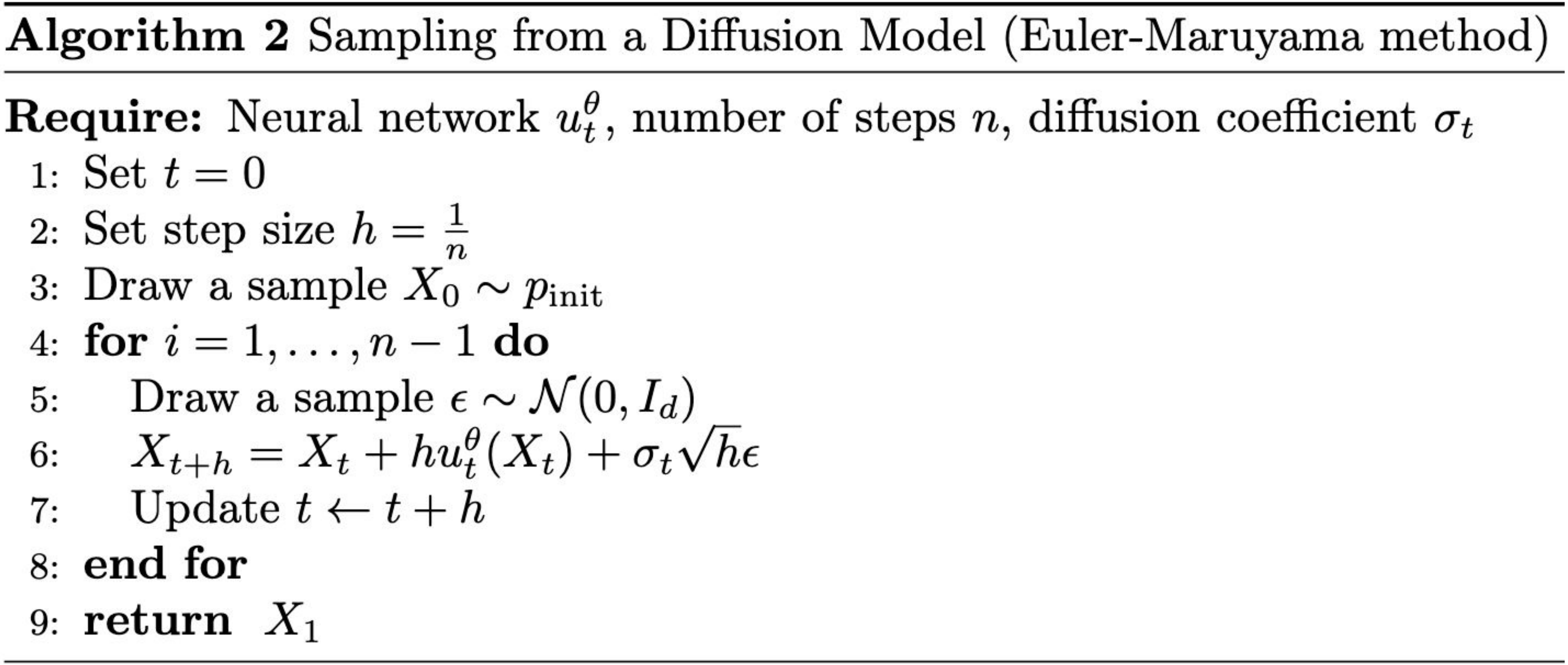
Add additional noise with var=h scaled by diffusion coefficient σ_t

Ornstein-Uhlenbeck Process

$$\mathrm{d}X_t = -\theta X_t \mathrm{d}t + \sigma dW_t$$



Increasing diffusion coefficient σ



Algorithm 2 Sampling from a Diffusion Model (Euler-Maruyama method)

```
Require: Neural network $u_t^\theta$, number of steps $n$, diffusion coefficient $\sigma_t$
Set $t = 0$
Set step size $h = \frac{1}{n}$
Draw a sample $X_0 \sim p_{\text{init}}$
for $i = 1, \ldots, n-1$ do
    Draw a sample $\epsilon \sim \mathcal{N}(0, I_d)$
    $X_{t+h} = X_t + h u_t^\theta(X_t) + \sigma_t \sqrt{h}\epsilon$
    Update $t \leftarrow t + h$
end for
return $X_1$
```

Logistics

Website: https://diffusion.csail.mit.edu/

How to pass this class:

1. Come to lecture
2. **Do the labs (necessary to pass)!**

Support:

1. Use the lecture notes (self-contained)
2. Come to office hours

Lab 1 is out today (see website)! We recommend doing it as soon possible for you!

Next class:

Wednesday (Tomorrow), 11am-12:30pm

E25-111 (same room)

Office hours: Wednesday (tomorrow), 3pm-4:30pm in 37-212