Stochastic Differential Equations models for Least Squares Stochastic Gradient Descent

Editor: Loucas Pillaud-Vivien, Selena Ge

Abstract

This is the final report on the Stochastic Gradient Descent for Least-Square problems. The link with continuous time dynamics such as Stochastic Differential Equations is introduced.

Keywords: Stochastic Gradient Descent, Stochastic Differential Equations, Least-Squares

Contents

1 Set-up: Stochastic Gradient Descent on Least Squares Problems

In this section, we introduce the least squares problem that we consider throughout the article. The stochastic gradient descent is introduced at the end of the section.

1.1 The least square problem: population loss.

We consider a regression problem with random input/output pair $(X, Y) \in \mathbb{R}^d \times \mathbb{R}$ distributed according to the *joint law* ρ. More specifically, the input X is distributed according to a d-dimensional Gaussian vector $\mathcal{N}(0, I_d)$. For the output, we assume that there exists $\theta^* \in \mathbb{R}^d$ and $\xi \sim \mathcal{N}(0, \sigma^2)$ a Gaussian random variable independent of X of mean zero and variance σ^2, such that $Y = \langle \theta^*, X\rangle + \xi$. To learn the rule linking inputs to outputs, we take a linear family of predictors $\{f_\theta : x \mapsto \langle \theta, x\rangle,\ \theta \in \mathbb{R}^d\}$ and aim at minimizing the average *square loss* on the penalty $\ell(\theta, (X, Y)) = \frac{1}{2}\big(\langle \theta, X\rangle - Y\big)^2$

$$L(\theta) := \frac{1}{2}\mathbb{E}_{(X,Y)\sim\rho}\left[\big(\langle \theta, X\rangle - Y\big)^2\right] , \tag{1}$$

where ρ is the joint law of (X, Y).

Lemma 1 *We have for all $\theta \in \mathbb{R}^d$, the equality $L(\theta) = \frac{1}{2}\left(\|\theta - \theta^*\|^2 + \sigma^2\right)$.*

Proof

$$\begin{aligned}
L(\theta) &= \frac{1}{2}\mathbb{E}_{(X,Y)\sim\rho}\left[\big(\langle \theta, X\rangle - Y\big)^2\right] \\
&= \frac{1}{2}\mathbb{E}_{(X)\sim N(0,Id)}\mathbb{E}_{(\xi)\sim N(0,\sigma^2)}\left[\big(\langle \theta, X\rangle - (\langle \theta^*, X\rangle + \xi)\big)^2\right] \\
&= \frac{1}{2}\mathbb{E}_{(X)\sim N(0,Id)}\mathbb{E}_{(\xi)\sim N(0,\sigma^2)}\left[\big(\langle \theta - \theta^*, X\rangle - \xi\big)^2\right] \\
&= \frac{1}{2}\mathbb{E}_{(X)\sim N(0,Id)}\mathbb{E}_{(\xi)\sim N(0,\sigma^2)}\left[\langle \theta - \theta^*, X\rangle^2 - 2\xi\langle \theta - \theta^*, X\rangle + \xi^2\right] \\
&= \frac{1}{2}\mathbb{E}_{(X)\sim N(0,Id)}\left[\langle \theta - \theta^*, X\rangle^2\right] + \frac{1}{2}\mathbb{E}_{(\xi)\sim N(0,\sigma^2)}\left[\xi^2\right], \text{ since X and } \xi \text{ are independent} \\
&= \frac{1}{2}\|\theta - \theta^*\|^2 + \frac{1}{2}\sigma^2, \text{ since } X \text{ is a Gaussian variable, } \mathbb{E}[\langle \theta - \theta^*, X\rangle^2] = \|\theta - \theta^*\|^2 \\
&= \frac{1}{2}\big(\|\theta - \theta^*\|^2 + \sigma^2\big)
\end{aligned}$$

■

Remark 2 *If we knew θ^*, We could use a technique called gradient descent to solve the minimization problem $\min_\theta L(\theta)$.*

1.2 The stochastic gradient descent

The stochastic gradient descent (SGD) aims at minimizing a function through unbiased estimates of its gradient. While this method has been developed for a different purpose in the early 50's (Robbins and Monro, 1951), it is remarkable how SGD fits perfectly the modern large-scale machine learning framework (Bottou et al., 2018). Indeed, the SGD iterative procedure corresponds to sample at each time $t \in \mathbb{N}^*$ an independent draw $(x_t, y_t) \sim \rho$ and update the predictor θ with respect to the local gradient calculated on this sample:

$$\theta_{t+1} = \theta_t - \gamma \nabla_\theta \ell(\theta_t, (x_t, y_t)), \tag{2}$$

where $\gamma > 0$ is the step size. **This method only rests and having access data $((x_1, y_1), \dots, (x_t, y_t))$, this is the reason why it is so popular in data science!**

Remark 3 *$\nabla_\theta \ell(\theta_t, (x_t, y_t))$ is an unbiased estimate of $\nabla_\theta L(\theta_t)$.*[1]

Proof We know that $\ell(\theta, (X, Y)) = \frac{1}{2}\big(\langle \theta, X\rangle - Y\big)^2$. $\nabla_\theta \ell(\theta_t, (x_t, y_t)) = x_t\big(\langle \theta_t, x_t\rangle - y_t\big)$.

$$\begin{aligned}
\mathbb{E}(\nabla_\theta \ell(\theta_t, (x_t, y_t))) &= \mathbb{E}[x_t\big(\langle \theta_t, x_t\rangle - y_t\big)] \\
&= \mathbb{E}[x_t(\langle \theta_t, x_t\rangle - y_t + \xi - \xi)] \\
&= \mathbb{E}[x_t(\langle \theta_t - \theta^*, x_t\rangle - \xi)] \\
&= \mathbb{E}[x_t\langle \theta_t - \theta^*, x_t\rangle] - 2\mathbb{E}(x_t \cdot \xi) \\
&= \mathbb{E}[x_t \cdot x_t^T \cdot (\theta_t - \theta^*)], \text{ since x is independent of } \xi \\
&= \mathbb{E}[x_t \cdot x_t^T](\theta_t - \theta^*) \\
&= \theta_t - \theta^*
\end{aligned}$$

Also, $L(\theta_t) = \frac{1}{2}\mathbb{E}_{(X,Y)\sim\rho}\left[\big(\langle \theta, X\rangle - Y\big)^2\right]$.

$$\begin{aligned}
\nabla_\theta L(\theta_t) &= \frac{1}{2}\mathbb{E}_{(X,Y)\sim\rho}[2X \cdot (\langle \theta_t, X\rangle - Y)] \\
&= \mathbb{E}_{(X,Y)\sim\rho}[X \cdot (\langle \theta_t, X\rangle - \langle \theta^*, X\rangle - \xi)] \\
&= \mathbb{E}_{(X,Y)\sim\rho}[X \cdot (\langle \theta_t - \theta^*, X\rangle - \xi)] \\
&= \mathbb{E}[X \cdot \langle \theta_t - \theta^*, X\rangle] - \mathbb{E}[X \cdot \xi] \\
&= \mathbb{E}[X \cdot X^T \cdot (\theta_t - \theta^*)] \\
&= \mathbb{E}[X \cdot X^T] \cdot (\theta_t - \theta^*), \text{ since x is independent of } \xi \\
&= \theta_t - \theta^*
\end{aligned}$$

Therefore $\mathbb{E}[\nabla_\theta \ell(\theta_t, (x_t, y_t))] = \nabla_\theta L(\theta_t) = \theta_t - \theta^*$ ■

1. We say that a random variable X is an unbiased estimate of a vector x, if $\mathbb{E}(X) = x$

Online SGD. The iteration Eq. (2) corresponds to what is refer to as *online SGD* in the literature. In this case, for each time $t \in \mathbb{N}^*$, we have an independent draw $(x_t, y_t) \sim \rho$.

Lemma 4 *The explicit form of the derivation of Eq.* (2) *for least-squares is*

$$\theta_{t+1} = \theta_t - \gamma x_t(\langle \theta_t, x_t \rangle - y_t)$$

Proof We take $\nabla_\theta \ell(\theta_t, (x_t, y_t)) = x_t\big(\langle \theta_t, x_t \rangle - y_t\big)$ in Eq. (2). ■

A convenient way to present these dynamics is to force the apparition of the true gradient and rewrite the rest as a noise (or martingale increment). Indeed

Lemma 5 *The SGD recursion reads*

$$\theta_{t+1} = \theta_t - \gamma(\theta_t - \theta_*) + \gamma m(\theta_t, (x_t, y_t)) \ , \tag{3}$$

where $m(\theta_t, (x_t, y_t)) := \mathbb{E}_\rho\left[(\langle \theta_t, X \rangle - Y)\, X\right] - (\langle \theta_t, x_t \rangle - y_t)\, x_t$.

Proof From lemma 4, we have $\theta_{t+1} = \theta_t - \gamma x_t(\langle \theta_t, x_t \rangle - y_t)$.
The first term in $m(\theta_t, (x_t, y_t))$ is the gradient of $L(\theta_t)$ which is equal to $\theta_t - \theta^*$.
Therefore $m(\theta_t, (x_t, y_t)) = \theta_t - \theta^* - x_t(\langle \theta_t, x_t \rangle - y_t)$.

$$\begin{aligned}
\theta_t - \gamma(\theta_t - \theta^*) + \gamma m(\theta_t, (x_t, y_t)) &= \theta_t - \gamma(\theta_t - \theta^*) + \gamma[\theta_t - \theta^* - x_t(\langle \theta_t, x_t \rangle - y_t)] \\
&= \theta_t - \gamma x_t(\langle \theta_t, x_t \rangle - y_t) \\
&= \theta_t - \gamma x_t(\langle \theta_t, x_t \rangle - y_t) \\
&= \theta_{t+1}
\end{aligned}$$

■

Lemma 6 *In equation* (3) *the variable* $m(\theta_t, (x_t, y_t))$ *is centered.*[2]

Proof From Lemma 5 we know that

$$\begin{aligned}
m(\theta_t, (x_t, y_t)) &= \mathbb{E}_\rho\left[(\langle \theta_t, X \rangle - Y)\, X\right] - (\langle \theta_t, x_t \rangle - y_t)\, x_t \\
&:= \mathbb{E}_\rho(Z) - Z, \text{ if we define } Z := X(\langle \theta_t, X \rangle - Y)
\end{aligned}$$

Then $\mathbb{E}(m) = \mathbb{E}(\mathbb{E}(Z)) - E(Z) = E(Z) - E(Z) = 0$.
$\Rightarrow$ m is centered. ■

2. We say that a random variable X is centered if $E(X) = 0$.

1.3 Simulations

We use the equation of Lemma 4 to write a Python code that simulates the SGD dynamics until time $t = 1000$, with step-size $\gamma = 0.01$, initialization $\theta_0 = \mathbf{0}$, the zero vector, $\theta^* = [0.1, -0.2, 1, 0.5, -0.5]$ and $\sigma = 0.1$.

(i) Display the test error curve upon time $\|\theta_t - \theta_*\|^2$ for several runs of the dynamics (meaning different data).

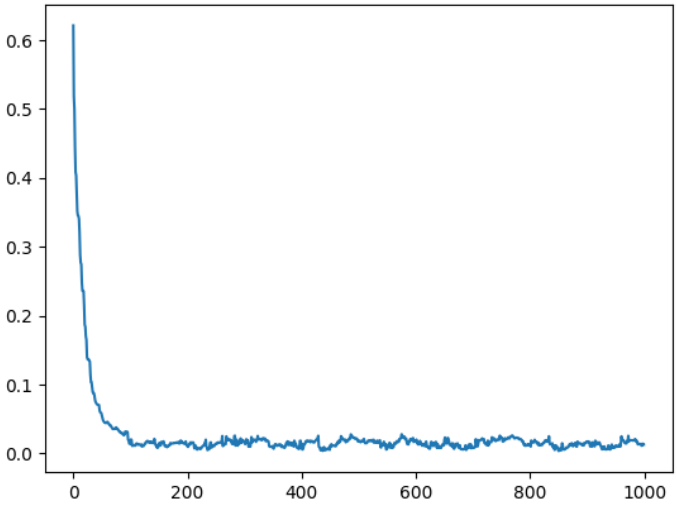

Figure 1: Simulation of test error curve

The test error drops to approximately 0 around t=200. This means the SGD dynamics become convergent around t=200.

(ii) Display the two first coordinates of $(\theta_t)_t$ as well as the ones of θ^*.

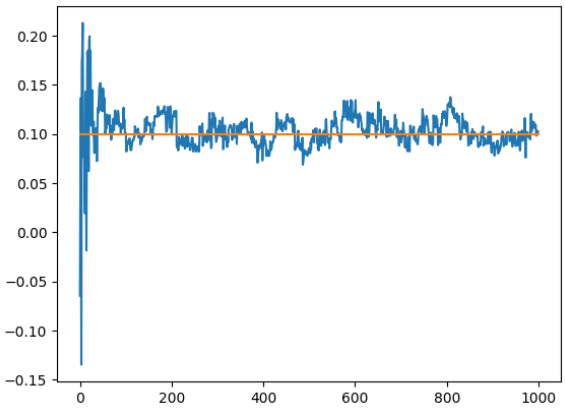

Figure 2: Simulation of θ_1

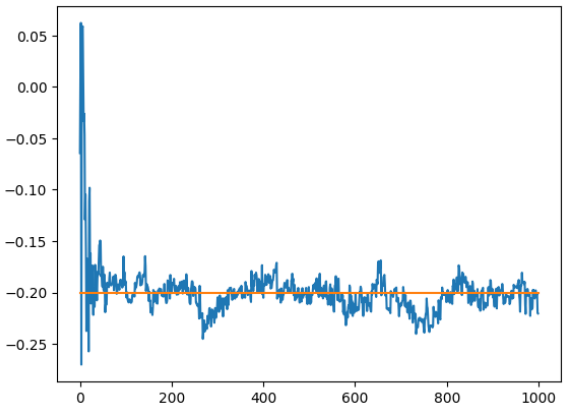

Figure 3: Simulation of θ_2

(iii) Change the variance σ to 1 or even 3: it fluctuates more when σ become higher.

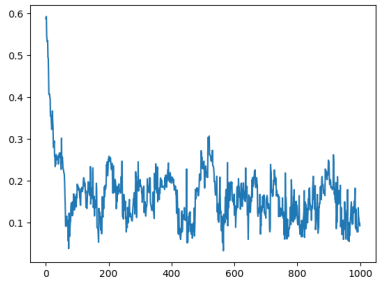

Figure 4: Test Error when $\sigma = 1$

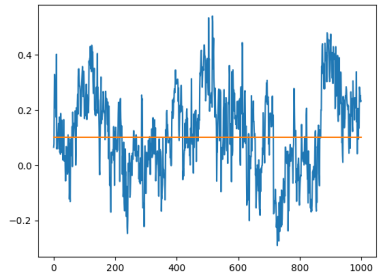

Figure 5: Simulation of θ_1 when $\sigma = 1$

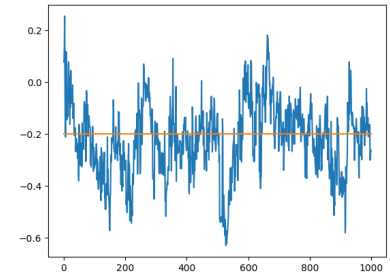

Figure 6: Simulation of θ_2 when $\sigma = 1$

(iv) When the step size is 10 times bigger, it explodes since the gradient changes too far each time.

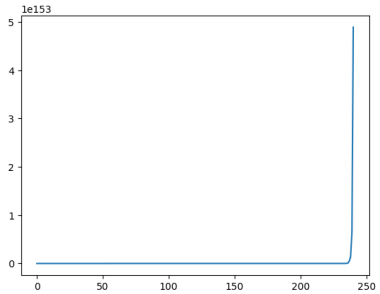

Figure 7: Test Error when step-size bigger

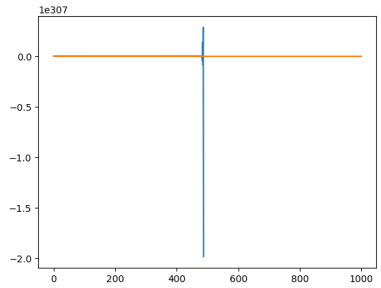

Figure 8: θ_1 when step-size bigger

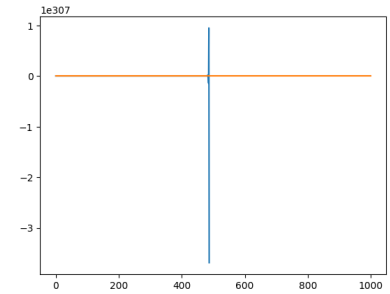

Figure 9: θ_2 when step-size bigger

When the step size is 10 times smaller, it moves slowly but we can see that it becomes more accurate after convergence.

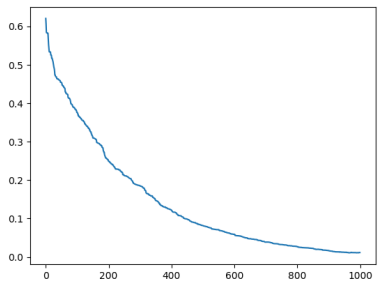

Figure 10: Test Error when step-size smaller

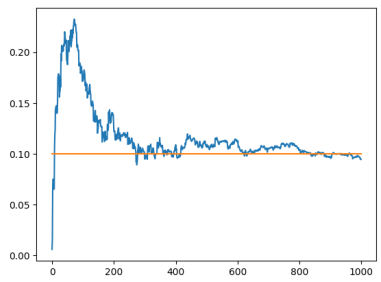

Figure 11: θ_1 when step-size smaller

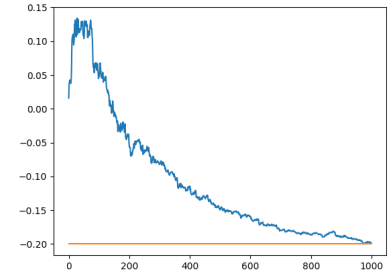

Figure 12: θ_2 when step-size smaller

(v) Change the step size to make it depend on the iterations: $\gamma = 0.1/t$. The simulation balanced between accuracy and velocity.

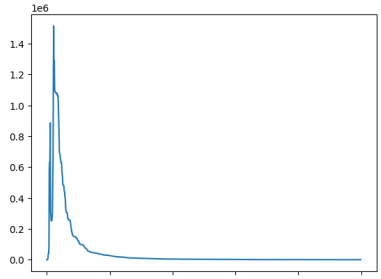

Figure 13: Test Error: Improved

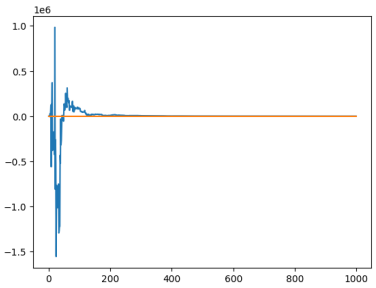

Figure 14: θ_1: Improved

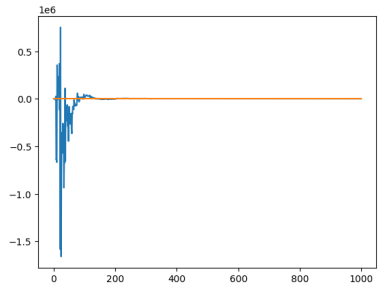

Figure 15: θ_2: Improved

2 Continuous model of SGD

In this section we give stochastic differential equations (SDE) models for SGD. For the first time, we provide general necessary conditions to model well SGD. We instantiate them more precisely in a second time.

2.1 The requirement of a SDE model

As said above, the decomposition of the SGD recursion as in equations (3) is generic feature of the stochastic descent (Benaïm, 2006), as it has led to the celebrated *ODE method* (Harold et al., 1997) to study stochastic approximation of this type. Going further, this decomposition between *a drift* ∇L and *local martingale term* $m(\cdot,(x,y))$ is reminiscent of the decomposition occurring for Itô processes, i.e. solution to an SDE of the type

$$d\theta_t = b(t,\theta_t)dt + \sigma(t,\theta_t)dB_t, \tag{4}$$

where $(B_t)_{t\geq 0}$ is a Brownian motion of $\mathbb{R}^d$. These types of models have been largely studied in the literature (Khasminskii, 2011), as, beyond their large modeling abilities, they offer useful tools, e.g. Itô calculus, when it comes to mathematical analysis. One of the aims of this article is to link the SGD dynamics to processes that are exemplary in the SDE literature. In order to establish this link, we first have to answer the following question

What SDE model fits well with the SGD dynamics?

This question has received a lot of attention in the last decade, and a good principle to answer this is to turn to *stochastic modified equations* (Li et al., 2019). This is a natural way to build models of SDE since they are consistent in the infinitesimal step-size limit with SGD. In order to build such a model, there are two requirements

Remark 7 *After reading (Li et al., 2019, Section 3.1), we conclude that*

(i) The drift term $b(t,\theta_t)$ should match $[-\nabla L(\theta_t)]$.

(ii) The noise factor σ should have the same covariance as $[\gamma\nabla L(\theta_t)]$, i.e.

$$[\gamma\mathbb{E}[(\nabla L(\theta_t) - \nabla l(\theta_t))(\nabla L(\theta_t) - \nabla l(\theta_t)^T|\theta_t].]$$

Proof From equation (1), we know that

$$L(\theta) = \frac{1}{2}E_{(X,Y)\sim\rho}((<\theta,x> - Y)^2)$$

Its gradient has already been computed before in Lemma 3, which is:

$$\nabla_\theta L(\theta_t) = (\theta_t - \theta^*)Id = \begin{pmatrix} \theta_t - \theta^* \\ \vdots \\ \theta_t - \theta^* \end{pmatrix} \in \mathbb{R}^d,$$

From equation(3) in the tutorial we have:

$$\begin{aligned}\theta_{t+1} - \theta_t &= -\gamma(\theta_t - \theta^*) + \gamma m(\theta_t, (x_t, y_t)) \\ &= -\gamma \nabla L(\theta_t) + \gamma(\mathbb{E}_\rho((< \theta_t, x_t > -y_t)x_t) - (< \theta_t, x_t > -y_t)x_t) \\ &= -\gamma \nabla L(\theta_t) + \gamma(\nabla L(\theta_t) - \nabla l(\theta_t))\end{aligned}$$

where $l(\theta) = \frac{1}{2}(< \theta_t, x_t > -y_t)^2$.
Considering equation (4) in the tutorial, which is:

$$d\theta_t = b(t, \theta_t)dt + \sigma(t, \theta_t)dB_t$$

If we apply the Euler-Maruyama discretization with step-size γ, approximating $X_{k\gamma}$ by $\hat{X}_k$, we obtain the following discrete iteration:

$$\hat{\theta_{t+1}} - \hat{\theta}_t = \gamma b(t, \hat{\theta}_t) + \sqrt{\gamma}\sigma(t, \hat{\theta}_t)Z_k \tag{5}$$

where $Z_k := B_{(k+1)\gamma} - B_{k\gamma}$ are d-dimensional i.i.d standard normal random variables.
Matching equation (5) with the previous deduction of equation (3), we know that:
(i) The drift term $b(t, \theta_t)$ should match $-\nabla L(\theta_t)$.
(ii) The noise factor σ should have the same covariance as $\gamma \nabla L(\theta_t)$, i.e. $\gamma\mathbb{E}[(\nabla L(\theta_t) - \nabla l(\theta_t))(\nabla L(\theta_t) - \nabla l(\theta_t)^T|\theta_t]$. This means $\sigma(t, \theta_t) = \sqrt{\gamma\Sigma(\theta)}$.
Then $d\theta_t = -\nabla L(\theta_t)dt + \sqrt{\gamma\Sigma(\theta)}dB_t$. ■

Besides technical assumptions, these are the two requirements presented in Li et al. (2019, Theorem 3) to show that the SDE model is *consistent* in the small step-size limit with the SGD recursion.

2.2 Explicit form of the SDE models

Let us first write explicitly the multiplicative noise factor $\sigma(t, \theta_t)$ in the population case. Then, we derive the expression of the SDE models that we analyze later. In the population case, the calculation has already been made in Ali et al. (2020) and, defining $r_X(\theta) = \langle \theta_t, X \rangle - Y \in \mathbb{R}$, the residual random variable, then we have that

$$\sigma(t, \theta_t)\sigma(t, \theta_t)^\top = \gamma \left(\mathbb{E}_\rho \left[r_X(\theta_t)^2 X X^\top \right] - \mathbb{E}_\rho \left[r_X(\theta_t) X \right] \mathbb{E}_\rho \left[r_X(\theta_t) X \right]^\top \right).$$

Let us simplify the covariance and assume that for all $\theta \in \mathbb{R}^d$,

$$\sigma(\theta) := \sqrt{\gamma\sigma^2} I_d. \tag{6}$$

We hence study the SDE.

Lemma 8 *The derivation of the SDE with the simplification given by Eq.* (6) *is:*

$$d\theta_t = -\nabla L(\theta_t)dt + \sqrt{\gamma\Sigma(\theta)}dB_t = (-(\theta_t - \theta^*)Id)dt + (\sqrt{\gamma\sigma^2}Id)dB_t$$

Lemma 9 *Each coordinate of θ_t satisfy an Ornstein-Uhlenbeck with parameters that can be specified.*

Proof In the SDE model, we have:

$$d\theta_t = (-(\theta_t - \theta^*)Id)dt + (\sqrt{\gamma\sigma^2}Id)dB_t \tag{7}$$

To show that each coordinate of θ_t satisfies an Ornstein-Uhlenbeck process, we need to match each parameter in this SDE with the standard form of an Ornstein-Uhlenbeck process:

$$d\theta_t = \kappa(\theta - \theta_t)dt + \sigma dW_t \tag{8}$$

Matching equation (7) and equation (8), we have:

- $\kappa = 1$.
- $\theta = \theta^*$, the long-term mean of the process matches θ^*.
- $\sigma = \sqrt{\gamma\sigma^2}$, the volatility term matches the noise factor.

■

For concise information on the Ornstein-Uhlenbeck process, visit the following link: https://planetmath.org/ornsteinuhlenbeckprocess.

Remark 10

$$\textit{The mean of the process is } \mathbb{E}(\theta_t) = \theta^* + (\mathbb{E}(\theta_0) - \theta^*)e^{-t}. \tag{9}$$

$$\textit{The variance of the process is } Var(X_t) = \frac{\gamma\sigma^2}{2}(1 - e^{-2t}) \tag{10}$$

Proof

$$d\mathbb{E}(\theta_t) = \kappa(\theta - \mathbb{E}(\theta_t))dt$$

This is an ODE in $\mathbb{E}(\theta_t)$ which can be solved to give:

$$\mathbb{E}(\theta_t) = \theta^* + (\mathbb{E}(\theta_0) - \theta^*)e^{-t}$$

The variance of the process can be found by solving another differential equation. The variance is given by:

$$Var(\theta_t) = \mathbb{E}(\theta_t^2) - \mathbb{E}(\theta_t)^2$$

To find $\mathbb{E}(\theta_t^2)$, we use Ito's formula and the fact that $dW_t^2 = dt$ to set up an ODE for it:

$$\frac{d}{dt}\mathbb{E}(\theta_t^2) = 2\kappa\mathbb{E}(\theta_t) - 2\kappa\mathbb{E}(\theta_t^2) + \sigma^2$$

Solving the ODE and plugging in the corresponding parameters gives:

$$Var(X_t) = \frac{\gamma\sigma^2}{2}(1 - e^{-2t})$$

■

Remark 11 *The process converges to Gaussian Distribution with mean θ^* and variance $\frac{\gamma\sigma^2}{2}$, since the mean reversion term represents a force that pulls the process back towards the mean θ^* when θ_t deviates from it.*

2.3 Simulation for the OU process through the Euler-Maruyama method

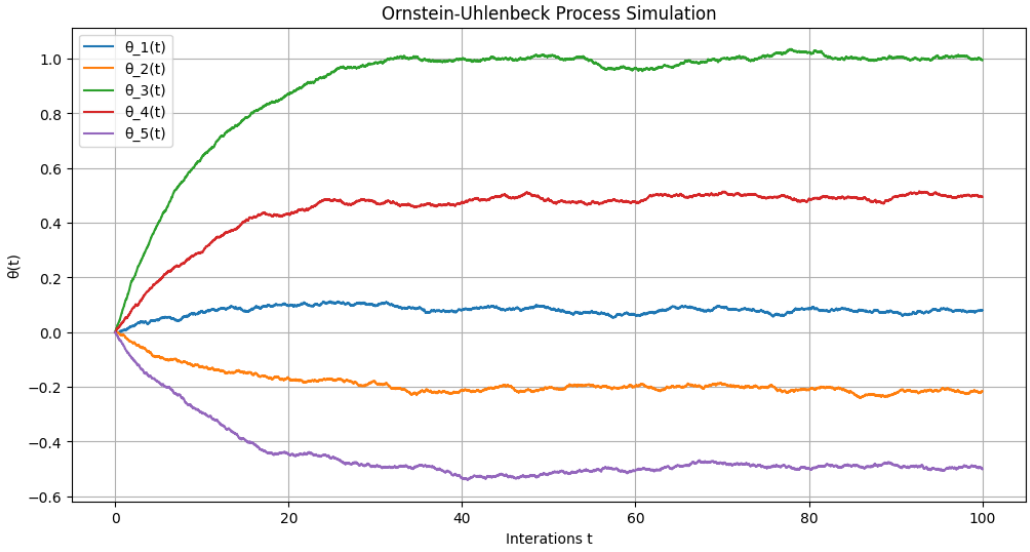

Figure 16: Simulation of OU process

3 Appendix

3.1 Stochastic Differential Equations

1. Definition [**Continuous-time stochastic process**]: A set of random variables X_t indexed by real numbers $t \geq 0$. Each realization of the stochastic process is a choice from the random variable X_t for each t, and is, therefore, a function of t. Any (deterministic) function $f(t)$ can be trivially considered as a stochastic process, with variance $V(f(t)) = 0$.
2. Definition[**Wiener process** W_t]: the formalization of Brownian Motion which defines the scaling limit of random walks as the step size and time interval between steps both goes to zero. Usually used to represent random, external influences on an otherwise deterministic system.
    - For each t, the random variable W_t is normally distributed with **mean 0** and **variance t**. i.e. $W_t - W_s \sim N(0, t-s)$.
    - For each $t_1 < t_2$, the normal random variable $W_{t_2} - W_{t_1}$ is **independent** of the random variable W_{t_1}, and in fact **independent** of all W_t, $0 \leq t \leq t_1$.
    - The Wiener process W_t can be represented by **continuous** paths.
3. Definition [**Diffusion process**]: a differential equation involving deterministic, or **drift terms**, and stochastic, or **diffusion terms**.
   The first term below is the drift term, and the latter term is the diffusion term, where X represents the stochastic process we are interested in, and W represents the Wiener process:
   $$dX = a(t, X)dt + b(t, X)dW_t \tag{11}$$
    - $a(t, X)dt$: drift term because it captures the average or expected rate of change of the process X if no randomness was involved.
    - $b(t, X)dW_t$: diffusion term because it scales the magnitude of the randomness by the increment of W.

   Eq (11) is given in differential form, which is continuous but not differentiable. The meaning of Eq (11) is, by definition:
   $$X(t) = X(0) + \int_0^t a(s, y)ds + \int_0^t b(s, y)dW_s$$
4. Method to solving Eq (11) [**Ito's formula**]: the chain rule for stochastic differentials introduced to find the differential of a function of a stochastic process.
   If $Y = f(t, X)$, then:
   $$dY = \frac{\partial f}{\partial t}(t, X)dt + \frac{\partial f}{\partial x}(t, X)dx + \frac{1}{2}\frac{\partial^2 f}{\partial x^2}(t, X)dxdx$$
   where the dx dx term is interpreted by using the identities:
   $$\begin{aligned} dtdt &= 0 \\ dtdW_t = dW_t dt &= 0 \\ dW_t dW_t &= dt \end{aligned}$$

Therefore after combining Eq(11) with it, we get:

$$df(t, X_t) = (\frac{\partial f}{\partial t} + a(t, X)\frac{\partial f}{\partial x} + \frac{1}{2}b^2(t, X)\frac{\partial^2 f}{\partial x^2})dt + b(t, X)\frac{\partial f}{\partial x}dW_t$$

5. Let $c = t_0 < r_1 < ... < t_{n-1} < t_n = d$ be a grid of points on the interval [c,d].

   - **[Riemann integral]**: $\int_c^d f(x)dx = \lim_{\Delta t \to 0} \sum_{i=1}^n f(t_i')\Delta t_i$, where $\Delta t_i = t_i - t_{i-1}$ and $t_{i-1} \leq t_i' \leq t_i$. Here t_i' may be chosen at any point in the interval (t_{i-1}, t_i).
   - **[Ito's integral]**: $\int_c^d f(t)dW_t = \lim_{\Delta t \to 0} \sum_{i=1}^n f(t_{i-1})\Delta W_i$, where $\Delta W_i = W_{t_i} - W_{t_{i-1}}$. Here t_i' must be the left endpoint of (t_{i-1}, t_i). Since f and W_t are random variables, so is the Ito's integral $I = \int_c^d f(t)dW_t$, thus:

     $$I = \int_c^d f dW_t$$

     is expressed in differential form as

     $$dI = f dW_t$$

     where dW_t is called white noise. A typical solution is a combination of drift and the diffusion of Brownian motion.

6. Example [**Black-Scholes DE**]: with constant μ and σ

$$\begin{cases} dX = \mu X dt + \sigma X dW_t \\ X(0) = X_0 \end{cases} \tag{12}$$

The solution is geometric Brownian Motion:

$$X(t) = X_o e^{(\mu - \frac{1}{2}\sigma^2)t + \sigma W_t}$$

To check this, write $X = f(t, Y) = X_0 e^Y$, where $Y = (\mu - \frac{1}{2}\sigma^2)t + \sigma W_t$. By Ito's formula:

$$dX = X_0 e^Y dY + \frac{1}{2}e^Y dY dY$$

Where $dY = (\mu - \frac{1}{2}\sigma^2)dt + \sigma dW_t$. Using the differential identities from the Ito formula,

$$dY dY = \sigma^2 dt$$

and therefore

$$\begin{aligned} dX &= X_0 e^Y (\mu - \frac{1}{2}\sigma^2)dt + X_0 e^Y \sigma dW_t + \frac{1}{2}\sigma^2 e^Y dt \\ &= X_0 e^Y \mu dt + X_0 e^Y \sigma dW_t \\ &= \mu X dt + \sigma X dW_t \end{aligned}$$

3.2 Numerical Methods for SDE

1. Method of solving equation (11) [**Euler-Maruyama method**]: To develop an approximate solution on the interval [c,d], assign a grid of points $c = t_0 < t_1 < t_2 < ... < t_n = d$. Approximate x values $\omega_0 < \omega_1 < \omega_2 < ... < \omega_n$ will be determined at the respective t points. Given the SDE initial value problem

$$\begin{cases} dX(t) = a(t, X)dt + b(t, X)dW_t \\ X(c) = X_c \end{cases}$$

we compute the approximate solution as follows:

$$\begin{aligned} \omega_0 &= X_0 \\ \omega_{i+1} &= \omega_i + a(t_i, \omega_i)\Delta t_{i+1} + b(t_i, \omega_i)\Delta W_{i+1} \end{aligned}$$

where

$$\begin{aligned} \Delta t_{i+1} &= t_{i+1} - t_i \\ \Delta W_{i+1} &= W(t_{i+1}) - W(t_i) \end{aligned}$$

How to model ΔW_i: Define N(0,1) to be the standard random variable that is normally distributed with mean 0 and standard deviation 1. Each random number ΔW_i is computed as

$$\Delta W_i = z_i\sqrt{\Delta t_i}$$

where z_i is chosen from N(0,1).

In the first example, we show how to apply the Euler-Maruyama method to the **Black-Scholes SDE**(12). The Euler-Maruyama method (11) have form

$$\begin{aligned} \omega_0 &= X_0 \\ \omega_{i+1} &= \omega_i + \mu\omega_i\Delta t_i + \sigma\omega_i\Delta W_i \end{aligned}$$

As another example,consider the **Langevin equation**:

$$dX(t) = -\mu X(t)dt + \sigma dW_t$$

where μ and σ are positive constants. The solution of the Langevin equation is a stochastic process called the **Ornstein-Uhlenbeck process**. It was generated from an Euler-Maruyama approximation, using the steps:

$$\begin{aligned} \omega_0 &= X_0 \\ \omega_{i+1} &= \omega_i - \mu\omega_i\Delta t_i + \sigma\Delta W_i \end{aligned}$$

for i =1, ...,n.

3.3 Empirical risk minimization(ERM)

$$\min_{x\in\mathbb{R}^d} f(x) := \mathbb{E}_\gamma\left[f_\gamma(x)\right]$$

where $f_\gamma : r \in \Gamma$ is a family of functions from $\mathbb{R}^d$ to $\mathbb{R}$ and γ is a Γ-valued random variable w.r.t which the expectation is taken. **Stochactis gradient algorithms(SGA)** are often used to solve optimation problems of ERM.
Solving EMR using the **standard gradient descent(GD)** on x gives the iteration scheme. First, let us define the gradient of f as for all $x = (x_1, \dots, x_d) \in \mathbb{R}^d$, for all $i = 1 \dots d$,

$$\nabla f(x) = \begin{pmatrix} \partial_{x_1} f(x_1, \dots, x_d) \\ \vdots \\ \partial_{x_d} f(x_1, \dots, x_d) \end{pmatrix} \in \mathbb{R}^d,$$

Then we have the recursion

$$x_{k+1} = x_k - \eta\nabla f(x_k) = x_k - \eta\nabla\mathbb{E}_\gamma\left[f_\gamma(x_k)\right] \tag{13}$$

for $k \geq 0$ and η is a small positive step-size known as the **learning rate**.
In its simplest form, the **stochastic gradient descent(SGD)** algorithm replaces the expectation of the gradient with a sampled gradient, i.e.

$$x_{k+1} = x_k - \eta\nabla f_{\gamma_k}(x_k) \tag{14}$$

where each γ_k is an i.i.d random variable with the same distribution as γ. We then have $\mathbb{E}[\nabla f_{\gamma_k}(x_k)|(x_k)] = \nabla\mathbb{E}f(x_k)$. So (14) is a sampled version of (13)

3.4 Stochastic Modified Equations

$$\frac{dx}{dt} = -\nabla f(x)$$

3.4.1 Heuristic motivations

1. rewrite equation (14) as

$$\begin{aligned} x_{k+1} &= x_k - \eta\nabla f_{\gamma_k}(x_k) \\ &= x_k - \eta\nabla f_{\gamma_k}(x_k) + \eta\nabla f(x_k) - \eta\nabla f(x_k) \\ &= x_k - \eta\nabla f(x_k) + \underbrace{\eta(\mathbb{E}_\gamma\left[f_{\gamma_k}(x_k)\right] - \nabla f_{\gamma_k}(x_k))}_{\sqrt{\eta}V_k} \end{aligned}$$

This means

$$x_{k+1} = x_k - \eta\nabla f(x_k) + \sqrt{\eta}V_k(x_k, \gamma_k) \tag{15}$$

where $V_k(x_k, \gamma_k) = \sqrt{\eta}(\nabla f(x_k) - \nabla f_{\gamma_k}(x_k))$ is a d-dimensional random vector. A straightforward calculation shows that:

$$\begin{aligned} \mathbb{E}[V_k|x_k] &= 0 \\ \mathrm{Cov}[V_k|x_k] &= \eta\Sigma(x_k) := \mathbb{E}[(\nabla f_{\gamma_k}(x_k) - \nabla f(x_k))(\nabla f_{\gamma_k}(x_k) - \nabla f(x_k))^T|x_k] \end{aligned}$$

i.e. conditional on x_k, $V_k(x_k)$ has 0 mean and covariance $\eta\sum(x_k)$. Here $\sum$ is simply the conditional covariance of the stochastic gradient approximation ∇f_γ or ∇f.

2. Now consider a time-homogeneous Ito stochastic differential equation (SDE) of the form

$$dX_t = b(X_t)dt + \sqrt{\eta}\sigma(X_t)dW_t$$

where $X_t \in \mathbb{R}^d$ for $t \geq 0$ and W_t is a standard d-dimensional Wiener process. The function b: $\mathbb{R}^d \to \mathbb{R}^d$ is known as the drift term, and $\sigma : \mathbb{R}^d \to \mathbb{R}^{d\times d}$ is the diffusion matrix. If we apply the Euler-Maruyama discretization with step-size η, approximating $X_{k\eta}$ by $\hat{X_k}$, we obtain the following discrete iteration:

$$\hat{X_{k+1}} = \hat{X_k} + \eta b(\hat{X_k}) + \eta\sigma(\hat{X_k})Z_k$$

where $Z_k := W_{(k+1)\eta} - W_{k\eta}$ are d-dimensional i.i.d standard normal random variables. Comparing with equation(15), if we set $b = -\nabla f$, $\sigma(x) = \Sigma(x)^{\frac{1}{2}}$, and identify t with $k\eta$, we then have matching first and second conditional moments. Hence, this motivates the approximating equation

$$dX_t = -\nabla f(X_t)dt + (\eta\Sigma(X_t))^{\frac{1}{2}}dW_t$$

Note that as this heuristic argument shows, the presence of the small parameter $\sqrt{\eta}$ on the diffusion term is necessary to model the fact that when the learning rate decreases, the fluctuations to the SGA iterates must also decrease.

3.5 Ornstein-Uhlenbeck process

3.5.1 Definition

The Ornstein-Uhlenbeck process is a stochastic process that satisfies the following SDE:

$$dX_t = \kappa(\theta - X_t)dt + \sigma dW_t \tag{16}$$

where W_t is a standard Brownian motion on $t \in [0, \infty)$. The constant parameters are:

- $\kappa > 0$ is the rate of mean reversion;
- θ is the long-term mean of the process;
- $\sigma > 0$ is the volatility or average magnitude, per square-root time, of the random fluctuations that are modeled as Brownian motions.

3.5.2 Mean-reverting property

If we ignore the random fluctuations in the process due to dW_t, then we see that X_t has an overall drift towards a mean value θ. The process X_t reverts to this mean exponentially, at rate κ, with a magnitude in direct proportion to the distance between the current value of X_t and θ.
This can be seen by looking at the solution to the ordinary differential equation $dx_t = \kappa(\theta - x)dt$ which is:

$$\frac{\theta - x_t}{\theta - x_0} = e^{-\kappa(t-t_0)}, \text{ or } x_t = \theta + (x_0 - \theta)e^{-\kappa(t-t_0)}$$

For this reason, the Ornstein-Uhlenbeck process is also called a mean-reverting process, although the latter name applies to other types of stochastic processes exhibiting the same property as well.

3.5.3 Solution

The solution to the stochastic differential equation (16) defining the Ornstein-Uhlenbeck process is, for any $0 \leq s \leq t$, is:

$$X_t = \theta + (X_s - \theta)e^{-\kappa(t-s)} + \sigma \int_s^t e^{-\kappa(t-u)} dW_u$$

where the integral on the right is the Itô integral.
For any fixed s and t, the random variable X_t, conditional upon X_s, is normally distributed with:

$$mean = \theta + (X_s - \theta)e^{-\kappa(t-s)}$$

$$variance = \frac{\sigma^2}{2\kappa}(1 - e^{-2\kappa(t-s)})$$

Observe that the mean of X_t is exactly the value derived heuristically in the solution of the ODE. The Ornstein-Uhlenbeck process is a time-homogeneous Itô diffusion.

3.6 Python code for simulations

Listing 1: Simulation code for section 1.3

```
import matplotlib.pyplot as plt
import numpy as np
import random

class least_square():
    def __init__(self,t:int,gamma:float,gamma_on_iteration:bool,
       theta:list,theta_star:list,sigma:float) -> None:
        self.t = t
        self.gamma = gamma
        self.on_iteration = gamma_on_iteration
        self.theta = theta
        self.theta_star = theta_star
        self.sigma = sigma
        self.temp_step = 0
        self.data = []
        self.err = []
        self.theta_1 = []
        self.theta_2 = []

    def generate_data(self):
        for _ in range (self.t):
            piece_data = []
            y = 0
            for dimension in self.theta_star:
                x = random.random()*5 #assume the x is scatter in
                    range (0,5)
                piece_data.append(x)
                y = y + x * dimension
            y = y + random.gauss(0,self.sigma) #y = \theta^* \cdot x
                + \xi where \xi ~ N(0,\sigma)
            self.data.append([piece_data,y])
        print("finish generating %d pieces of data" %(self.t))

    def _inner_product(self,x1:list,x2:list) -> float:   # np.dot(x1
       ,x2) is better.
        if(len(x1)!=len(x2)):
            print("dimension not match with the first %d,and the
               second %d" %(len(x1),len(x2)))
        else:
            result = 0
            result = np.dot(x1,x2)
        return result

    def _2_norm(self,x1:list,x2:list) -> float: # 0.5 * np.linalg.
       norm(x1 - x2)
        if(len(x1)!=len(x2)):
```

```
            print("dimension not match with the first %d,and the
                second %d" %(len(x1),len(x2)))
        else:
            result = 0
            # print(x1)
            # print(x2)
            # print(x1-x2)
            result = 0.5 * np.linalg.norm(x1-x2) #Check if there is
                a square missing
            # print(result)
            return result

    def _update_theta(self,coefficient:float,x:list):
        for i in range(len(self.theta)):
            self.theta[i] = self.theta[i] - coefficient * x[i]

    def iter(self):
        if(self.on_iteration):
            for i in range (self.t):
                x = self.data[i][0]
                y = self.data[i][1]
                product = self._inner_product(theta,x)
                coefficient = 2 * self.gamma/(i+1) * (product - y)
                self._update_theta(coefficient,x)
                self.err.append(self._2_norm(theta,theta_star))
                self.theta_1.append(self.theta[0])
                self.theta_2.append(self.theta[1])

        else:
            for piece in self.data:
                x = piece[0]
                y = piece[1]
                #theta_{t+1} = theta_t - 2gamma(<theta_t,x_t>-y_t)
                    x_t, in particular, 2 is always ignored.
                product = self._inner_product(theta,x)
                coefficient = 2 * self.gamma * (product - y)
                self._update_theta(coefficient,x)
                self.err.append(self._2_norm(theta,theta_star))
                self.theta_1.append(self.theta[0])
                self.theta_2.append(self.theta[1])
                #print(coefficient)
                #print(theta)


    def show_err(self):
        plt.plot(self.err)
        plt.show()

    def show_theta1(self):
        theta_1_star = [self.theta_star[0] for _ in range(self.t)]
        plt.plot(self.theta_1)
```

```
        plt.plot(theta_1_star)
        plt.show()

    def show_theta2(self):
        theta_2_star = [self.theta_star[1] for _ in range(self.t)]
        plt.plot(self.theta_2)
        plt.plot(theta_2_star)
        plt.show()


#(i) and (ii)
if __name__ == '__main__':
    theta = np.zeros(5) #[0. ,0. ,0.,0. ,0.]
    theta_star = [0.1,-0.2,1,0.5,-0.5]
    my_ls = least_square(1000,0.01,False,theta,theta_star,0.1)
    my_ls.generate_data()
    print(len(my_ls.data)) #1000
    my_ls.iter()
    print(my_ls.theta)
    my_ls.show_err()
    my_ls.show_theta1()
    my_ls.show_theta2()

#(iii) change to 1: it fluctuates more
if __name__ == '__main__':
    theta = np.zeros(5)
    theta_star = [0.1,-0.2,1,0.5,-0.5]
    my_ls = least_square(1000,0.01,False,theta,theta_star,1)
    my_ls.generate_data()
    print(len(my_ls.data)) #1000
    my_ls.iter()
    print(my_ls.theta)
    my_ls.show_err()
    my_ls.show_theta1()
    my_ls.show_theta2()

#(iii) change to 3: it fluctuates more and more
if __name__ == '__main__':
    theta = np.zeros(5)
    theta_star = [0.1,-0.2,1,0.5,-0.5]
    my_ls = least_square(1000,0.01,False,theta,theta_star,3)
    my_ls.generate_data()
    print(len(my_ls.data)) #1000
    my_ls.iter()
    print(my_ls.theta)
    my_ls.show_err()
    my_ls.show_theta1()
    my_ls.show_theta2()

#(iv) change to 10 times bigger: it explodes
if __name__ == '__main__':
```

```
    theta = np.zeros(5)
    theta_star = [0.1,-0.2,1,0.5,-0.5]
    my_ls = least_square(1000,0.1,False,theta,theta_star,0.1)
    my_ls.generate_data()
    print(len(my_ls.data)) #1000
    my_ls.iter()
    print(my_ls.theta)
    my_ls.show_err()
    my_ls.show_theta1()
    my_ls.show_theta2()

#(iv) change to 10 times smaller: it moves slowly
if __name__ == '__main__':
    theta = np.zeros(5)
    theta_star = [0.1,-0.2,1,0.5,-0.5]
    my_ls = least_square(1000,0.001,False,theta,theta_star,0.1)
    my_ls.generate_data()
    print(len(my_ls.data)) #1000
    my_ls.iter()
    print(my_ls.theta)
    my_ls.show_err()
    my_ls.show_theta1()
    my_ls.show_theta2()

#(v)
if __name__ == '__main__':
    theta = np.zeros(5)
    theta_star = [0.1,-0.2,1,0.5,-0.5]
    my_ls = least_square(1000,0.5,True,theta,theta_star,0.1)
    my_ls.generate_data()
    print(len(my_ls.data)) #1000
    my_ls.iter()
    print(my_ls.theta)
    my_ls.show_err()
    my_ls.show_theta1()
    my_ls.show_theta2()
```

Listing 2: Simulation for section 2.3

```
import numpy as np
import matplotlib.pyplot as plt

# Parameters for the OU process
theta_star = np.array([0.1, -0.2, 1, 0.5, -0.5])  # Long-term mean
sigma = 0.1  # Standard deviation
gamma = 0.01  # Step-size
theta_0 = np.zeros(len(theta_star))  # Initial condition (zero
    vector)
t_final = 100  # Final time
dt = 0.01  # Time step

# Time steps
timesteps = int(t_final / dt)

# Initialize the array to store the theta values
theta = np.zeros((timesteps + 1, len(theta_star)))
theta[0] = theta_0

# Simulate the OU process
for t in range(1, timesteps + 1):
    dW = np.sqrt(dt) * np.random.normal(0, 1, len(theta_star))  #
        Brownian motion
    theta[t] = theta[t-1] + 0.1 * (theta_star - theta[t-1]) * dt +
        np.sqrt(gamma * sigma**2/2) * dW #Equation from Ex10

# Plotting the result
plt.figure(figsize=(12, 6))
for i in range(len(theta_star)):
    plt.plot(np.linspace(0, t_final, timesteps + 1), theta[:, i],
        label=f" _ {i+1}(t)")

plt.title("Ornstein-Uhlenbeck␣Process␣Simulation")
plt.xlabel("Interations␣t")
plt.ylabel("  (t)")
plt.legend()
plt.grid(True)
plt.show()
```

References

Alnur Ali, Edgar Dobriban, and Ryan Tibshirani. The implicit regularization of stochastic gradient flow for least squares. In *International conference on machine learning*, pages 233–244. PMLR, 2020.

Michel Benaïm. Dynamics of stochastic approximation algorithms. In *Seminaire de probabilites XXXIII*, pages 1–68. Springer, 2006.

Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

J Harold, G Kushner, and George Yin. Stochastic approximation and recursive algorithm and applications. *Application of Mathematics*, 35, 1997.

Rafail Khasminskii. *Stochastic stability of differential equations*, volume 66. Springer Science & Business Media, 2011.

Qianxiao Li, Cheng Tai, and E Weinan. Stochastic modified equations and dynamics of stochastic gradient algorithms i: Mathematical foundations. *The Journal of Machine Learning Research*, 20(1):1474–1520, 2019.

H. Robbins and S. Monro. A stochastic approximation method. *Ann. Math. Statistics*, 22: 400–407, 1951.