The Derivation: From SDE to Python Code

Objective: Solve the Ornstein-Uhlenbeck Stochastic Differential Equation (SDE) to find the formula for $X_{t+\Delta t}$ given X_t.

1. The Continuous "Physics" (The SDE)

We start with the Ornstein-Uhlenbeck equation, which models a mean-reverting process (like a rubber band pulling the price back to a center).

$$dX_t = \theta(\mu - X_t)dt + \sigma dW_t$$

- dX_t: The tiny change in the spread over a tiny time dt.
- θ: The Mean Reversion Rate. How strong is the rubber band? (High θ = snaps back fast).
- μ: The Long-Term Mean. Where is the rubber band anchored?
- X_t: The current value of the spread.
- σdW_t: The Noise. dW_t is a random shock (Brownian Motion).

The “chemistry intuition”

- μ: the equilibrium concentration. No matter where the concentration starts, the system "wants" to eventually settle at μ.
- θ: the rate constant.
    - High θ: The reaction is fast. If you disturb the system, it snaps back to equilibrium instantly.
    - Low θ: The reaction is sluggish. If the price drifts away, it might take weeks to come back.
- $(\mu - X_t)$: the driving force. This is the concentration gradient or "distance from equilibrium."
- σdW_t: the thermal fluctuations. σ is the volatility (strength of the noise), and dW_t is the increment of a Wiener process (Brownian motion).

2. Rearranging for Solution

We want to solve for X_t. First, let's rearrange the terms to look like a standard linear Ordinary Differential Equation (ODE). Move the X_t term to the left:

$$dX_t + \theta X_t dt = \theta\mu dt + \sigma dW_t$$

3. The "Integrating Factor" Trick

This is a standard ODE technique. We want to collapse the left side ($dX_t + \theta X_t dt$) into a single derivative. We do this by multiplying the entire equation by the Integrating Factor $e^{\theta t}$.

Multiply both sides by $e^{\theta t}$:

$$e^{\theta t}dX_t + e^{\theta t}\theta X_t dt = e^{\theta t}\theta\mu dt + e^{\theta t}\sigma dW_t$$

Key Observation (The Product Rule):

In standard calculus, the derivative of a product $d(X_t e^{\theta t})$ is:

$$d(X_t e^{\theta t}) = e^{\theta t} dX_t + X_t(\theta e^{\theta t} dt)$$

Notice that the right side of this Product Rule matches the left side of our equation exactly! So, we can rewrite the left side as a single derivative:

$$d(X_t e^{\theta t}) = \theta \mu e^{\theta t} dt + \sigma e^{\theta t} dW_t$$

4. Integration

Now we integrate both sides from the current time s to a future time t:

$$\int_s^t d(X_u e^{\theta u}) = \int_s^t \theta \mu e^{\theta u} du + \int_s^t \sigma e^{\theta u} dW_u$$

The left side simplifies instantly (fundamental theorem of calculus):

$$X_t e^{\theta t} - X_s e^{\theta s} = \int_s^t \theta \mu e^{\theta u}\, du + \int_s^t \sigma e^{\theta u} dW_u$$

Solve the first integral on the right (it's just a standard exponential integral):

$$\int_s^t \theta \mu e^{\theta u}\, du = \theta \mu \left[\frac{e^{\theta u}}{\theta}\right]_s^t = \mu(e^{\theta t} - e^{\theta s})$$

Now the equation is:

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

5. Solving for X_t

To isolate X_t, multiply the entire equation by $e^{-\theta t}$:

$$X_t = X_s e^{\theta(s-t)} + \mu\left(1 - e^{\theta(s-t)}\right) + \sigma e^{-\theta t} \int_s^t e^{\theta u} dW_u$$

6. Discretization (The Step Required for Python)

In your code, you are stepping from today (t) to tomorrow $(t + \Delta t)$.

Let's set:

- Current time $s = t$
- Future time $t = t + \Delta t$
- Therefore, $s - t = -\Delta t$

Substitute these into our solution:

$$X_{t+\Delta t} = X_t e^{-\theta\Delta t} + \mu\left(1 - e^{-\theta\Delta t}\right) + \text{Noise Term}$$

This equation says the future price is a weighted average of two things:

- $X_{current} \cdot e^{-\theta\Delta t}$: memory of the past
- $\mu\left(1 - e^{-\theta\Delta t}\right)$: pull of the mean

The Noise Term:

The messy integral term $\sigma e^{-\theta(t+\Delta t)} \int_t^{t+\Delta t} e^{\theta u} dW_u$ looks scary, but for a computer, it is just a random number!

It is a Gaussian Random Variable (Normal Distribution) with mean 0 and variance:

$$\text{Variance} = \frac{\sigma^2}{2\theta}\left(1 - e^{-2\theta\Delta t}\right)$$

7. Connection to Kalman Filter

The Kalman Filter assumes the world works like this linear equation:

$$x_k = F \cdot x_{k-1} + B \cdot u_k + w_k$$

$$z_k = H \cdot x_k + v_k$$

We do not "derive" the Kalman Filter from the Ornstein-Uhlenbeck (OU) equation. Instead, we use the Kalman Filter as a generic **container** (algorithm) and fill it with the specific **physics** (equations) of the OU process.

We mapped the continuous SDE solution ($X_{t+\Delta t}$) to these discrete matrices:

Kalman Parameter	**OU Equivalent (Derived)**	**Physical Meaning**
State (x)	X_t (True Spread)	The invisible, fundamental economic difference between the assets.
Transition (F)	$e^{-\theta\Delta t}$	**The Memory.** How much of yesterday's spread survives to today? (Decay factor).
Control (B)	$\mu\left(1 - e^{-\theta\Delta t}\right)$	**The Gravity.** The force pulling the spread back to the long-term mean (μ).
Process Noise (Q)	$\frac{\sigma^2}{2\theta}\left(1 - e^{-2\theta\Delta t}\right)$	**The Fog.** The accumulation of new uncertainty/randomness over time step Δt.

F (The State Transition Matrix)

What is it? F is the "Physics Engine." It tells the Kalman Filter how the system evolves naturally if there was no random noise.

$$F = e^{-\theta \Delta t}$$

- Physical meaning: F is the decay factor
- If Δt is small (1 second), F is close to 0.999 (The price hardly changes).
- If Δt is huge (1 year), F is close to 0 (The current price doesn't matter; we have fully reverted to the mean).

Q (The Process Noise Covariance)

What is it? Q is the "Uncertainty Bubble." It tells the Kalman Filter: "Even if your Physics Engine is perfect, the world is chaotic. How much error should we expect over this specific time step?"

Q is the variance of the integral term:

$$Q = \text{Variance(Noise)} = \frac{\sigma^2}{2\theta}\left(1 - e^{-2\theta \Delta t}\right)$$

- Physical meaning: Q accumulates time
- If Δt is small, Q is tiny (The price can't jump very far in 1 millisecond).
- If Δt is huge, Q grows (Anything could happen in a year).

8. The Engineering Parameters (The "Sensors")

A. The Measurement (Z)

- **Definition:** The "Observed Spread."
- **Formula:** $Z_t = \text{Price}_{Canada} - (\beta \times \text{Price}_{Australia})$
- **Meaning:** This is a calculated synthetic value derived from noisy market prices. It is our "dirty window" into the true spread.

B. Measurement Noise Covariance (R)

- **Definition:** The variance of the sensor noise (v_k).
- **Physical Meaning:** How much do we trust the market price?
  - **High R:** "The market is chaotic/glitchy. Ignore small jumps." (Filter is lazy).
  - **Low R:** "The market is efficient/perfect. Every tick is real." (Filter is reactive).

C. Estimation Uncertainty (P)

- **Definition:** The variance of the error between our Estimate and the Truth ($P_k = E\left[\left(x - \hat{x}\right)^2\right]$).

- **Physical Meaning:** The filter's internal **"Self-Doubt Score."**
  - **Increases:** When we predict (Time passes $\to$ Fog accumulates).
  - **Decreases:** When we update (Data arrives $\to$ We see reality).

D. The Kalman Gain (K)

- **Definition:** The "Trust Factor" or "Blending Weight" (Calculated automatically).
- **Formula:** $K = P/(P + R)$.
- **Physical Meaning:** Who does the filter trust?
  - **If P is High (Confused Model):** $K \to 1$. Trust the Measurement.
  - **If R is High (Noisy Market):** $K \to 0$. Trust the Physics.

9. The Complete Execution Workflow

Step 1: Input (The Raw Material)

We start with two raw time-series: P_{EWC} (Canada) and P_{EWA} (Australia).

Step 2: Pre-Process (Constructing Z)

We convert raw prices into a single "Observed Spread" series using the static Beta.

- **Operation:** $Z_t = P_{EWC,t} - \beta P_{EWA,t}$.
- **Result:** A jagged, noisy line representing the market's opinion of the spread.

Step 3: Initialization (Turning the Key)

We instantiate the Kalman Filter.

- **User Guess:** We provide guesses for θ, μ, σ (Physics) and R (Noise).
- **Internal Calculation:** The Kalman Filter calculates matrices F, B, Q **once** and locks them in.
- **Initial State:**
  - self.x = 0.0 (Start at equilibrium).
  - self.P = 1.0 (Start with high uncertainty "I don't know").

Step 4: The "Closed-Open Eyes" Loop

We iterate through the data one day at a time.

Phase A: Predict (Closed Eyes)

- **Context:** It is morning. We have not seen today's price yet.
- **Logic:** Use Physics (F, B) to project where the spread *should* be.

- **Parameter Change (State):** self.x decays toward μ.
    - *Example:* If yesterday was 10.0 and $F = 0.9$, self.x becomes 9.0.
- **Parameter Change (Uncertainty):** self.P **Increases**.
    - *Math:* $P_{new} = P_{old} \times F^2 + Q$.
    - *Why:* Time passed. The "Fog" (Q) made our estimate staler.

Phase B: Update (Open Eyes)

- **Context:** Market closes. We receive measurement Z_t (e.g., 12.0).
- **Logic:** Compare Prediction (9.0) vs Reality (12.0). The difference (3.0) is the "Innovation."
- **Parameter Change (Kalman Gain):** Calculate $K = P/(P + R)$.
    - *Decision:* If P is high vs R, K is large (e.g., 0.8).
- **Parameter Change (State):** Correct self.x.
    - *Math:* $x_{new} = x_{pred} + K(Z - x_{pred})$.
    - *Result:* $9.0 + 0.8(3.0) = 11.4$. (We shifted closer to the measurement).
- **Parameter Change (Uncertainty):** self.P **Decreases**.
    - *Math:* $P_{new} = (1 - K)P_{old}$.
    - *Why:* We saw data. The mystery is resolved. Uncertainty drops.

Step 5: Output (The Deliverable)

We extract two lists from the loop:

1. **kf_states:** The "True Spread." This is the denoised signal we will eventually use for trading.
2. **kf_uncertainty:** The history of P. This tells us how "scared" or "confident" the model was on any given day.

10. Why do we do this?

State Estimation (Denoising)

The market price (Z) is a mixture of Signal (X) and Noise (v).

$Z = X + v$

We cannot trade Z directly because the noise triggers false signals. The "Closed-Open Eyes" loop mathematically strips away v based on our noise setting R, leaving us with the pure signal X.

Currently, we are **guessing** R=0.5.

In the next phase, we will use **Maximum Likelihood Estimation (MLE)** to mathematically calculate the *optimal* R and θ that maximize the probability of the data we observed.