1 Brownian Motion and the Wiener Process

Consider a stochastic process, where per each iteration, we add a Δ to the previous value, this Δ is distributed by a normal distribution with a mu of zero, and these are independent stochastic events this is called the Wiener process. In discrete time, we can define the Wiener process with the following equation

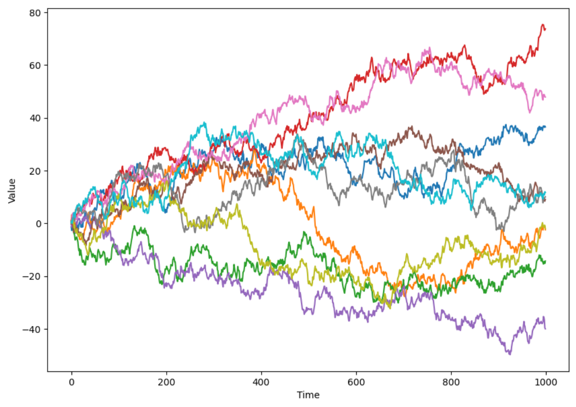
$$W_{t+1} = W_t + \mathcal{N}(0, \sigma^2)$$

Where W_{t+1} is the next value of the process (under discrete time), $\mathcal{N}(0, \sigma^2)$ is a normally distributed drift variable with the variance σ^2.

Intuitively, the expected value, i.e. the limit of this process as it approaches infinity, should be zero, since the normal distribution is symmetric. (maybe prove this? Using a limits on the $\mathbb{E}[W]$ and the normal dist)

The spread of the Wiener process grows at a rate of $\sigma_t = \sigma\sqrt{t}$, due to each timestep adding additional compounded uncertainty.

Here's an example of a Wiener process with $W_0 = 0$ and $\sigma^2 = 1$:



In the diagram, we can clearly observe that the the variance of the Wiener process gets larger as the time progresses.

The wiener process is crucial and has many financial applications, for example, equity prices under the assumption of the efficient market hypothesis follow a purely-stochastic Wiener process. This although disregards stochastic drift, which applies directional pressure on equity prices, i.e., the Wiener process is market-agnostic, and applies to equity which do not follow the upward trend of the market due to the positive risk-free rate.

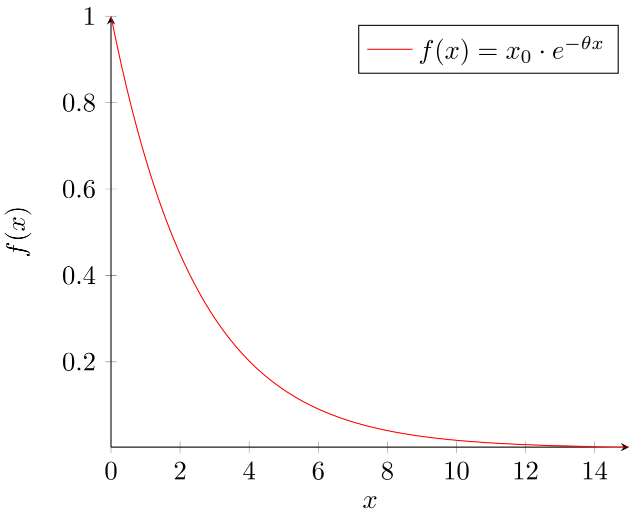
2 Ornstein-Uhlenbeck Process

2.1 Deriving a Mean Reversion Mechanism

But what if we don't want the variance of the process to get larger as time progresses? We call these processes mean-reverting, as in they tend to revert their values back towards the mean μ. This has numerous financial applications (not to mention its applications outside of finance), for example: statstical arbitrage, interest rate models, heston model, and mean reverting equities.

So how do we implement mean reversion? We can start by thinking about what mean reversion actually is? Mean reversion is the tendency of a variable (or usually a stochastic process) to revert back to its long-term mean. If we consider the roots of brownian motion - physics -, we can consider a mean-reverting term to be similar to a force pushing a particle back towards its mean; note that this doesn't happen instantaneuously, it happens over a period of time, where the steadyness of this decay is dictated by the rate of mean-reversion, denoted as θ.

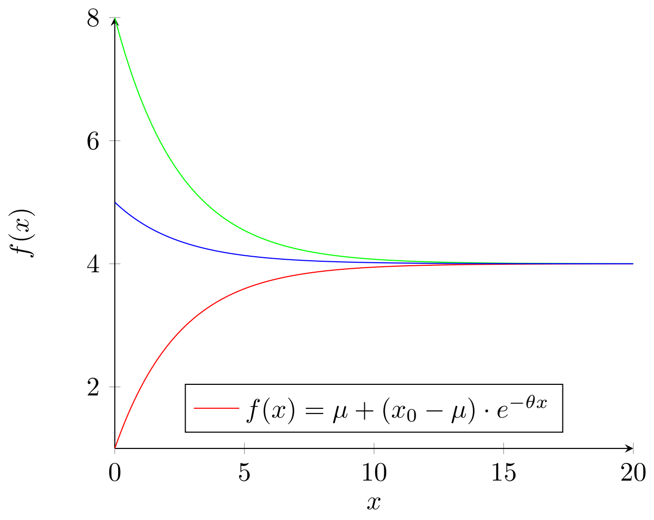
We know that an exponential function with a negative coefficient applied to the exponent will produce a function that will slowly decay.



But this is not exactly mean reverting, we need to adjust the function to not only have the tendency to approach zero, but rather, the function must approach the mean. And the function must be applied from any starting point x_0. Considering that $f(0) = x_0$, we can adjust the function and introduce a base term: $f(x) = x_0 e^{-\theta x}$, this ensures the initial value is x_0, since $e^0 = 1$. Now we have to ensure that the function converges at the mean, not at zero. Initially, we can try to add a the mean as a constant term, such that $f(x) = \mu + x_0 e^{-\theta x}$, but when graphing this function, we notice that we break our first condition - the initial value of the function $f(0)$ is not equal to x_0. We have to adjust for this in the coefficient of the exponential term. We notice that the difference between x_0 and $f(0)$ is equal to the mean, thus modifying the function to the following will solve our problem:

$$f(x) = \mu + (x_0 - \theta)e^{-\theta x}$$

The function produces the following graph, assuming $x_0 = 1$, $\theta = 0.4$ and varying the initial value x_0 to be $x_0 \in \{8, 5, -1\}$ for each of the separate lines respectivelly:



2.2 Differentiating for the Mean-reverting Force

We can now think about creating a stochastic process that will take the Wiener process as a basis, but add mean-reverting properties in order to ensure the variance stays within a certain range. Just adding a term for the brownian motion will not be enough, this would be misinterpreting the mean-reverting effect and would be mostly useless for our purposses, thus we can't do something like:

$$X_t = \mu + (X_0 - \mu) \cdot e^{-\theta t} + \sigma W_t$$

We must now calculate the tug or the actual amount by which the function changes per unit t. This will be necessary for our stochastic process, as they are inherently defined in differential form (in descrete time), where the next value depends on the previous value and we simply add a modifying term, e.g. $M_{t+1} = M_t + \sigma$. We can do this by taking a simple derivative of the function. This derivative can then be thought of as the per-unit t amount of tug applied for each value above or below the mean.

$$\frac{d}{dt}\left(\mu + (X_0 - \mu) \cdot e^{-\theta t}\right)$$
$$\frac{d}{dt}\mu + \frac{d}{dt}\left(X_0 - \mu\right) e^{-\theta t}$$

We know that the derivative of the constant μ is zero, thus we can eliminate the term. We are left with the derivative of a product, for which we can use the product rule of differentiation: $\frac{d}{dt} x \cdot y = x'y + xy'$. In our case, the two terms are $(X_0 - \mu)$ and $e^{-\theta t}$. The first term is a constant that does not chage with respect to t, thus it will be zero. The second term can be differentiated using the chain rule.

We need to define an outer and an inner function in order to apply the chain rule, the inner function will be e^x and the outer function will be $-\theta t$. The $\frac{d}{dx} e^x = e^x$ is a standard differential. And the $\frac{d}{dx} - \theta x$ will simply collapse down to $-\theta$, since it's a constant that is applied to our differentiating variable. The chain rule states:

$$\frac{d}{dx} f(g(x)) = f'(g(x)) \cdot g'(x)$$
$$\frac{d}{dt} e^{-\theta x} = e^{-\theta x} \cdot (-\theta) = -\theta e^{-\theta x}$$

We can now revisit the prudct rule and apply it, we know that the derivative of our first term is zero, thus we can ignore the first term of the product rule. Only leaving the following: $\frac{d}{dt}x \cdot y = xy'$, we have already calculated $y' = -\theta e^{-\theta x}$ in the previous step. Thus we can Just multiply $-\theta e^{-\theta x}$ with $(X_0 - \mu)$, and we get the following result for our whole derivative:

$$\frac{d}{dt}\left(\mu + (X_0 - \mu) \cdot e^{-\theta t}\right) = (-\theta e^{-\theta x}) \cdot (X_0 - \mu)$$

We can rearrange this to:

$$\frac{d}{dt}\left(\mu + (X_0 - \mu) \cdot e^{-\theta t}\right) = -\theta(X_0 - \mu)e^{-\theta t}$$

This differential desribes the mean-reverting tug applied to our brownian motion model.

2.3 Deriving a full Ornstein Uhlenbeck Process

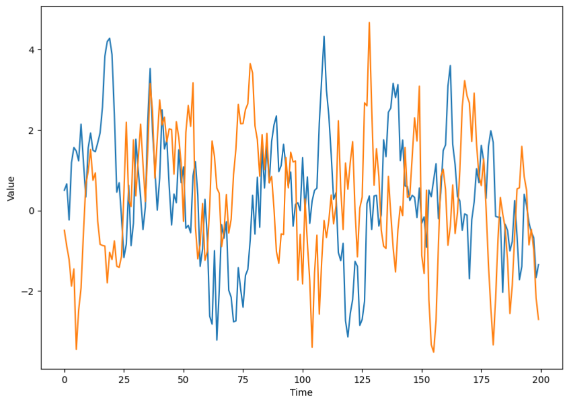
NOT SURE IF THIS IS THE BEST WAY TO DO IT, MIGHT NOT BE IN THE FINAL IA.

We can now try to perform a few experiments on a naively-formulated OU (Ornstein Uhlenbeck) process. We can begin by simply adding a brownian motion to our stochastic process, and assuming (naive step) $X_0 = X_t$, since at each point, the differential will get us tug, this will mean that our t is now zero. We are left with the following stochastic process:

$$X_{t+1} = -\theta(X_t - \mu)e^{-\theta \cdot 0} + W_t$$

$$X_{t+1} = -\theta(X_t - \mu)e + \mathcal{N}(0, \sigma^2)$$

We can not create stochastic paths along the process and they should be mean reverting.



Here we can see that the stochastic process is much more contained within a range, and is thus mean-reverting.

3 Appendix

Code for generating a sample Wiener process:

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np

NUM_PATHS = 10
SAMPLES = 1000

MU = 0
SIGMA = 1

rng = np.random.default_rng()

def compute_wiener() -> list[float]:
    value = 0
    wiener = []

    for _ in range(SAMPLES):
        s = rng.normal(MU, SIGMA)
        value += s

        wiener.append(value)

    return wiener


df_data = {}
for i in range(NUM_PATHS):
    path = compute_wiener()
    df_data[f'WPath {i}'] = path
```

```
plt.figure(figsize=(10, 7))

data = pd.DataFrame(df_data)
sns.lineplot(data=data)

plt.legend([], [], frameon=False)
plt.show()
```

Code for generating a sample naive Ornstein-Uhlenbeck process:

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np

NUM_PATHS = 2
SAMPLES = 200
THETA = 0.2
MEAN = 0

MU = 0
SIGMA = 1

rng = np.random.default_rng()

def compute_wiener() -> list[float]:
    value = 0
    wiener = []

    for _ in range(SAMPLES):
        mean_rev_term = -THETA * (value - MEAN)
        s = mean_rev_term + rng.normal(MU, SIGMA)
        value += s

        wiener.append(value)

    return wiener


df_data = {}
for i in range(NUM_PATHS):
    path = compute_wiener()
    df_data[f'WPath {i}'] = path

data = pd.DataFrame(df_data)

long_form = pd.melt(data.reset_index(), id_vars='index', var_name='Path', value_name=
long_form.rename(columns={'index': 'Time'}, inplace=True)

plt.figure(figsize=(10, 7))
sns.lineplot(data=long_form, x='Time', y='Value', hue='Path')

plt.legend([], [], frameon=False)
plt.show()
```

Note that this is just a draft, code will be refactored and common functions will be declared to shorten the appendix and remove redundancies.