STATISTICAL ARBITRAGE · บทที่ 15

Dynamic Hedging ด้วย Kalman Filter: β ที่เปลี่ยนตามเวลา

β_t ไม่คงที่ — Kalman Filter track β แบบ real-time โดยไม่ต้อง refit

ตัวอย่างหลัก: Bybit BTC Perp ↔ Lighter BTC Perp

แก่นของบทนี้ — อ่านก่อน

β ระหว่าง Bybit และ Lighter ไม่คงที่ตลอด — เปลี่ยนตามสภาพตลาด Kalman Filter ให้ estimate β_t แบบ online โดยไม่ต้อง refit ทั้งหมด ทำให้ hedge ratio ตามทัน regime เปลี่ยน

State: β_t = β_{t-1} + w_t | Observation: r_A,t = β_t · r_B,t + v_t

15.1 ปัญหาของ Static β

บทก่อนๆ ใช้ β ที่ estimate จาก OLS บน window คงที่ เช่น 30 วัน แนวทางนี้มีปัญหาหลายประการ

ปัญหาของ Rolling OLS

- **Lag problem:** β วันนี้คือค่าเฉลี่ยของ 30 วันที่ผ่านมา ไม่ใช่ค่า "ตอนนี้"
- **Window selection:** window 7 วัน responsive แต่ noisy; window 60 วัน stable แต่ช้า ไม่มี window ที่ถูกเสมอ
- **Regime change:** เมื่อ volatility หรือ liquidity เปลี่ยนกะทันหัน β OLS ปรับไม่ทัน
- **Computational cost:** Refit OLS ทุก tick = matrix inversion ซ้ำๆ

Kalman Filter แก้ปัญหาอย่างไร?

- **Online update:** ปรับ β_t ทุก timestep ด้วยสูตรเดียว — ไม่ต้อง refit ทั้งหมด
- **ไม่มี window:** ใช้ Q และ R แทน window — ทั้งคู่มีความหมายทางสถิติชัดเจน
- **Uncertainty tracking:** P_t บอก confidence ใน estimate ปัจจุบัน
- **Fast computation:** O(1) per update — เหมาะกับ real-time มาก

15.2 State Space Model — สองสมการหลัก

Kalman Filter ทำงานบน State Space Model ซึ่งแบ่งเป็นสองส่วน

สมการ STATE SPACE

Process equation: β_t = β_{t-1} + w_t, w_t ~ N(0, Q)

Observation equation: r_{A,t} = β_t · r_{B,t} + v_t, v_t ~ N(0, R)

β_t = state ที่ซ่อนอยู่ (true hedge ratio ณ เวลา t) — ไม่สังเกตโดยตรง
r_{A,t}, r_{B,t} = return ของ Bybit และ Lighter — สังเกตได้
w_t = process noise — β เปลี่ยนแค่ไหนต่อ step (Q ใหญ่ = เปลี่ยนเร็ว)
v_t = observation noise — return มี noise เท่าไร (R ใหญ่ = noisy มาก)

พารามิเตอร์	ความหมาย	ค่าที่ใช้ทั่วไป	ตัวอย่างตัวเลข	ผลเมื่อเพิ่ม
Q (process noise)	β เปลี่ยนเร็วแค่ไหนต่อ step	1e-5 ถึง 1e-3	Q=1e-3 → β drift ≈ √(1e-3) ≈ 0.032 ต่อ bar Q=1e-5 → β drift ≈ 0.003 ต่อ bar (10× ช้ากว่า)	β ตอบสนองเร็วขึ้น แต่ noisy
R (observation noise)	Return มี noise เท่าไร	1e-3 ถึง 1e-1	R=0.01 → return std ≈ 1%; R=0.001 → ≈ 0.3% สำหรับ BTC perp hourly: R ≈ 0.0001–0.001	Kalman gain ลด → ปรับช้าลง
P_0 (initial uncertainty)	ความไม่แน่ใจใน β ตอนเริ่ม	1.0	P_0=1.0 → warm-up ~50 bars; P_0=0.01 → เริ่มเชื่อค่า β เดิมมาก	warm-up period สั้นลง

15.3 Kalman Filter Steps — Predict และ Update

ทุก timestep มีสองขั้นตอน: **Predict** (คาดการณ์ล่วงหน้า) และ **Update** (ปรับด้วยข้อมูลใหม่)

ขั้นตอน Predict

```
β̂_{t|t-1} = β̂_{t-1|t-1} (คาด β จาก prior) P_{t|t-1} = P_{t-1|t-1} + Q
(uncertainty เพิ่มขึ้น เพราะเวลาผ่าน)
```

เพราะ process equation บอกว่า β drift ด้วย w_t ~ N(0,Q) → uncertainty เพิ่มทุก step

ขั้นตอน Update (เมื่อเห็น r_A,t และ r_B,t)

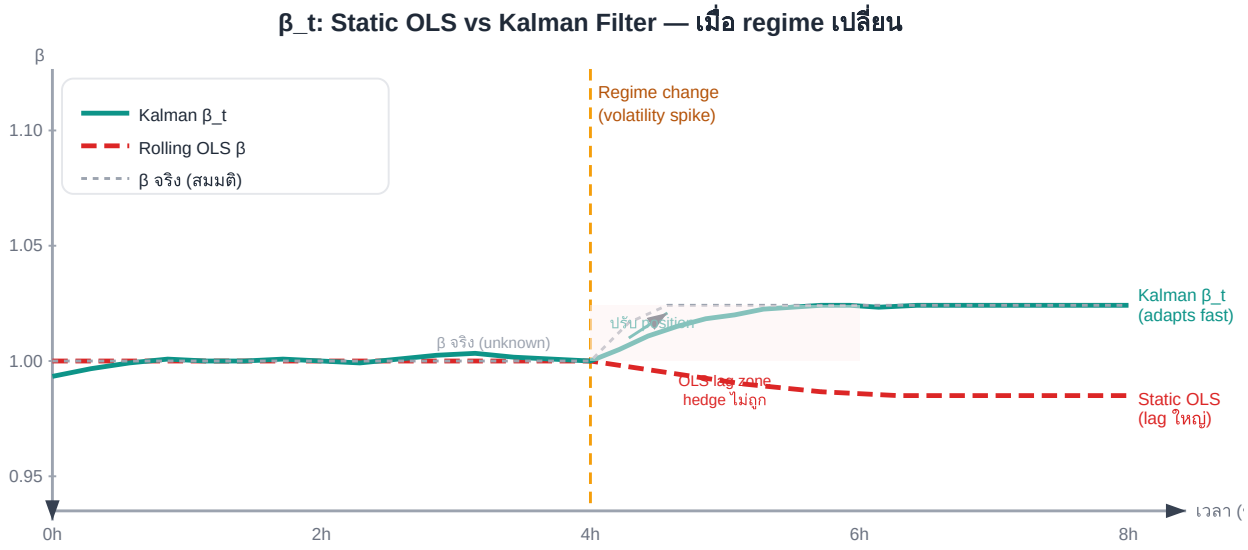
```
innovation = r_{A,t} - β̂_{t|t-1} · r_{B,t} (ความต่างจากที่คาด) S = r_{B,t}² ·
P_{t|t-1} + R (variance ของ innovation) K_t = P_{t|t-1} · r_{B,t} / S
(Kalman Gain) β̂_{t|t} = β̂_{t|t-1} + K_t · innovation (update β) P_{t|t} =
(1 - K_t · r_{B,t}) · P_{t|t-1} (update uncertainty)
```

ความหมายของ Kalman Gain K_t

K_t อยู่ระหว่าง 0 ถึง 1 และบอกว่า "เชื่อข้อมูลใหม่มากแค่ไหน"

- **K_t ใกล้ 1:** เชื่อข้อมูลใหม่มาก (Q ใหญ่หรือ R เล็ก) — β ปรับเร็ว
- **K_t ใกล้ 0:** เชื่อ prior มาก (Q เล็กหรือ R ใหญ่) — β ปรับช้า
- ระยะแรก (P สูง): K สูง — warm up เร็ว; เมื่อ P ลด: K ลด — stable estimate

15.4 SVG Diagram — β_t Kalman vs Static OLS



เมื่อ regime เปลี่ยน Kalman ปรับ β_t เร็วกว่า OLS มาก ลด hedge error ใน zone สีชมพู

15.5 Pseudo-code: kalman_beta_update()

```
def kalman_beta_update(beta, P, r_A, r_B, Q=1e-5, R=1e-3): # --- Predict step --- beta_pred = beta # β ไม่เปลี่ยน (random walk assumption) P_pred = P + Q # uncertainty เพิ่มขึ้น # --- Update step --- innovation = r_A - beta_pred * r_B # ความต่างจากที่คาด S = r_B**2 * P_pred + R # variance ของ innovation K = P_pred * r_B / S # Kalman Gain beta_new = beta_pred + K * innovation # update β P_new = (1 - K * r_B) * P_pred # update uncertainty return beta_new, P_new
```

วิธีใช้ใน loop จริง

```
# Initialization beta = 1.0 # เริ่มจาก 1:1 (prior) P = 1.0 # high uncertainty ช่วงแรก # Real-time update for each new price tick: r_A = log(price_A_now / price_A_prev) r_B = log(price_B_now / price_B_prev) beta, P = kalman_beta_update(beta, P, r_A, r_B, Q=1e-5, R=1e-3) # Recompute spread with new beta epsilon = log(price_A) - beta * log(price_B) z_score = (epsilon - epsilon_mean) / epsilon_std
```

> ⚠️ **Implementation Trap: epsilon_mean และ epsilon_std ต้องอัปเดตตาม β**
>
> **ทำไม θ และ σ_stat เดิมใช้ไม่ได้?** เพราะ ε = log(P_A) − β·log(P_B) — ถ้า β เปลี่ยน ค่า ε **ทุกจุดในอดีตก็เปลี่ยนด้วย** ราวกับว่าเรา recompute ε series ทั้งหมดใหม่ด้วย β ใหม่ ดังนั้น mean และ std ที่ calibrate ไว้บน ε เดิมจะ mismatch กับ ε ชุดใหม่ทันที
>
> ตัวอย่าง: β เปลี่ยนจาก 1.000 → 1.005 เมื่อ log(P_B) ≈ 10 → ε เปลี่ยนทุกจุดไป −0.05 → θ เดิมที่ 0.001 กลายเป็น −0.049 ทันที ถ้า z-score ยังใช้ θ เก่า จะ off ไปถาวร
>
> วิธีแก้ที่แนะนำ: ใช้ **EMA ที่อัปเดต real-time ทุก bar** แทนค่าคงที่จาก calibration:

```
# ✅ Concrete solution: Welford-style EMA mean+variance # Initialization จาก historical calibration alpha_ema = 0.02 # EMA decay (effective window ≈ 1/α = 50 bars) epsilon_mean = epsilon_0 # ค่าเริ่มต้นจาก historical mean ของ ε epsilon_var = epsilon_var0 # variance เริ่มต้นจาก historical var # Real-time update loop for each new price tick: r_A = log(price_A_now / price_A_prev) r_B = log(price_B_now / price_B_prev) # 1. Update β via Kalman beta, P = kalman_beta_update(beta, P, r_A, r_B, Q=1e-5, R=1e-3) # 2. Recompute epsilon ด้วย beta ใหม่ epsilon = log(price_A) - beta * log(price_B) # 3. ✅ FIX: คำนวณ diff ก่อน update mean (Welford-style) # ป้องกัน variance bias จากการใช้ updated mean ใน variance update diff = epsilon - epsilon_mean # ใช้ OLD mean # 4. Update mean epsilon_mean = epsilon_mean + alpha_ema * diff # 5. Update variance ด้วย
```

```
OLD diff (EMA variance: w_new*diff² + w_old*var_old) epsilon_var = (1 -
alpha_ema) * epsilon_var + alpha_ema * diff**2 epsilon_std = sqrt(epsilon_var +
1e-10) # guard zero division # 6. z-score ที่ unbiased z_score = (epsilon -
epsilon_mean) / epsilon_std
```

15.6 Choosing Q and R — Tuning พารามิเตอร์

การเลือก Q และ R เป็น trade-off ระหว่าง responsiveness กับ stability

สถานการณ์	Q ที่แนะนำ	R ที่แนะนำ	ผลที่ได้
β เปลี่ยนเร็ว (volatile market)	ใหญ่ (1e-4)	ปกติ (1e-3)	β ตอบสนองเร็ว แต่ noisy
β เปลี่ยนช้า (stable pair)	เล็ก (1e-6)	ปกติ (1e-3)	β smooth แต่ lag เมื่อ regime เปลี่ยน
Data noisy มาก (thin market)	ปกติ (1e-5)	ใหญ่ (1e-2)	ไม่ over-react ต่อ noise
Data สะอาด (deep market)	ปกติ (1e-5)	เล็ก (1e-4)	update เร็วขึ้นจาก each observation

วิธีหา Q และ R จากข้อมูล

Empirical Calibration

- **R:** ประมาณจาก variance ของ residual ε บน historical data
  `R ≈ Var(r_A − β_static · r_B)`
- **Q:** ประมาณจาก variance ของการเปลี่ยนแปลง β ระหว่าง rolling windows
  `Q ≈ Var(β_{t} − β_{t-1})` จาก rolling OLS ที่ window เล็ก
- **Cross-validate:** test บน out-of-sample เพื่อหา (Q,R) ที่ให้ spread mean-revert ที่สุด

15.7 Position Adjustment เมื่อ β เปลี่ยน

เมื่อ Kalman ให้ β_new ≠ β_old คุณต้องปรับ position เพื่อรักษา hedge ratio ที่ถูกต้อง

สูตรปรับ POSITION

$$\text{delta_position_B} = (\beta_new - \beta_old) \times size_A$$

delta_position_B > 0: β เพิ่ม → ต้อง short B เพิ่มขึ้น (หรือ long B น้อยลง)
delta_position_B < 0: β ลด → ต้อง short B น้อยลง (หรือ long B เพิ่ม)
ทำ adjustment เฉพาะถ้า |delta| > threshold เพื่อลด transaction cost

Threshold สำหรับ Rebalancing

อย่าปรับทุก tick เพราะ transaction cost กินกำไร ตั้ง threshold เช่น:

```
if abs(beta_new - beta_old) > 0.01: # β เปลี่ยน > 1% delta = (beta_new -
beta_old) * size_A adjust_position(venue="Lighter", delta=delta) beta_old
= beta_new
```

0.01 เป็น rule of thumb — calibrate จาก fee structure จริง

15.8 Running Example — Bybit↔Lighter β Drift

Running Example: β เปลี่ยนจาก 0.98 → 1.05 ใน 2 ชั่วโมง

สถานการณ์: BTC มี funding rate surge บน Lighter ทำให้ราคา Lighter ขึ้นเร็วกว่า Bybit ชั่วคราว

```
เริ่มต้น: beta = 0.98, P = 0.001 t=0h: r_A=+0.12%, r_B=+0.10% → innovation ≠ 0 → beta เริ่มปรับขึ้น t=0.5h: r_A=+0.15%, r_B=+0.14% → beta ≈ 0.995 t=1.0h: r_A=+0.18%, r_B=+0.17% → beta ≈ 1.015 t=1.5h: r_A=+0.20%, r_B=+0.19% → beta ≈ 1.035 t=2.0h: r_A=+0.22%, r_B=+0.21% → beta ≈ 1.05 ✓
```

การปรับ position:

- size_A = 1.0 BTC (Long Bybit)
- เริ่มต้น short Lighter = 0.98 BTC (β=0.98)
- หลัง 2 ชั่วโมง β=1.05 → delta = (1.05−0.98)×1.0 = **0.07 BTC**
- ต้อง short Lighter เพิ่มอีก 0.07 BTC เพื่อ hedge ถูกต้อง

ค่า ε ถ้าไม่ปรับ: ε = log(P_A) − 0.98×log(P_B) แต่ β จริงเป็น 1.05 → ε drift ขึ้นจาก basis error ไม่ใช่ alpha จริง → อาจ trigger entry ปลอม

15.9 แบบฝึกหัด

โจทย์ 15.1 — คำนวณ Kalman Update ด้วยมือ

ให้: beta = 1.00, P = 0.001, Q = 1e-5, R = 1e-3
Observation ใหม่: r_A = +0.15%, r_B = +0.12%

ถาม: คำนวณ beta_new และ P_new ทีละขั้น

▶ ดูคำตอบ

โจทย์ 15.2 — ผลของ Q ต่าง

ทดสอบ Kalman กับ Q = 1e-6 และ Q = 1e-3 บน series เดียวกันที่ β จริงเปลี่ยนจาก 1.0 → 1.1 ใน 100 steps

ถาม: describe ว่า Q แต่ละค่าจะ track β_true ได้อย่างไร มีผลดีผลเสียอะไร?

▶ ดูคำตอบ

โจทย์ 15.3 — Rebalancing Threshold

β เปลี่ยนจาก 1.00 → 1.03 (delta = 0.03), size_A = 5 BTC, Lighter taker fee = 0.07%
ราคา BTC = $65,000

ถาม: ค่า transaction cost ของการ rebalance คือเท่าไร? ควร rebalance หรือไม่ ถ้า expected P&L จากการแก้ hedge error = $35?

▶ ดูคำตอบ

อ้างอิงสำคัญ (References)

- **Kalman, R. E. (1960).** A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1), 35–45. — บทความต้นฉบับ Kalman filter; state-space recursion สำหรับ dynamic β_t ใน Ch.15

- **Welch, G., & Bishop, G. (2006).** *An Introduction to the Kalman Filter.* UNC Chapel Hill TR 95-041. — tutorial ที่อ่านง่ายที่สุดสำหรับ practical Kalman implementation; ครอบคลุม process noise Q และ measurement noise R
- **Elliott, R. J., van der Hoek, J., & Malcolm, W. P. (2005).** Pairs trading. *Quantitative Finance*, 5(3), 271–276. — ประยุกต์ Kalman filter กับ pairs trading spread ที่คล้าย Ch.15 โดยตรง
- **Vidyamurthy, G. (2004).** *Pairs Trading: Quantitative Methods and Analysis.* Wiley. — Ch.8 ของหนังสือนี้กล่าวถึง dynamic hedge ratio ด้วย Kalman