Statistical Arbitrage · บทที่ 0

เริ่มต้นที่นี่ — วิธีอ่านหนังสือเล่มนี้

หนังสือเล่มนี้สำหรับใคร · จะได้อะไรเมื่ออ่านจบ · แผนที่ 23 บท · สัญลักษณ์ที่ใช้ตลอดเล่ม

แก่นของหนังสือเล่มนี้

ราคาสินทรัพย์เดี่ยว ๆ เดินแบบ **random walk** — ไม่มีแรงดึงกลับ ทำนายไม่ได้ แต่ถ้านำสองสินทรัพย์ที่สัมพันธ์กันมา "หักล้าง" กันด้วยอัตราส่วนที่ถูกต้อง จะได้ **process ใหม่ (ε)** ที่วนเวียนรอบค่ากลาง — และสิ่งที่กลับหาค่ากลางได้ คือสิ่งที่เรา trade ได้อย่างมีระบบ หนังสือเล่มนี้สอนตั้งแต่ทฤษฎีรากฐานจนถึงการลงมือเทรดและบริหารความเสี่ยงจริง

$\varepsilon = \log(P_A) - \beta \cdot \log(P_B)$

ก่อนเข้าบทที่ 1 บทนี้จะตอบ 4 คำถาม: หนังสือนี้เหมาะกับคุณไหม · อ่านจบแล้วทำอะไรได้ · 23 บทเรียงกันอย่างไร · และสัญลักษณ์ที่จะเจอตลอดเล่มมีอะไรบ้าง ใช้เวลา 10 นาทีอ่านบทนี้ จะช่วยให้ 23 บทที่ตามมาไหลลื่นขึ้นมาก

0.1 หนังสือเล่มนี้สำหรับใคร

หนังสือเล่มนี้ออกแบบมาสำหรับ **ผู้ที่ไม่เคยเรียน statistical arbitrage หรือ quantitative trading มาก่อน** แต่ละแนวคิดถูกสร้างขึ้นจากศูนย์ — มีแรงจูงใจ (ทำไมต้องรู้) ก่อนสมการเสมอ

ควรมีพื้นฐานนี้มาก่อน	ไม่จำเป็นต้องรู้มาก่อน — หนังสือสอนให้
สถิติพื้นฐาน: ค่าเฉลี่ย, ส่วนเบี่ยงเบนมาตรฐาน, การแจกแจงปกติ	ประสบการณ์เทรดจริง — ไม่ต้องเคยเทรดมาก่อน
พีชคณิตและสมการระดับมัธยมปลาย · เข้าใจ log เบื้องต้น	Stochastic calculus / Itô's lemma — สอนในบทที่ 2
อ่าน pseudo-code / โค้ดอย่างง่ายได้ (เห็นตรรกะ if/loop เข้าใจ)	Cointegration, OU process, GARCH, Kalman — สอนทีละบท
ความอดทนที่จะทำแบบฝึกหัดท้ายบท	กลไกตลาด crypto perpetual / CFD — อธิบายเมื่อถึงบทที่ใช้

ถ้าคณิตศาสตร์ไม่ใช่จุดแข็ง

บทที่ 2 (Stochastic Calculus) คือบทที่ "ชัน" ที่สุด — ถ้าอ่านแล้วติด ให้จับใจความ *ผลลัพธ์* ของแต่ละสมการ (กล่อง "แก่นของบทนี้") แล้วเดินหน้าต่อ ไม่ต้องพิสูจน์เองทุกบรรทัด คุณกลับมาเก็บรายละเอียดทีหลังได้ เพราะบทถัด ๆ ไปจะใช้ *ผลลัพธ์* ไม่ใช่กระบวนการพิสูจน์

0.2 เมื่ออ่านจบ คุณจะทำอะไรได้

เป้าหมายปลายทางคือ — สร้างและบริหารกลยุทธ์ statistical arbitrage **หนึ่งกลยุทธ์ได้ครบวงจรจริง** โดยเฉพาะ:

- **เลือกคู่สินทรัพย์** ที่มีความสัมพันธ์เชิงเศรษฐกิจจริง และผ่านการทดสอบทางสถิติ (correlation → cointegration → คุณภาพ residual)
- **หา hedge ratio β** ที่ทำให้ ε เป็น residual จริง ไม่ใช่ directional bet ปลอมตัว
- **วัดและสร้างแบบจำลอง ε** — ความเร็วการกลับค่ากลาง (half-life), ความผันผวน (GARCH), การกระโดด (jump), การเปลี่ยน regime
- **แปลง ε เป็นสัญญาณเทรด** ด้วย z-score และ state machine เข้า/ออกที่ชัดเจน
- **กำหนดขนาด position** ด้วย Kelly + ข้อจำกัด drawdown
- **คำนวณ edge หลังหักต้นทุนจริง** (fee, spread, slippage, funding) — และรู้ว่าเมื่อไรไม่ควรเทรด
- **บริหารความเสี่ยงระดับพอร์ต** และนำกรอบเดียวกันไปใช้กับตลาดอื่น (commodity, options, FX/CFD บน MT5)

สิ่งที่หนังสือเล่มนี้ไม่ใช่

นี่ไม่ใช่ "สูตรลับรวยเร็ว" — statistical arbitrage คือ **ระบบ**ที่ edge บางและต้องอาศัยวินัย ต้นทุนการเทรดจริงกินกำไรได้หมดถ้าประเมินผิด หนังสือจึงเน้นความซื่อสัตย์ต่อตัวเลข (ดูตัวอย่าง edge ติดลบใน บทที่ 19) มากกว่าการขายความฝัน

0.3 แผนที่หนังสือ — 23 บท แบ่งเป็น 4 ภาค

23 บทเรียงกันเป็นลำดับการเรียนรู้เดียว แต่ละบทต่อยอดจากบทก่อนหน้า แนะนำให้อ่านตามลำดับในรอบแรก

ภาค 1 — รากฐาน: process ที่กลับหาค่ากลางคืออะไร
บทที่ 1–3

บทที่ 1 Synthetic Process · บทที่ 2 Stochastic Calculus พื้นฐาน · บทที่ 3 Ornstein-Uhlenbeck Process
ได้อะไร: เข้าใจว่าทำไม spread กลับหาค่ากลางได้ทั้งที่ราคาเดี่ยวทำไม่ได้ และวัด "ความเร็ว" การกลับค่ากลางด้วย OU process

ภาค 2 — สร้างและตรวจสอบ Spread
บทที่ 4–10

บทที่ 4 Hedge Ratio β · บทที่ 5 Cointegration · บทที่ 6 Hurst & Stationarity Tests · บทที่ 7 GARCH · บทที่ 8 Jump-Diffusion OU · บทที่ 9 Regime-Switching OU · บทที่ 10 Z-score & Robust Statistics
ได้อะไร: หา β ที่ถูกต้อง พิสูจน์ว่าคู่สินทรัพย์ trade ได้จริง และสร้างแบบจำลอง ε ที่สมจริง (ผันผวน, กระโดด, เปลี่ยน regime)

ภาค 3 — เปลี่ยน ε ให้เป็นกลยุทธ์ที่เทรดได้

บทที่ 11–16

บทที่ 11 Entry/Exit State Machine · บทที่ 12 Position Sizing · บทที่ 13 Perpetual Basis · บทที่ 14 Cross-venue · บทที่ 15 Kalman Filter · บทที่ 16 Multi-leg Portfolios
ได้อะไร: ระบบเข้า/ออกที่ชัดเจน · ขนาด position ที่เหมาะสม · กลยุทธ์จริงบน crypto perpetual และ hedge ratio แบบ dynamic
ภาค 4 — ขยายตลาด · ต้นทุน · ความเสี่ยง
บทที่ 17–23

บทที่ 17 Commodity Calendar Spread · บทที่ 18 Options Put-Call Parity · บทที่ 19 Cost Analysis & Edge · บทที่ 20 Risk Management & Portfolio · บทที่ 21 Spot↔CFD Swap Arbitrage บน MT5 · บทที่ 22 Options-Based Stat Arb (IV Surface · Box Spread · Risk Reversal · VRP) · บทที่ 23 บทเสริม Advanced: Kalman Filter & t-Distribution
ได้อะไร: นำกรอบเดิมไปใช้กับ futures, options และ FX/CFD · ประเมิน edge หลังต้นทุนจริง · บริหารความเสี่ยงทั้งพอร์ต

0.4 วิธีอ่าน — เครื่องมือช่วยเรียนในแต่ละบท

ทุกบทใช้เครื่องมือช่วยเรียนชุดเดียวกัน รู้จักไว้ก่อนจะอ่านได้เร็วขึ้น:

กล่อง "แก่นของบทนี้ — อ่านก่อน"
กล่องสีเขียว-ฟ้าแบบนี้อยู่ต้นทุกบท — สรุปแนวคิดหลักใน 2–3 บรรทัด ถ้ามีเวลาน้อย อ่านเฉพาะกล่องนี้ของทุกบทก็เห็นภาพรวมได้
ตัวอย่างที่ใช้ตลอดเล่ม (Running Example)

กล่องเส้นประสีส้มคือ **ตัวอย่างหลัก** — คู่ **Bybit BTC Perpetual ↔ Lighter BTC Perpetual** ถูกใช้ต่อเนื่องตั้งแต่บทที่ 1 ถึงบทท้าย ๆ เพื่อให้เห็นกลยุทธ์เดียวถูกสร้างขึ้นทีละชั้น ส่วนบทที่ 17–23 เพิ่มตัวอย่าง commodity, options, FX/CFD และ IV Surface Stat Arb

สีของกล่องมีความหมาย — จำไว้จะอ่านได้เร็วขึ้น:

กล่องเขียว
สิ่งที่ถูกต้อง · แนวปฏิบัติที่ดี · เกณฑ์ที่ "ผ่าน"
กล่องแดง
กับดัก · ข้อผิดพลาดที่พบบ่อย · คำเตือนสำคัญ
กล่องเหลือง
ข้อควรระวัง · กรณีก้ำกึ่งที่ต้องตัดสินใจ
กล่องฟ้า
ตัวอย่าง · ข้อมูลเสริม · การคำนวณประกอบ
กล่องม่วง
ประโยคจำ · ข้อสรุปเชิงแนวคิดที่ควรจดจำ

นอกจากนี้ทุกบทมี **แบบฝึกหัดท้ายบท** (กรอบสีเขียวมิ้นต์) พร้อมเฉลยแบบกดเปิด — แนะนำให้ลองทำเองก่อนเปิดเฉลย เพราะ stat arb เข้าใจได้จริงจากการลงมือคำนวณ ไม่ใช่แค่การอ่าน

0.5 สัญลักษณ์หลักที่ใช้ตลอดเล่ม

หนังสือใช้สัญลักษณ์ชุดเดียวกันทั้ง 23 บท ตารางนี้คือคำแปลอ้างอิง — ยังไม่ต้องเข้าใจทั้งหมดตอนนี้ แต่ละตัวจะถูกอธิบายเต็มเมื่อถึงบทที่ใช้

สัญลักษณ์	ความหมาย	แนะนำในบท
ε (epsilon)	spread / residual — process ใหม่ที่เราสร้างขึ้นและนำไป trade	บทที่ 1
β (beta)	hedge ratio — อัตราส่วนผสมสองขาเพื่อหักล้าง exposure ร่วม	บทที่ 4
θ (theta)	ค่ากลางที่ ε วนเวียนกลับเข้าหา (long-run mean)	บทที่ 3
κ (kappa)	ความเร็วการกลับค่ากลางของ OU process	บทที่ 3
σ (sigma)	ความผันผวน (volatility) ของ process	บทที่ 3
σ_stat	ส่วนเบี่ยงเบนมาตรฐานของ ε ที่สภาวะคงตัว = σ/√(2κ)	บทที่ 3
z (z-score)	ε ห่างจาก θ กี่ส่วนเบี่ยงเบนมาตรฐาน = (ε−θ)/σ_stat	บทที่ 10
$\tau_{1/2}$ (half-life)	เวลาที่ ε ใช้กลับเข้าหาค่ากลางครึ่งทาง = ln(2)/κ	บทที่ 3
φ (phi)	สัมประสิทธิ์ AR(1) ของ ε — ใช้วัดความถี่การตัดผ่านค่ากลาง	บทที่ 5
H (Hurst)	Hurst exponent — H<0.5 mean-revert, =0.5 random walk, >0.5 trending	บทที่ 6
ADF / KPSS	การทดสอบทางสถิติว่า ε เป็น stationary (กลับค่ากลาง) หรือไม่	บทที่ 5, บทที่ 6
f*	Kelly fraction — สัดส่วนของทุนที่ควรลงในแต่ละ trade	บทที่ 12

พร้อมเริ่มแล้ว

ถ้าจับใจความได้ว่า — ราคาเดี่ยวทำนายไม่ได้ แต่ "ส่วนต่างที่ผสมถูกสัดส่วน" กลับหาค่ากลางได้ — คุณพร้อมสำหรับบทที่ 1 แล้ว ส่วนรายละเอียดที่เหลือจะค่อย ๆ ประกอบกันขึ้นทีละบท

Ch.0 เริ่มต้นที่นี่ | ต่อไป → **Ch.1: Synthetic Process คืออะไร?**

Statistical Arbitrage · บทที่ 1

Synthetic Process คืออะไร?

สร้าง process ใหม่ที่ "กลับหาค่ากลาง" จาก 2 assets ที่เดิน random walk
ตัวอย่างหลัก: Bybit BTC Perp ↔ Lighter BTC Perp
แก่นของบทนี้ — อ่านก่อน
ราคา BTC ขึ้นลงไม่มีทิศทาง (random walk) แต่ถ้าเอาราคา BTC จากสองตลาดมาหักล้างกันด้วยอัตราส่วนที่เหมาะสม จะได้ process ใหม่ที่ **วนเวียนอยู่รอบค่ากลาง** — นั่นคือสิ่งที่เราจะ trade
$\varepsilon = \log(P_A) - \beta \cdot \log(P_B)$

1.1 ปัญหาของการ Trade Single Asset

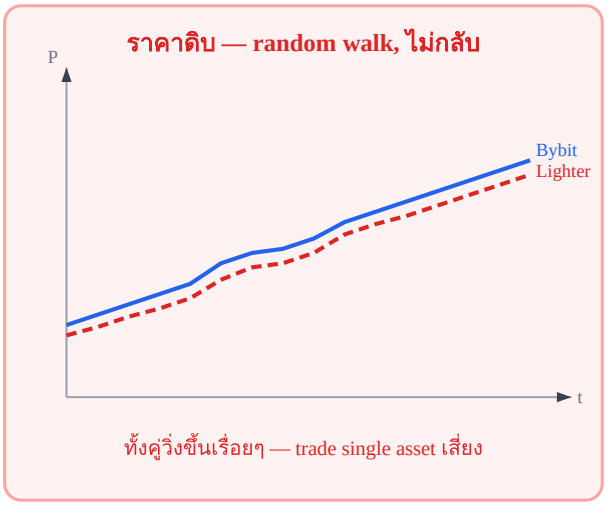
ลองเทียบสองกลยุทธ์:

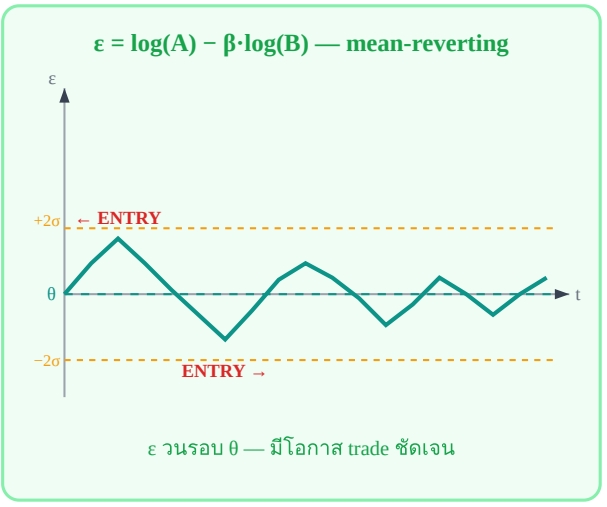
กลยุทธ์ A — Short BTC เมื่อราคาสูง (อันตราย)

Short BTC ที่ $65,000 เพราะ "แพงเกินไป" — แต่ราคาวิ่งไป $90,000
ขาดทุน $25,000 ต่อ BTC และ **ไม่มีเหตุผลว่าราคาจะ "กลับมา"** ที่ $65,000
Random walk ไม่มีแรงดึงกลับ — ราคาอาจไปเรื่อยๆ

กลยุทธ์ B — Trade Spread ระหว่างสองตลาด (ปลอดภัยกว่า)

Bybit BTC perp ราคา $65,050, Lighter BTC perp ราคา $64,980
Spread = $70 (สูงกว่าปกติที่ ~$20)
Trade: Short Bybit, Long Lighter — รอให้ spread กลับมา $20
แรงดึงกลับมีอยู่จริง: ทั้งสองตลาดต้อง track ราคา BTC เดียวกัน





ซ้าย: ราคาดิบทั้งสองตลาดวิ่งขึ้นพร้อมกัน ยากจะ time | ขวา: ε = ความต่าง กลับหาค่ากลางซ้ำๆ

1.2 สร้าง Synthetic Process: สูตรและส่วนประกอบ

สูตรหลัก
$\varepsilon = F(A) - \beta \cdot F(B)$

F(·) = transformation (log price, basis, IV)
β = อัตราส่วนที่ทำให้ ε กลับหาค่ากลาง
ε = process ใหม่ที่เราจะ trade ("spread" หรือ "epsilon")

1.3 β คือ Entry Ratio — ไม่ใช่แค่ตัวเลขในสูตร

นี่คือจุดที่หลายคนเข้าใจผิดที่สุด: **β บอกว่าต้อง trade ในอัตราส่วนเท่าไร** เพื่อให้ ε ไม่ drift

ตัวอย่างเพื่อเห็นภาพ

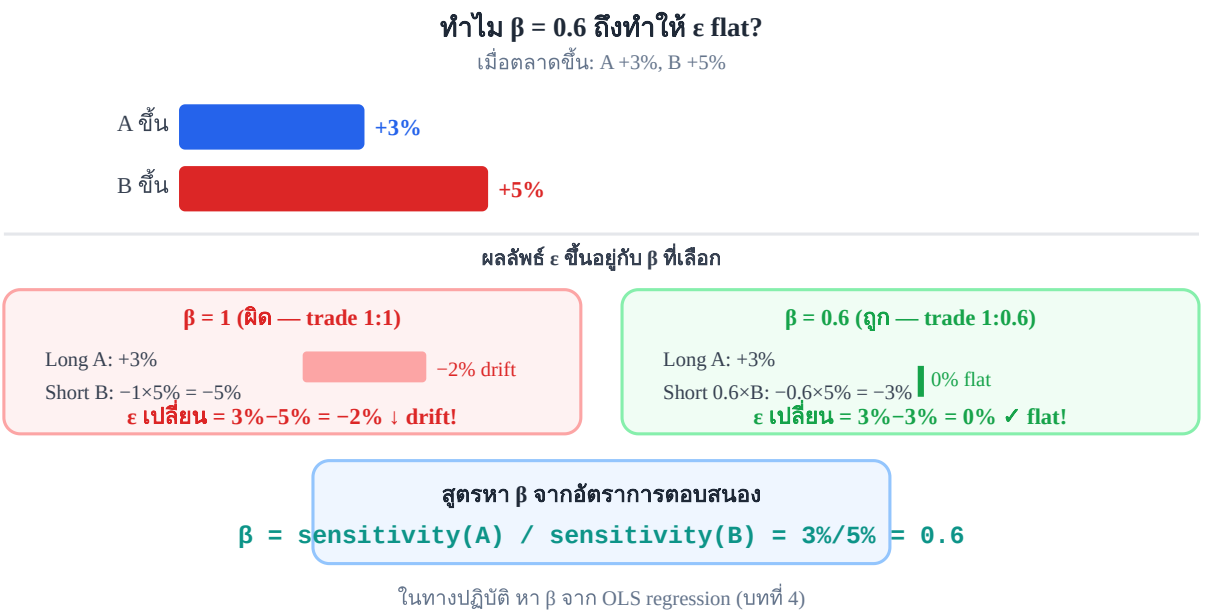
สมมติ: A ขึ้น **+3%** ทุกครั้งที่ B ขึ้น **+5%** (A sensitive น้อยกว่า B)

ถ้า trade 1:1 → ε เปลี่ยน = +3% − +5% = **−2% ทุกครั้ง** → ε drift ลง (ไม่ดี)

ถ้า trade 1:0.6 → ε เปลี่ยน = +3% − 0.6×5% = 3% − 3% = **0% ทุกครั้ง** → ε flat ✓

ดังนั้น β = 3%/5% = **0.6** คือ "อัตราส่วนที่ถูกต้อง"

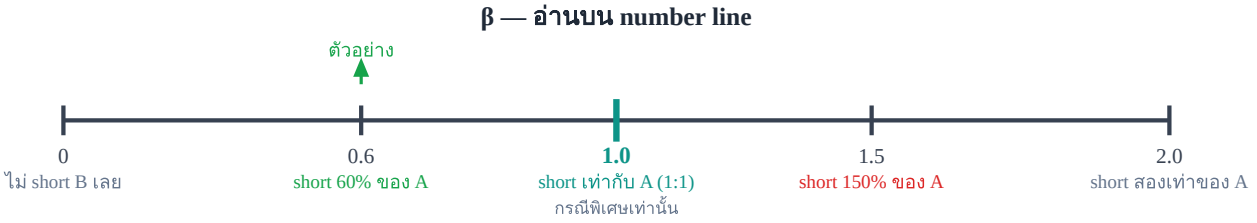
หมายเหตุ: β = sensitivity ratio นี้เป็นเพียงสัญชาตญาณในกรณีอุดมคติที่ไม่มี noise — ในทางปฏิบัติ β ที่เชื่อถือได้ประมาณจาก OLS regression ใน[บทที่ 4]



β = 0.6 หมายถึง: Long A 100,000 บาท → Short B เพียง 60,000 บาท (ไม่ใช่ 1:1)

ตาราง: β ต่างกัน → Position Size ต่างกัน

β	ถ้า Long A = $100,000	ต้อง Short B เท่าไร	ความหมาย
β = 1.00	$100,000	$100,000	A และ B sensitive เท่ากันพอดี
β = 0.60	$100,000	$60,000	B volatile กว่า → short น้อยกว่า
β = 1.50	$100,000	$150,000	B stable กว่า → short มากกว่า
β = 1.003	$100,000	$100,300	Bybit↔Lighter: ต่างนิดหน่อย แต่ไม่ใช่ 1:1



β = 1 เป็นแค่กรณีพิเศษ ในทางปฏิบัติ β แทบไม่เคยเป็น 1 พอดี

1.4 ทำไม Bybit↔Lighter ถึงมี β ≠ 1?

Running Example: Bybit BTCUSDT Perp ↔ Lighter BTCUSDT Perp

ราคาทั้งสองเกือบเท่ากัน (~$65,000) แต่ β ≠ 1 เพราะ:

- **Funding rate mechanism ต่างกัน:** Bybit ใช้ 8h, Lighter ใช้ 1h — ทำให้ราคา drift ต่างกันในช่วงสั้น
- **Liquidity depth ต่างกัน:** Bybit มี volume มากกว่า → ราคาตอบสนองต่อ order flow ต่างกัน

- **Execution latency ต่างกัน:** ราคาที่ fetch มาไม่ synchronous พอดี → ทำให้ sensitivity วัดได้ต่างกัน

สมมติ β = 1.003 จากการ calibrate → Long Bybit $100,000 ต้อง Short Lighter $100,300 → ต่างจาก 1:1 เพียง $300 แต่สำคัญมากเมื่อ trade บ่อย

1.5 ε ที่ดี ต้องมีคุณสมบัติอะไร?

ไม่ใช่ทุก "spread" ที่จะ trade ได้ ε ที่ดีต้องผ่านเกณฑ์:

คุณสมบัติ	ความหมาย	วัดอย่างไร
Stationary	ε วนอยู่รอบค่ากลาง ไม่ drift ออกไปเรื่อยๆ	ADF test (บทที่ 5)
Mean-reverting	ยิ่งห่างจาก θ มาก แรงดึงกลับยิ่งแรง	Half-life (บทที่ 3)
Half-life trade-able	กลับเร็วพอสำหรับ strategy ที่วางแผน	2h–2wk เหมาะสม
Spread profit > ค่าธรรมเนียม	σ_ε ใหญ่กว่า fee+slippage	Edge analysis (บทที่ 19)

1.6 รูปแบบ F(·) ที่ใช้บ่อย

F(·)	สูตร ε	ใช้กับ
Log price	log(P_A) − β·log(P_B)	Bybit↔Lighter perp (เล่มนี้)
Absolute price	P_A − β·P_B	ราคาระดับเดียวกัน, unit เดียว
Basis	P_perp − P_spot	Perpetual vs spot arbitrage
Implied Vol	IV_A − β·IV_B	Options volatility surface arb
Calendar spread	P_front − β·P_back	Futures term structure

ทำไมต้อง log price สำหรับ Bybit↔Lighter?

ราคา BTC ระดับ $65,000 → ถ้าใช้ P_A − β·P_B ตรงๆ spread จะเปลี่ยนตามระดับราคา (high price = large absolute spread)
log(P_A) − β·log(P_B) คือ log ratio → เป็น % deviation → ไม่ขึ้นกับระดับราคา → **stationary กว่า**

โจทย์ 1.1 — คำนวณ position size จาก β

Bybit BTC perp ขึ้น 2.1% ในชั่วโมงที่แล้ว ขณะที่ Lighter BTC perp ขึ้น 2.3%
จาก calibration: β = 0.913

ถาม: ถ้า Long Bybit $500,000 ต้อง Short Lighter เท่าไร?

▶ ดูคำตอบ
โจทย์ 1.2 — ε drift บอกอะไร?

trader ใช้ β=1.0 (แทนที่จะเป็น β=0.913) trade Bybit↔Lighter มา 30 วัน
สังเกตว่า ε ค่อยๆ ลดลงทุกวัน แม้ไม่ได้เปิด position

ถาม: อธิบายว่าเกิดอะไรขึ้น และ drift นี้ส่งผลอย่างไรต่อ P&L ถ้า trader มักจะ short spread?

▶ ดูคำตอบ

โจทย์ 1.3 — เลือก F(·) ที่เหมาะสม

คุณต้องการ trade spread ระหว่าง BTC June futures ($65,200) กับ BTC September futures ($66,100)

ถาม: ควรใช้ F(·) แบบไหน และ spread นี้คือ "basis" ประเภทไหน?

▶ ดูคำตอบ

Ch.1 Synthetic Process | ต่อไป → **Ch.2: Stochastic Calculus พื้นฐาน**

Statistical Arbitrage · บทที่ 2

Stochastic Calculus พื้นฐาน

Brownian Motion · $(dW)^2 = dt$ · Itô's Lemma
เครื่องมือที่จำเป็นสำหรับอ่าน SDE ของ spread
ทำไมต้องเรียนบทนี้

Spread ε เดิน SDE แบบ OU Process: `dε = κ(θ−ε)dt + σdW`
เพื่ออ่านสมการนี้ได้ต้องรู้ว่า dW คืออะไร และทำไม $(dW)^2$ ถึงไม่ตัดทิ้ง

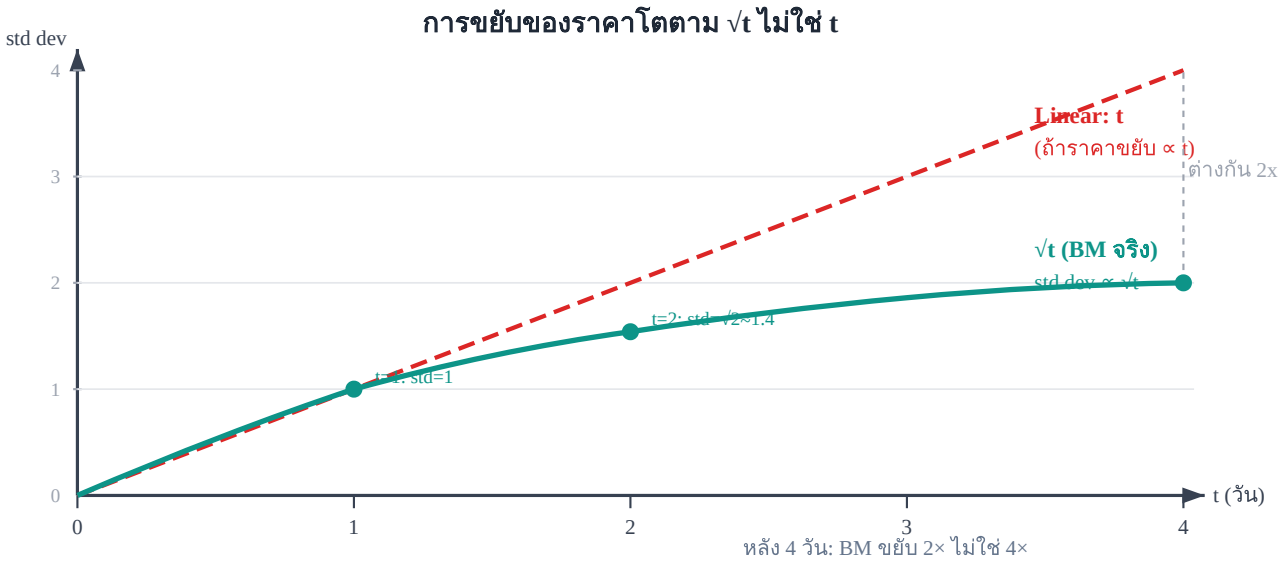
2.1 Brownian Motion — ราคาเดิน Random Walk อย่างไร

Brownian Motion (หรือ Wiener Process) W(t) คือโมเดลที่ใช้อธิบายการเคลื่อนที่แบบสุ่มของราคา มีคุณสมบัติหลัก 3 ข้อ:

3 คุณสมบัติสำคัญของ W(t)

1. **W(0) = 0:** เริ่มต้นที่จุดเดิม
2. **Increments อิสระ:** การขยับในช่วง [s,t] ไม่ขึ้นกับการขยับในช่วง [r,s] — "อดีตไม่ส่งผลต่ออนาคต"
3. **W(t)−W(s) ~ N(0, t−s):** ในช่วงเวลา Δt = t−s ราคาขยับประมาณ ±√Δt (ไม่ใช่ ±Δt)

ข้อ 3 สำคัญมาก: ราคาขยับตาม √t ไม่ใช่ t



Brownian Motion โต "ช้ากว่าที่คิด" — ต้องรอ 4 วันจึงจะขยับ 2 เท่าของวันเดียว

2.2 dW คืออะไร และทำไม $(dW)^2 = dt$

เราใช้ dW_t แทน W(t+dt) − W(t) — การขยับเล็กๆ ในช่วงเวลา dt

dW_t ~ N(0, dt) E[dW_t] = 0 ← ค่าเฉลี่ยเป็นศูนย์ (ไม่รู้จะขึ้นหรือลง) Var[dW_t] = dt ← variance = dt

ทำไม $(dW)^2 = dt$ — ไม่ตัดทิ้งแบบ Calculus ธรรมดา?

Calculus ธรรมดา: $(dx)^2 \to 0$

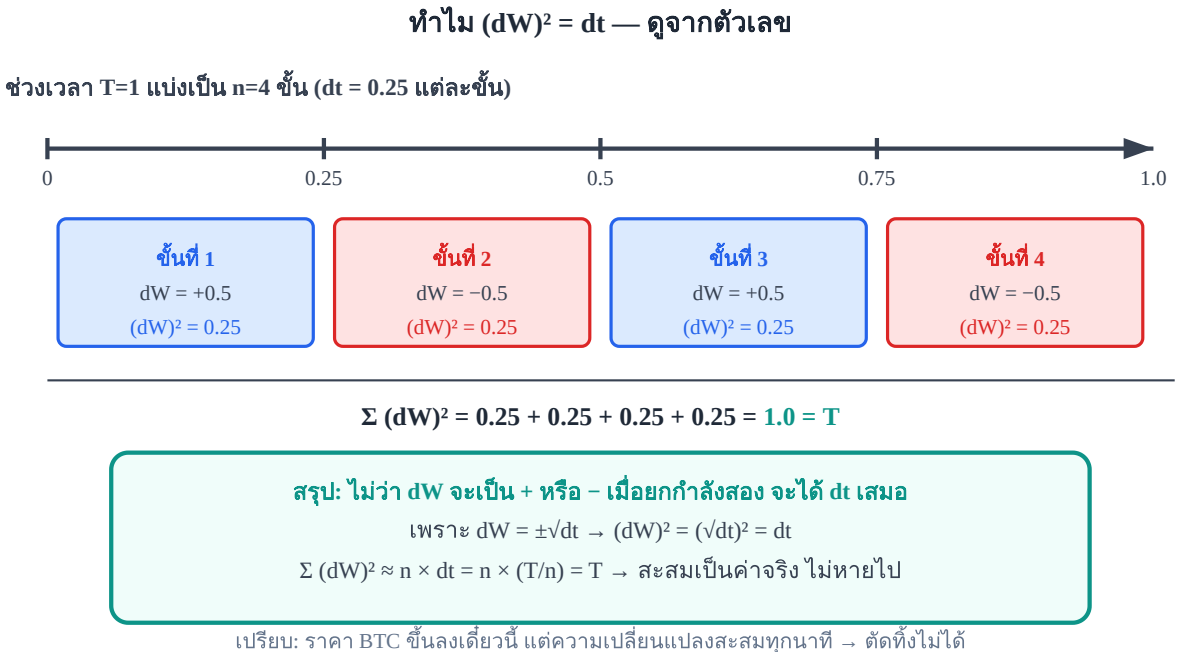
ถ้า x เดินแบบ smooth: $dx = v \cdot dt$
$(dx)^2 = v^2 \cdot (dt)^2 \to$ ตัดทิ้งได้ เพราะ dt^2 เล็กมาก

เช่น $dt = 0.001 \to dt^2 = 0.000001 \approx 0$

Stochastic: $(dW)^2 = dt$ (ตัดไม่ได้!)

W เดินแบบ random: $dW \approx \pm\sqrt{dt}$
$(dW)^2 \approx (\pm\sqrt{dt})^2 = dt \to$ **ไม่ใช่ dt^2 แต่เป็น dt**

เช่น $dt = 0.001 \to (dW)^2 \approx 0.001 \neq 0$



เปรียบ: ราคา BTC ขึ้นลงเดี๋ยวนี้ แต่ความเปลี่ยนแปลงสะสมทุกนาที → ตัดทิ้งไม่ได้

สัญชาตญาณ: dW ขยับสลับ +/− แต่ $(dW)^2$ เป็นบวกเสมอ → สะสมได้ → ไม่ตัดทิ้ง

2.3 Stochastic Differential Equation (SDE)

SDE คือสมการที่บอกว่า process X เปลี่ยนอย่างไรในแต่ละช่วง dt:

รูปแบบ SDE ทั่วไป
$dX_t = \mu(X,t)\, dt + \sigma(X,t)\, dW_t$

แยกส่วนของ SDE

ตัวอย่าง SDE ที่สำคัญ

GBM (Black-Scholes):	OU Process (Spread):	Random Walk:
$\mu=\mu S,\ \sigma=\sigma S$	$\mu=\kappa(\theta-\varepsilon),\ \sigma=\sigma$	$\mu=0,\ \sigma=\sigma$

OU Process: drift = $\kappa(\theta-\varepsilon)$ → ดึงกลับเมื่อ ε ห่างจาก θ | Random Walk: drift = 0 → ไม่มีแรงดึง

2.4 Itô's Lemma — Chain Rule ของ Stochastic World

ถ้าเราต้องการหา SDE ของ **f(X)** เมื่อรู้ SDE ของ **X** ใช้ Itô's Lemma:

Itô's Lemma
$df = (\partial f/\partial t + \mu\cdot\partial f/\partial X + \tfrac{1}{2}\sigma^2\cdot\partial^2 f/\partial X^2)\, dt + \sigma\cdot\partial f/\partial X \cdot dW$

เทียบกับ Chain Rule ธรรมดา ทีละขั้น

#	Calculus ธรรมดา	Itô's Lemma (Stochastic)	เหตุผลที่ต่างกัน
1	$df = (\partial f/\partial t)dt + (\partial f/\partial x)dx$	เริ่มเหมือนกัน	—
2	แทน $dx = \mu dt + \sigma dW$	แทน $dX = \mu dt + \sigma dW$	—
3	คิด Taylor: $(dx)^2 \to 0$ ตัดทิ้ง	คิด Taylor: $(dX)^2 = \sigma^2 dt$ **ไม่ตัด!**	$(dW)^2=dt$ ไม่ใช่ 0
4	$df = (\partial f/\partial t + \mu\partial f/\partial x)dt + \sigma\partial f/\partial x\, dW$	$df = (\partial f/\partial t + \mu\partial f/\partial X + \mathbf{\tfrac{1}{2}\sigma^2\partial^2 f/\partial X^2})dt + \sigma\partial f/\partial X\, dW$	Term พิเศษ $\tfrac{1}{2}\sigma^2(\partial^2 f/\partial X^2)dt$

Term พิเศษ $\tfrac{1}{2}\sigma^2(\partial^2 f/\partial X^2)dt$ มาจากไหน?

Taylor expansion ของ f(X+dX): $f(X+dX) = f(X) + f'\cdot dX + \tfrac{1}{2}f''\cdot(dX)^2 + \ldots$

ใน calculus ธรรมดา: $(dX)^2 = (\mu dt + \sigma dW)^2 \approx \sigma^2(dW)^2$ → ถ้า $(dW)^2=0$ ก็ตัดทิ้ง

ใน stochastic: $(dW)^2 = dt$ → $\tfrac{1}{2}f''\cdot\sigma^2\cdot dt$ **ไม่หายไป** → กลายเป็น term ใหม่ในสมการ

ตัวอย่างสำคัญ: log(S) เมื่อ S ~ GBM

1
กำหนด: $dS = \mu S\, dt + \sigma S\, dW$, $f(S) = \log(S)$
$\partial f/\partial S = 1/S$, $\partial^2 f/\partial S^2 = -1/S^2$
2

แทนใน Itô's Lemma:
$d(\log S) = [\mu S\cdot(1/S) + ½(\sigma S)^2\cdot(-1/S^2)]\, dt + \sigma S\cdot(1/S)\, dW$
$= [\mu - ½\sigma^2]\, dt + \sigma\, dW$
3
ความหมาย: $\log(S)$ มี drift $= \mu - ½\sigma^2$ (ไม่ใช่ μ)
Term $-½\sigma^2$ คือ "Itô correction" — เกิดจาก Jensen's inequality บนฟังก์ชัน concave log
Itô correction สำคัญสำหรับ Stat Arb อย่างไร?

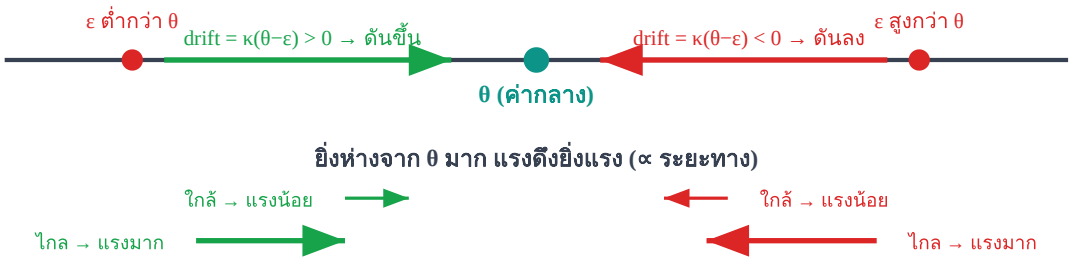
ถ้าใช้ log price spread: $\varepsilon = \log(P_A) - \beta\cdot\log(P_B)$
drift ของ ε จาก Itô $= (\mu_A - ½\sigma_A^2) - \beta(\mu_B - ½\sigma_B^2)$
ถ้าไม่รวม Itô correction → ประเมิน drift ผิด → เลือก β ผิด → ε drift ไม่หาย

2.5 Preview: OU Process — SDE ที่เราต้องการ

สิ่งที่เราต้องการสำหรับ spread ε คือ SDE ที่มี drift "ดึงกลับ" เมื่อห่างจากค่ากลาง:

Ornstein-Uhlenbeck Process
$d\varepsilon = \kappa(\theta - \varepsilon)\, dt + \sigma\, dW$

drift ของ OU Process: κ(θ−ε)



เหมือนสปริง: ยิ่งดึง ยิ่งสปริงออกแรงกลับมากขึ้น (Hooke's Law)

OU drift ทำงานเหมือนสปริง — ยิ่งดึง spread ออกไปไกล แรงดึงกลับยิ่งมาก
โจทย์ 2.1 — $(dW)^2 = dt$

สมมติแบ่งช่วงเวลา T=2 ชั่วโมง เป็น n=8 ขั้น (dt = 0.25 ชม.) และ $\sigma = 0.1\%$ ต่อ √ชม.

ถาม: (a) dW แต่ละขั้นมี std dev เท่าไร? (b) $\Sigma(dW)^2$ ทั้งหมดมีค่าเท่าไร (โดย E)?

▶ ดูคำตอบ
โจทย์ 2.2 — Itô's Lemma

กำหนด $dX = 3\, dt + 2\, dW$ และ $f(X) = X^2$

ถาม: หา df โดยใช้ Itô's Lemma และบอกว่า Itô correction เท่าไร

▶ ดูคำตอบ

โจทย์ 2.3 — อ่าน OU drift

Spread ε เดิน OU: $d\varepsilon = 2(0.001 - \varepsilon)\,dt + 0.003\,dW$
ขณะนี้ $\varepsilon = 0.005$

ถาม: drift ตอนนี้เท่าไร (ต่อวัน) และบอกทิศทางและขนาด

▶ ดูคำตอบ

Ch.2 Stochastic Calculus | ต่อไป → **Ch.3: Ornstein-Uhlenbeck Process**

Statistical Arbitrage · บทที่ 3

Ornstein-Uhlenbeck Process

หัวใจของ Stat Arb — โมเดลที่อธิบายว่า spread "วนกลับ" อย่างไร
κ (speed) · θ (mean) · σ (noise) · Half-life · Calibration
แก่นของบทนี้ — อ่านก่อน

OU Process คือ "สปริงที่มี noise" — ยิ่งดึง spread ออกจากค่ากลาง สปริงยิ่งดึงกลับแรง
Half-life บอกว่า spread ใช้เวลากลับครึ่งทางนานแค่ไหน → กำหนดว่าเราจะ trade แบบ intraday หรือ swing

$d\varepsilon = \kappa(\theta - \varepsilon)\, dt + \sigma\, dW$

3.1 Parameters ของ OU Process

Parameter	ชื่อ	ความหมาย trade	ค่าปกติ
$\kappa > 0$	Mean-reversion speed	ยิ่งมาก spread กลับเร็ว → ต้อง execute เร็ว	1–5 ต่อวัน
θ	Long-run mean	ค่ากลางที่ entry/exit อิงอยู่ — ถ้าเปลี่ยนบ่อย ต้อง re-calibrate	≈0 หรือ avg funding diff
$\sigma > 0$	Diffusion (noise)	ยิ่งมาก spread ผันผวนมาก → threshold ต้องกว้างกว่า fee	0.01–0.5% ต่อ √วัน

3.2 OU Solution — ε เดินไปไหนได้บ้าง

SDE นี้ solve ได้ closed-form ทำให้เราคำนวณ expected value ได้ทุกเวลา:

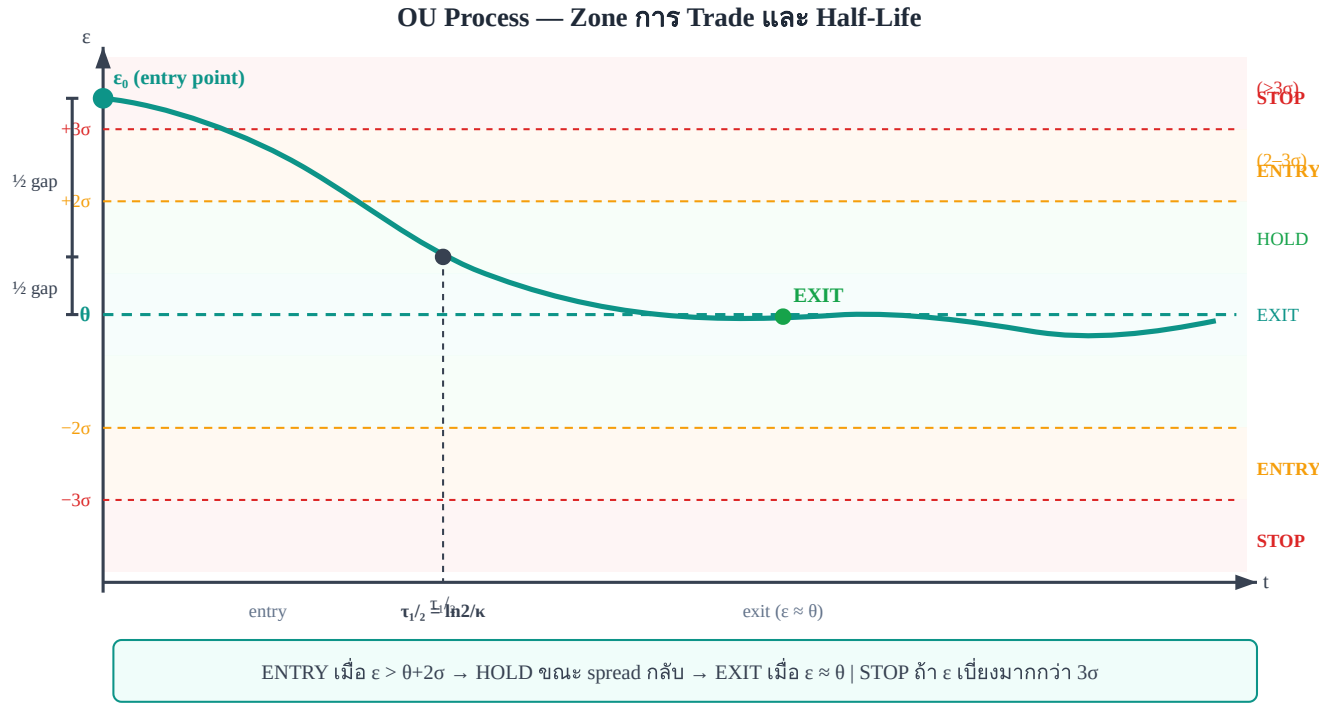
$\varepsilon_t = \theta + (\varepsilon_0 - \theta)\cdot e^{-\kappa t} + \sigma\cdot\int_0^t e^{-\kappa(t-s)}\, dW_s$ ↑ ↑ ↑ ค่ากลาง decay จากจุดเริ่ม noise สะสม
$E[\varepsilon_t \mid \varepsilon_0] = \theta + (\varepsilon_0 - \theta)\cdot e^{-\kappa t}$

เริ่มที่ $\varepsilon_0 = \theta + gap_0$ (ห่างจากค่ากลาง gap_0)
หลังเวลา t: ε คาดว่าจะอยู่ที่ $\theta + gap_0\cdot e^{-\kappa t}$
gap ลดลงด้วยอัตรา $e^{-\kappa t}$ → ยิ่งนาน ยิ่งใกล้ θ

3.3 Half-Life — ตัวเลขที่สำคัญที่สุดสำหรับ Trader

Half-Life
$\tau_{1/2} = \ln(2) / \kappa \approx 0.693 / \kappa$

เวลาที่ gap ระหว่าง ε และ θ ลดลงครึ่งหนึ่ง
ถ้า spread เบี่ยง 1% จาก θ → หลัง $\tau_{1/2}$ จะเหลือ 0.5%



Half-life = ช่วงเวลาที่ spread เดินทางครึ่งทางกลับ — ใช้วางแผน expected holding time

Half-life กับประเภท Strategy

κ (ต่อวัน)	Half-life	ประเภท Trade	ข้อกำหนด Execution
0.1	~7 วัน	Position trade / swing	ไม่เร่ง ต้องมี margin buffer
0.5	~33 ชม.	Swing ข้ามวัน	Monitor วันละครั้ง
1–2	**8–17 ชม.**	**Intraday (เหมาะสุด)**	**Check ทุก 1–2 ชม.**
5	~3 ชม.	Fast intraday	Algo-assisted execution
10+	<2 ชม.	Near HFT	ต้องการ co-location, API

3.4 Stationary Distribution — Band ของ Trade

เมื่อ t → ∞ OU process เข้าสู่ stationary distribution ซึ่งเป็น Normal:

$\varepsilon_\infty \sim N(\theta, \sigma^2/(2\kappa))$ $\sigma_\varepsilon = \sigma/\sqrt{(2\kappa)}$ ← stationary std dev (ใช้ตั้ง entry band)
σ_ε ใช้ทำอะไร?

- Entry threshold: $\varepsilon > \theta + 2\sigma_\varepsilon$ หรือ $\varepsilon < \theta - 2\sigma_\varepsilon$ (เกิดราว 5% ของเวลา)
- Exit threshold: $\varepsilon \approx \theta \pm 0.5\sigma_\varepsilon$
- Stop-loss: $|\varepsilon - \theta| > 3\sigma_\varepsilon$ (spread "หลุด" ผิดปกติ)
- ตรวจสอบ: ε ควรอยู่ในช่วง $\theta \pm 3\sigma_\varepsilon$ กว่า 99.7% ของเวลา — ถ้าไม่ใช่ = signal ให้ re-calibrate

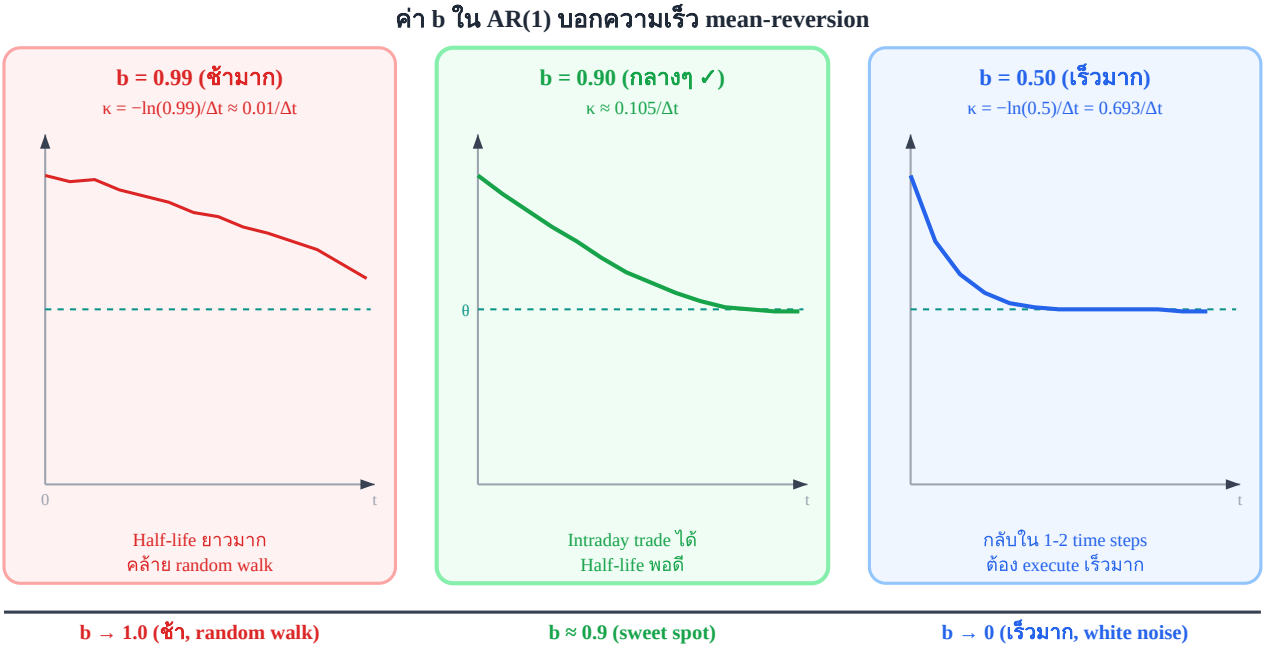
3.5 AR(1): OU Process ในโลกข้อมูล Discrete

ข้อมูลจริงเป็น discrete (รายนาที รายชั่วโมง) เมื่อแปลง OU เป็น AR(1):

OU → AR(1) (time step Δt)
$\varepsilon_t = a + b \cdot \varepsilon_{t-1} + \eta_t$

```
b = e^{−κΔt} ← mean-reversion coefficient
a = θ(1−b) ← intercept
η_t ~ N(0, σ²(1−b²)/(2κ)) ← noise
```

อ่าน b บน Scale — b บอกอะไร?



b ≈ 0.9–0.95 ต่อ time step มักเป็น sweet spot สำหรับ intraday stat arb

สูตรแปลง AR(1) → OU parameters
$\kappa = -\ln(b) / \Delta t \qquad \theta = a / (1 - b)$

ตัวอย่าง: b = 0.90, Δt = 1 ชม. = 1/24 วัน

$\kappa = -\ln(0.90) / (1/24)$ → $-\ln(0.90) =$ **0.1054** → $0.1054 \times 24 =$ **2.53 ต่อวัน**

Half-life = ln(2) / 2.53 = **0.274 วัน = 6.6 ชม.**

3.6 Calibration — หา κ, θ, σ จากข้อมูลจริง

เราเพิ่งเรียนรู้ว่า AR(1) ที่ fit บน spread ε_t ก็คือ OU process ในรูปแบบ discrete — ดังนั้นวิธีหา κ, θ, σ จากข้อมูลจริงคือ **fit AR(1) แล้วแปลงค่า** ตามขั้นตอนนี้:

1
สร้าง spread series: $\varepsilon_t = \log(P_{Bybit,t}) - \beta \cdot \log(P_{Lighter,t})$
(β ได้จาก OLS บน log returns — บทที่ 4)
2

Regress AR(1): $\varepsilon_t = a + b\cdot\varepsilon_{t-1} + \eta_t$
บันทึก: a, b, และ residuals η
3
คำนวณ OU parameters:
$\theta = a/(1-b)$, $\kappa = -\ln(b)/\Delta t$, $\sigma_{stat} \approx \mathrm{std}(\varepsilon - \theta)$
หมายเหตุ: $\mathrm{std}(\varepsilon - \theta)$ เป็นตัวประมาณค่า (sample estimator) ของ $\sigma_{stat} = \sigma/\sqrt{(2\kappa)}$ ตัวเดียวกันจาก §3.4 — ทั้งสองค่าจะลู่เข้าหากันเมื่อจำนวนข้อมูลมากพอ
4
คำนวณ trade parameters:
$\tau_{1/2} = \ln(2)/\kappa$ → กำหนด holding time
Entry band = $\theta \pm 2\sigma_{stat}$ → กำหนด entry threshold

```
# Pseudo-code: OU Calibration
function calibrate_ou(eps_series, delta_t):
  # Step 1: AR(1) regression
  y = eps_series[1:] # ε_t
  x = eps_series[:-1] # ε_{t-1}
  b, a = ols_slope_intercept(x, y)
  residuals = y - (a + b * x)

  # Step 2: OU parameters
  theta = a / (1 - b)
  kappa = -log(b) / delta_t
  sigma_stat = std(eps_series - theta)
  half_life = log(2) / kappa

  # Step 3: Trade bands
  entry_upper = theta + 2 * sigma_stat
  entry_lower = theta - 2 * sigma_stat
  stop_upper = theta + 3 * sigma_stat
  stop_lower = theta - 3 * sigma_stat

  return {theta, kappa, sigma_stat, half_life,
        entry_upper, entry_lower, stop_upper, stop_lower}
```

ตัวอย่างจริง: Bybit↔Lighter BTC — Calibration ผล

ข้อมูล: 30 วัน, รายนาที (Δt = 1/1440 วัน) $\varepsilon_t = \log(P_{Bybit}) - 1.003\cdot\log(P_{Lighter})$ OLS AR(1) result: b = 0.9985, a = 2.0×10^{-6} Residual std = 0.00012 OU Parameters: θ = 2.0e-6 / (1−0.9985) = 0.00133 (basis ≈ 0.133%) $\kappa = -\ln(0.9985)/(1/1440) = 2.16$ ต่อวัน $\sigma_{stat} \approx 0.00080$ (≈ 0.08%) Trade Parameters: $\tau_{1/2} = 0.693/2.16 = 7.7$ ชั่วโมง ← intraday trade ได้ Entry: $\varepsilon > 0.293\%$ หรือ $\varepsilon < -0.027\%$ ($\theta \pm 2\times0.08\%$) Stop: $\varepsilon > 0.373\%$ หรือ $\varepsilon < -0.107\%$ ($\theta \pm 3\times0.08\%$)

สรุป: spread กลับใน ~8 ชม. เปิด position ตอนเช้า → ปิดก่อนค่ำ

3.7 เมื่อไร OU เดี่ยวไม่พอ — และบทถัดไปจะแก้อย่างไร

สมมติฐาน OU	เมื่อไรที่ผิด	สัญญาณที่เห็น	โมเดลที่ใช้แทน

κ, θ คงที่	ตลาด regime change: funding rate เปลี่ยนโครงสร้าง	θ drift ออก, ε ไม่กลับ	Kalman (Ch.15), Regime-Switch (Ch.9)
σ คงที่	ช่วง volatile: market crash, news	residual variance พุ่ง	GARCH on residuals (Ch.7)
Gaussian noise	liquidation cascade, news spike	residual มี fat tail, jump	Jump-Diffusion OU (Ch.8)
β คงที่	ตลาดเปลี่ยน correlation	ε drift กลับมา	Kalman Filter β (Ch.15)

Rule of thumb: เมื่อไรต้อง re-calibrate?

- θ เคลื่อนออกจากค่าเดิม > 1σ_stat ใน 5 วัน → re-calibrate β และ θ
- Residual variance พุ่ง > 2× baseline → switch ไป GARCH
- Half-life ยาวขึ้น > 2× → regime เปลี่ยน พิจารณา exit ทั้งหมด

3.8 ตารางสัญลักษณ์ (Notation Reference) — ใช้ตลอดทั้งเล่ม

ทั้งเล่มใช้ σ หลายแบบ ต่างกันที่จุดประสงค์และวิธีคำนวณ — ตารางนี้เป็นอ้างอิงกลางที่ควรกลับมาดูทุกครั้งที่สับสน:

สัญลักษณ์	ชื่อเต็ม	ความหมาย	คำนวณจาก	ใช้ใน
σ (SDE)	Diffusion coefficient	ขนาด noise ใน OU SDE: dε = κ(θ−ε)dt + σ·dW	σ = σ_stat · √(2κ)	Ch.2–3 สมการทฤษฎี
σ_stat	Stationary std	Std ของ spread ε ณ steady state — ใช้กำหนด entry/exit band	σ_stat = σ/√(2κ) ≈ std(ε − θ)	Ch.3–4 entry band θ ± 2σ_stat
σ_t (GARCH)	Conditional std	Std ของ spread ε ณ เวลา t — ขยายตัวช่วง volatile	$\sigma_t^2 = \omega + \alpha\cdot\varepsilon^2_{t-1} + \beta_G\cdot\sigma^2_{t-1}$	Ch.7 GARCH z-score
σ_robust	Robust std (MAD-based)	Estimate σ_stat ที่ทนต่อ outlier — ไม่กระทบจาก jump	σ_robust = MAD × 1.4826	Ch.10 robust z-score
σ_EMA	EMA-smoothed std	Rolling std ด้วย exponential weighting — Welford-style	Welford online update (Ch.15)	Ch.15 dynamic β tracking

กฎง่าย ๆ ในการเลือก σ

- **ตลาดปกติ**: ใช้ σ_stat (OLS residual std) — เร็วและง่าย
- **ช่วง volatile** (ratio 7d/30d std > 1.5): เปลี่ยนเป็น σ_t จาก GARCH(1,1)
- **มี outlier / jump**: ใช้ σ_robust = MAD × 1.4826 แทน std()
- **β เปลี่ยนตาม Kalman**: ε เปลี่ยนทุก bar → ใช้ σ_EMA แบบ Welford update

สัญลักษณ์อื่น ๆ	ความหมาย

ε_t	Spread residual at time t: ε = log(P_A) − β·log(P_B)
θ	Long-run mean ของ OU (equilibrium spread level)
κ	Mean-reversion speed (ต่อวัน); $\tau_{1/2}$ = ln(2)/κ
β	Hedge ratio (OLS, TLS, หรือ Kalman)
H	Hurst exponent; H < 0.5 = mean-reverting
z_t	Z-score ของ spread: (ε_t − θ)/σ_stat หรือ (ε_t − θ)/σ_t
$\tau_{1/2}$	Half-life ของ mean-reversion: $\tau_{1/2}$ = ln(2)/κ
Δt	Bar size ในหน่วยวัน (1-hour bar: Δt = 1/24)

3.9 Walk-Forward Calibration — ป้องกัน Overfitting

แก่นความคิด — อ่านก่อน

การ backtest แบบ **Static Window คือดู historical data ครั้งเดียว → รู้อนาคตโดยไม่รู้ตัว (lookahead bias)**
Walk-Forward แก้ปัญหานี้โดยแบ่งเวลาแบบ rolling ลำดับเสมอ — เราใช้ข้อมูลอดีตเพื่อ fit แล้วทดสอบบนข้อมูลอนาคตที่โมเดลยังไม่เคยเห็น

Concept Drift — ทำไม Parameters เปลี่ยนตลอดเวลา

OU parameters (κ, θ, σ) ไม่คงที่ตลอดกาล ตลาดเปลี่ยนแปลงตามสภาวะเศรษฐกิจ ความเชื่อมั่นของนักลงทุน และโครงสร้าง funding rate — สิ่งนี้เรียกว่า **Concept Drift**

- **Static window**: fit OU บนข้อมูลทั้งหมดครั้งเดียว → parameters เป็นค่าเฉลี่ยของทุกช่วงเวลา → overfits กับช่วงอดีตที่ตลาดอาจไม่มีอีกแล้ว
- **Walk-Forward**: fit OU บน window อดีต → apply บน window อนาคตที่ยังไม่เคยเห็น → สะท้อน parameters ของตลาด ณ ช่วงนั้นจริง ๆ

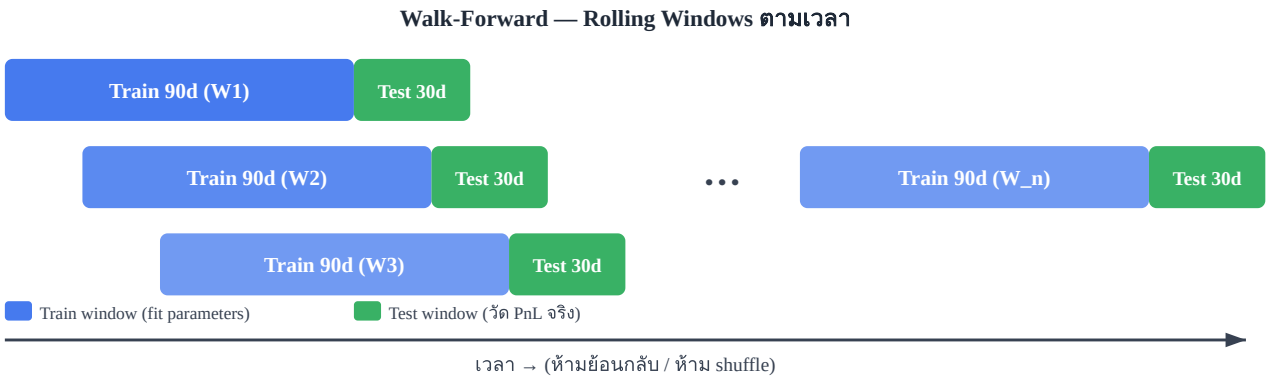
อันตรายของ Static Calibration

- θ ที่ fit ได้อาจเป็นค่าเฉลี่ยของ regime หลายช่วงที่ไม่เคยเกิดพร้อมกัน
- κ ที่ได้อาจสูงเกินจริง เพราะ data ในช่วง volatile ดูเหมือน mean-revert เร็ว
- เมื่อ trade จริง parameters เหล่านั้นไม่ตรงกับตลาดปัจจุบัน → signal ผิด → ขาดทุน

Walk-Forward Algorithm — 3 ขั้นตอนหลัก

1
แบ่ง Window: Train window = 90 วัน, Test window = 30 วัน
เริ่มที่วันที่ 1–90 เป็น train, วันที่ 91–120 เป็น test — **ห้าม shuffle หรือ random split เด็ดขาด** ต้องเรียงตามเวลาเสมอ
2

Fit แล้ว Apply: Calibrate OU parameters บน train window → นำ parameters ไป generate signals บน test window → บันทึก PnL ของ test window
ไม่มี lookahead: test window ถูก "เปิดเผย" ทีละ bar ไม่ใช่ทั้งหมด
3
เลื่อน Window: ขยับ window ไป 30 วัน (= test_days) → วนซ้ำขั้นตอน 1–2
แต่ละรอบให้ PnL 1 window — รวมทุก window ได้ equity curve ที่ไม่มี lookahead



แต่ละ train block (สีน้ำเงิน) ให้ parameters → apply บน test block (สีเขียว) ที่อยู่ถัดไปในอนาคต — เลื่อนซ้ำตลอดช่วงข้อมูล

Pseudo-code — Walk-Forward Calibration

def walk_forward_calibration(prices_A, prices_B, train_days=90, test_days=30): results = [] i = 0 while i + train_days + test_days <= len(prices_A): train_A = prices_A[i : i+train_days] train_B = prices_B[i : i+train_days] test_A = prices_A[i+train_days : i+train_days+test_days] test_B = prices_B[i+train_days : i+train_days+test_days] # Fit OU parameters on train beta = estimate_beta_ols(train_A, train_B) epsilon_train = log(train_A) - beta * log(train_B) kappa, theta, sigma_stat = calibrate_ou(epsilon_train) # Apply on test (no lookahead) epsilon_test = log(test_A) - beta * log(test_B) signals = generate_signals(epsilon_test, kappa, theta, sigma_stat) pnl = simulate_pnl(signals, test_A, test_B) results.append({"window_start": i, "beta": beta, "kappa": kappa, "pnl": pnl}) i += test_days # slide forward return results
จุดสำคัญที่ต้องระวัง

- `i += test_days` — เลื่อน window ด้วย test_days ไม่ใช่ 1 วัน เพื่อให้แต่ละ test window ไม่ overlap กัน
- Train window ขนาด 90 วัน ต้องมี data เพียงพอสำหรับ AR(1) regression — น้อยกว่า 30 obs จะไม่ reliable
- ห้าม reuse test data เป็น train ของ window เดิม — window ต้องเลื่อนเป็นลำดับเสมอ

ตัวอย่างจริง: Bybit↔Lighter BTC — Walk-Forward 360 วัน

ใช้ข้อมูล 360 วัน → walk-forward ได้ **9 windows** (train=90d, test=30d, slide=30d)

Window 1: train day 1–90, test day 91–120 → half-life = 6.1h Window 2: train day 31–120, test day 121–150 → half-life = 5.2h Window 3: train day 61–150, test day 151–180 → half-life = 8.7h Window 4: train day 91–180, test day 181–210 → half-life = 9.4h Window 5: train day 121–210, test day 211–240 → half-life = 7.1h Window 6: train day 151–240, test day 241–270 → half-life = 11.3h Window 7: train day 181–270, test day 271–300 → half-life = 6.8h Window 8: train day 211–300, test

day 301–330 → half-life = 5.9h Window 9: train day 241–330, test day 331–360 → half-life = 6.5h เฉลี่ย: 7.4h | Range: 5.2–11.3h

Half-life เฉลี่ยจาก 9 windows = **7.4 ชั่วโมง** (range: 5.2–11.3h) — ความผันผวนของ half-life ระหว่าง windows แสดงว่า OU parameters เปลี่ยนตามช่วงเวลาจริง → **justifies การใช้ walk-forward แทน static calibration** ถ้าใช้ static จะได้ค่าเดียวตลอด ซึ่งผิดพลาดในหลายช่วง
โจทย์ 3.4 — Walk-Forward Window Count

มีข้อมูล **180 วัน**, ตั้งค่า train = 90 วัน, test = 30 วัน, slide = 30 วัน

ถาม: (a) ได้กี่ windows? (b) first trade เริ่มวันที่เท่าไหร่? (c) ถ้าลด train เหลือ 60 วัน จะได้กี่ windows?

▶ ดูคำตอบ
โจทย์ 3.1 — คำนวณ half-life และ trade parameters

OLS AR(1) บน spread รายชั่วโมง (Δt = 1/24 วัน): b = 0.94, a = 0.00012
std(residuals) = 0.00015

ถาม: (a) κ ต่อวัน (b) half-life เป็นชั่วโมง (c) entry band (d) spread ปัจจุบัน 0.05% จาก θ — ควร enter ไหม?

▶ ดูคำตอบ
โจทย์ 3.2 — แปลง b → strategy design

Spread รายชั่วโมง ให้ผล AR(1): b = 0.97
ค่า fee รวม (open+close) = 0.06% per round-trip

ถาม: (a) half-life (b) ถ้าตั้ง entry ที่ $2\sigma_{stat}$ = 0.10% จะถือ position นานแค่ไหนโดยเฉลี่ย และ expected profit คืออะไร (ก่อน fee)?

▶ ดูคำตอบ
โจทย์ 3.3 — วินิจฉัย OU breakdown

ใช้ OU trade Bybit↔Lighter มา 2 สัปดาห์ κ เดิม = 2 ต่อวัน, θ = 0.1%
สัปดาห์ที่ 3: spread ออกไปถึง 0.4% และค้างอยู่นาน 3 วัน ไม่กลับ

ถาม: เกิดอะไรขึ้น และควรทำอะไร?

▶ ดูคำตอบ

Ch.3 OU Process | ต่อไป → **Part II: Hedge Ratio β (Ch.4–6)**

Statistical Arbitrage · บทที่ 4

Hedge Ratio β

OLS · TLS · Kalman Filter — สามวิธีหา "อัตราส่วนที่ถูกต้อง"
ตัวอย่าง: Bybit ↔ Lighter BTC Perpetual
แก่นของบทนี้

β คือตัวเลขที่ทำให้ ε = log(A) − β·log(B) เป็น stationary
วิธีหา β มีผลต่อ entry ratio และ P&L โดยตรง — OLS, TLS, และ Kalman ให้ β ต่างกัน

β_OLS = Cov(r_A, r_B) / Var(r_B)

4.1 ทำไมวิธีหา β จึงสำคัญ

จากบทที่ 1 เราทราบแล้วว่า β ≠ 1 — แต่คำถามคือ **หา β ด้วยวิธีไหน?** วิธีต่างกันให้ β ต่างกัน และ β ที่ต่างกันหมายถึงอัตราส่วน trade ต่างกัน → P&L ต่างกัน

OLS
Ordinary Least Squares
เร็ว, simple, bias ถ้าทั้งสองตัวมี noise

TLS
Total Least Squares
Symmetric, เหมาะเมื่อทั้งคู่ noisy เท่ากัน

Kalman
Dynamic β
β เปลี่ยนตามเวลา, แม่นยำสุดแต่ซับซ้อนกว่า

4.2 OLS — วิธีที่ใช้บ่อยที่สุด

OLS หา β ที่ minimize ระยะทางแนวตั้ง (vertical residual) จากจุดข้อมูลถึงเส้น regression:

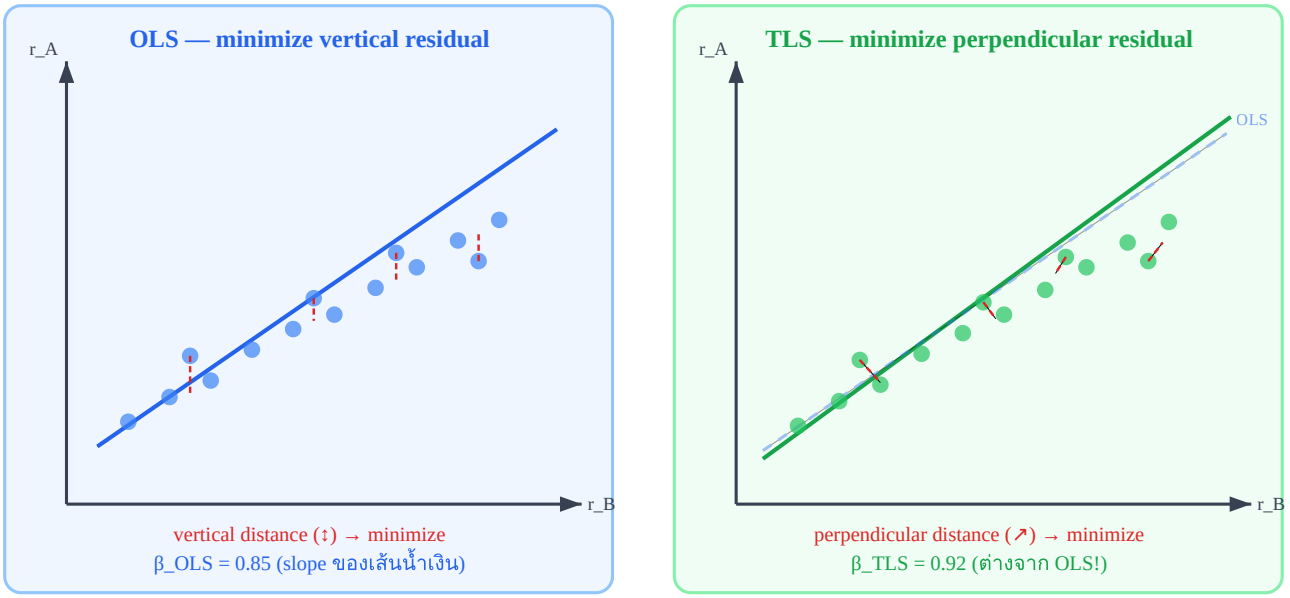
สูตร OLS β
β_OLS = Cov(r_A, r_B) / Var(r_B) = Σ(r_A − r̄_A)(r_B − r̄_B) / Σ(r_B − r̄_B)²

หรือถ้า regress log(P_A) บน log(P_B) โดยตรง (level regression):

log(P_A) = α + β·log(P_B) + ε β_OLS = slope ของ regression นี้

หมายเหตุ: β จาก return regression และ β จาก price-level regression เป็นคนละค่ากัน — สำหรับ synthetic process ε = log(P_A) − β·log(P_B) ค่า β ที่เป็นหลัก (canonical) คือ slope ของ log-price regression

OLS vs TLS — ต่างกันที่ "minimize อะไร"



OLS และ TLS ให้ β ต่างกัน — OLS ใช้ B เป็น "หลัก" (no noise), TLS สมมติทั้งคู่ noisy เท่ากัน

สูตร TLS (Total Least Squares)

$\beta_TLS = [Var(r_A) - Var(r_B) + \sqrt{((Var(r_A)-Var(r_B))^2 + 4\cdot Cov(r_A,r_B)^2)}] / (2\cdot Cov(r_A,r_B))$
หรือใช้ SVD: $\beta_TLS = v_{12}/v_{22}$ (eigenvector ของ covariance matrix [r_A, r_B])

เลือก OLS หรือ TLS?

สถานการณ์	แนะนำ	เหตุผล
B เป็น "benchmark" (spot price, reference)	**OLS**	B มี noise น้อยกว่า A — OLS assume B error-free
ทั้งสองตลาด liquidity ใกล้เคียงกัน	**TLS**	ทั้งคู่ noisy พอๆ กัน — symmetric
ต้องการ β เสถียร ปรับน้อย	**OLS rolling**	คำนวณง่าย เปลี่ยนได้ตามรอบ
β เปลี่ยนเร็ว ต้องการ adaptive	**Kalman**	อัปเดตทุก tick

4.2b OLS เชิงลึก — จากหลักการแรกถึงการใช้งานจริง

OLS · Linear Regression · Linear Programming — คนละเรื่องกัน

สามคำนี้ใช้คำว่า "linear" ร่วมกันแต่มาจากคนละโดเมนกันโดยสิ้นเชิง:

	Linear Regression	OLS	Linear Programming (LP)
คืออะไร	Model — สมมติว่าความสัมพันธ์เป็นเส้นตรง	Method — วิธีหนึ่งที่ใช้ fit model	Optimization — หา solution ที่ดีที่สุดภายใต้ constraint

"Linear" หมายถึง	β เข้า model แบบ linear: y = α + β·x	—	objective + constraints เป็นเส้นตรง
สมการหลัก	y = α + β·x + ε	minimize Σε²	minimize c·x s.t. Ax ≤ b
ใช้ทำอะไร	อธิบายความสัมพันธ์ระหว่างตัวแปร	หา β ที่ดีที่สุดใน linear regression	จัดสรรทรัพยากร, portfolio weight

ความสัมพันธ์ที่มีจริง: OLS เป็น *วิธีหนึ่ง* ในการ fit Linear Regression — ไม่ใช่สิ่งเดียวกัน และ Linear Programming คือคนละโลกกันเลย ไม่เกี่ยวกับ regression เลย

Linear Regression (model) วิธี fit ที่ใช้ได้: ├── OLS → minimize Σε² ← วิธีที่หนังสือนี้ใช้ (quadratic objective) ├── MLE → maximize likelihood ├── Ridge → minimize Σε² + λΣβ² └── LAD → minimize Σ|ε| ← แปลงเป็น LP ได้! แต่ OLS ไม่ใช่ LP

จุดที่สับสนบ่อย: OLS minimize Σε² ซึ่งเป็น quadratic objective — จึงเป็น *Quadratic Program (QP)* ไม่ใช่ Linear Program แม้จะอยู่ใน "linear regression"
หมายเหตุ: หนังสือเล่มนี้ใช้ OLS เท่านั้น — MLE/Ridge/LAD แสดงไว้เพื่อเทียบนิยามเท่านั้น ไม่ได้ใช้ใน practice
จุดประสงค์ของ OLS ใน Stat Arb

เราไม่ได้ทำ OLS เพื่อ "ทำนาย" ราคา A จาก B — เราทำ OLS เพื่อ **แยก common movement ออกจาก idiosyncratic movement** ส่วนที่เหลือหลังแยก (residual ε) คือสิ่งที่เราเทรด

ε_t = log(P_A,t) − α − β·log(P_B,t)
สัญลักษณ์ที่ใช้ใน §4.2b — อ่านก่อนเริ่ม

สัญลักษณ์	ความหมาย	หมายเหตุ
y_t	log(P_A,t) — ราคา A ในรูป log	dependent variable ที่ต้องการอธิบาย
x_t	log(P_B,t) — ราคา B ในรูป log	independent variable ที่ใช้อธิบาย
α	intercept — ค่าคงที่ปรับระดับ	ไม่ใช่ขา trade — เป็นแค่ constant ใน model
β	population slope (ค่าจริงไม่รู้)	เราประมาณ β̂ จากข้อมูล — β̂ คือ estimator
β̂	sample estimate ของ β จาก OLS	ต่างกันทุก dataset — converges → β เมื่อ N→∞
ε_t	residual = y_t − α − β·x_t	สิ่งที่เทรด — ต้องการให้ mean-revert
φ	AR(1) coefficient ของ ε	บทที่ 3 เรียกค่าเดียวกันว่า b: ε_t = a + b·ε_{t-1} + η_t
BLUE	Best Linear Unbiased Estimator	β̂ เป็น BLUE ถ้า Gauss-Markov assumptions ถือ
Gauss-Markov	4 เงื่อนไขที่ทำให้ OLS เหมาะสมที่สุด	ε zero-mean, uncorrelated กับ x, homoskedastic, no autocorrelation

Homoskedastic Var(ε_t) คงที่ทุก t	ในราคาจริง Var(ε) เปลี่ยนตาม vol — heteroskedastic

หมายเหตุ: ทำไมใช้ log price ไม่ใช่ raw price → ดู §4.6.1 (สรุปสั้น: log → β เป็น elasticity ไม่ขึ้นกับ scale ราคา)

OLS จากหลักการแรก — ทำไมต้อง minimize $\Sigma\varepsilon^2$

ปัญหา: กำหนด N คู่ราคา (P_A,1,…,P_A,N) และ (P_B,1,…,P_B,N) — หา α และ β ที่ทำให้ residual ε_t ขึ้นๆ ลงๆ รอบ 0 น้อยที่สุด (ε มี variance ต่ำสุดที่เป็นไปได้)

ทำไมต้อง minimize $\Sigma\varepsilon^2$ ไม่ใช่ $\Sigma|\varepsilon|$?

- **เหตุผล 1 — penalize outlier มากกว่า:** ε=3 ให้ $\varepsilon^2=9$ แต่ |ε|=3 — squared ทำให้จุดที่ห่างจากเส้นมากถูก penalize หนักกว่า ทำให้เส้น regression ไม่เบี่ยงจากกลุ่มข้อมูลหลัก
- **เหตุผล 2 — มี closed-form solution:** $\Sigma\varepsilon^2$ เป็น differentiable ทุกจุด → หา derivative แล้วตั้งเป็น 0 ได้ → ได้สูตร β = Cov/Var โดยตรง ส่วน Σ|ε| ไม่ differentiable ที่ ε=0 → ต้องใช้ linear programming แก้
- **เหตุผล 3 — residual ⊥ x:** $\Sigma\varepsilon^2$ ทำให้ residual uncorrelated กับ x (Gauss-Markov) — สมบัตินี้สำคัญสำหรับ inference และ diagnostics

เขียนปัญหาเป็นสมการ:

ให้ y_t = log(P_A,t) และ x_t = log(P_B,t) ε_t = y_t − α − β·x_t ← residual คือสิ่งที่เราสร้าง objective: minimize S(α, β) = Σ ε_t^2 = Σ $(y_t - \alpha - \beta\cdot x_t)^2$ t=1 t=1

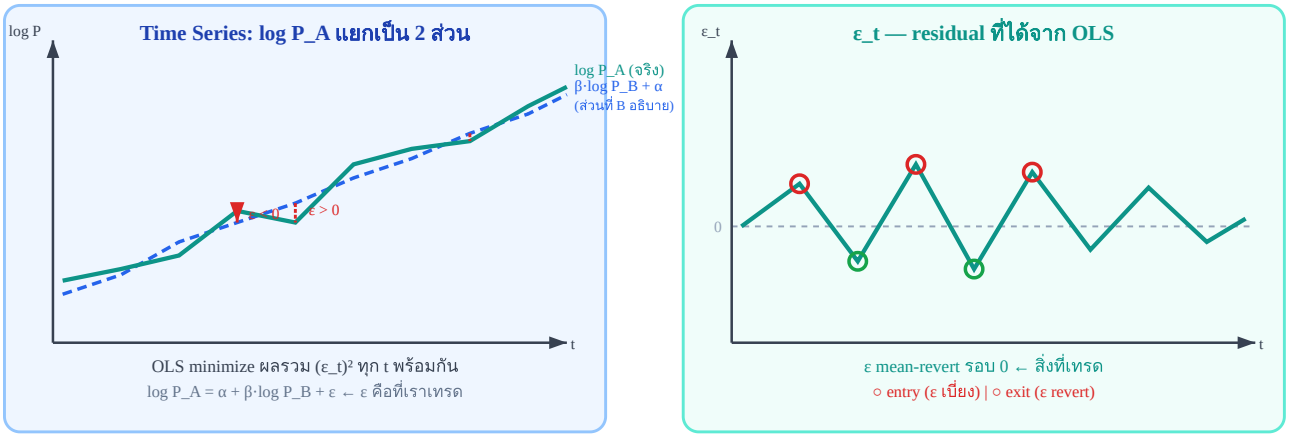
หา minimum โดยหา partial derivative เทียบเท่า 0:

∂S/∂α = −2·Σ(y_t − α − β·x_t) = 0 → Σy_t = N·α + β·Σx_t ∂S/∂β = −2·Σ x_t·(y_t − α − β·x_t) = 0 → Σx_t·y_t = α·Σx_t + β·Σx_t^2 แก้ระบบสมการ (Normal Equations): β = [N·Σ(x_t·y_t) − Σx_t·Σy_t] / [N·Σx_t^2 − $(\Sigma x_t)^2$] = Cov(x, y) / Var(x) ← รูปกระทัดรัด (x = log P_B) α = ȳ − β·x̄ ← α เป็นตัวปรับระดับ
ความหมายของ β จาก OLS

β = Cov(log P_A, log P_B) / Var(log P_B)
ตีความ: **"ถ้า log(P_B) เพิ่มขึ้น 1 หน่วย (≈ B ขยับ +1%) ค่าเฉลี่ยของ log(P_A) ที่ OLS คาดไว้ จะเพิ่มขึ้น β หน่วย"**
หรือพูดในเชิง % ≈ "B ขยับ 1% → A คาดว่าขยับ β%"

ε_t = ส่วนที่ A ขยับ *มากกว่า* หรือ *น้อยกว่า* ที่ B อธิบายได้ — นี่คือ inefficiency ที่เราเทรด

OLS แยก y_t ออกเป็น 2 ส่วน: ส่วนที่ B อธิบาย + residual ε



OLS แยก log P_A ออกเป็น "ส่วนที่ B อธิบาย" + residual ε — ε ที่ดีควร mean-revert รอบ 0

คำนวณ OLS ทีละขั้นตอน — ตัวอย่างตัวเลข

สมมติมีข้อมูล log price รายวัน 6 วัน (ย่อให้คำนวณมือได้):

t	log P_B (x)	log P_A (y)	x − x̄	y − ȳ	(x−x̄)(y−ȳ)	(x−x̄)²
1	10.000	7.225	−0.200	−0.125	0.0250	0.0400
2	10.050	7.265	−0.150	−0.085	0.0128	0.0225
3	10.100	7.285	−0.100	−0.065	0.0065	0.0100
4	10.200	7.385	+0.000	+0.035	0.0000	0.0000
5	10.350	7.435	+0.150	+0.085	0.0128	0.0225
6	10.500	7.505	+0.300	+0.155	0.0465	0.0900
รวม	**61.200**	**44.100**	**—**	**—**	**Σ = 0.1035**	**Σ = 0.1850**

ขั้น 1 — คำนวณ mean: x̄ = 61.200/6 = 10.200 (log P_B เฉลี่ย) ȳ = 44.100/6 = 7.350 (log P_A เฉลี่ย) ขั้น 2 — คำนวณ β (slope): β = Σ(x−x̄)(y−ȳ) / Σ(x−x̄)² = 0.1035 / 0.1850 = 0.5595 ≈ 0.56 ขั้น 3 — คำนวณ α (intercept): α = ȳ − β·x̄ = 7.350 − 0.5595 × 10.200 = 7.350 − 5.707 = 1.643 ขั้น 4 — คำนวณ ε_t สำหรับแต่ละ t: ε_t = y_t − 1.643 − 0.5595·x_t t=1: ε = 7.225 − 1.643 − 0.5595×10.000 = 7.225 − 1.643 − 5.595 = −0.013 t=2: ε = 7.265 − 1.643 − 5.623 = −0.001 t=3: ε = 7.285 − 1.643 − 5.651 = −0.009 t=4: ε = 7.385 − 1.643 − 5.707 = +0.035 t=5: ε = 7.435 − 1.643 − 5.791 = +0.001 t=6: ε = 7.505 − 1.643 − 5.875 = −0.013 Σε_t = −0.013 − 0.001 − 0.009 + 0.035 + 0.001 − 0.013 = 0.000 ✅

สิ่งที่ OLS รับประกัน (Gauss-Markov Theorem)

Gauss-Markov: ถ้า ε มี zero-mean, ไม่ correlate กับ x, Var(ε) คงที่ (homoskedastic), และไม่มี autocorrelation → β̂ จาก OLS เป็น BLUE (Best Linear Unbiased Estimator) — ดีที่สุดในบรรดา estimator linear ที่ unbiased ทั้งหมด

- Σε_t = 0 เสมอ — residuals sum to zero
- Σ(x_t · ε_t) = 0 — residual uncorrelated กับ x
- β_OLS = BLUE (Best Linear Unbiased Estimator) *ถ้า* Gauss-Markov assumptions ถือ (ε i.i.d., homoskedastic)

- ในราคาจริง assumptions เหล่านี้มักฝ่าฝืน → ดู §4.2b-5 ข้อจำกัด

OLS ใน Python — 3 วิธีทำ

```
### วิธี 1: statsmodels (แนะนำ — ได้ diagnostics ครบ)
import statsmodels.api as sm
import numpy as np

y = np.log(price_A) # log P_A — dependent variable
x = np.log(price_B) # log P_B — independent variable

# add constant → fit α and β together
X = sm.add_constant(x) # X = [1, x] (N×2 matrix)
model = sm.OLS(y, X).fit()

alpha = model.params.iloc[0] # α (intercept) — iloc safe กับทุก statsmodels version
beta = model.params.iloc[1] # β (hedge ratio)
epsilon = model.resid # ε_t (residuals) — ใช้ทดสอบ stationarity

r2 = model.rsquared # R² — explained variance
p_beta = model.pvalues[1] # p-value ของ β (ต้องการ ≪ 0.05)
ci_beta = model.conf_int().iloc[1] # 95% CI ของ β [lower, upper]

print(f"β = {beta:.4f} (p={p_beta:.4f})")
print(f"R² = {r2:.4f}")
print(model.summary()) # สรุปครบ — AIC, residual std, t-stat

### วิธี 2: numpy (เร็ว ง่าย สำหรับ loop)
beta, alpha = np.polyfit(x, y, 1) # fit linear y = β·x + α
epsilon = y - (alpha + beta * x)

### วิธี 3: สูตรโดยตรง (เข้าใจ mechanism)
beta = np.cov(y, x)[0,1] / np.var(x, ddof=1)
alpha = np.mean(y) - beta * np.mean(x)
epsilon = y - alpha - beta * x
```

เลือก window ข้อมูลอย่างไร

- **Train window:** ข้อมูล N วันล่าสุด (เช่น 30 วัน) — fit β บน window นี้เท่านั้น
- **อย่าใช้ข้อมูลอนาคต:** ห้าม fit β บน full dataset แล้วย้อนไปทดสอบ → look-ahead bias
- **Walk-forward:** fit β วันนี้ → ใช้ trade พรุ่งนี้ → roll window ไป 1 วัน → fit ใหม่ (ดู บทที่ 3 §3.9)

ตรวจ ε ที่ได้ — 4 การทดสอบหลัก

หลังจาก fit OLS และได้ ε มาแล้ว ต้องตรวจว่า ε มีคุณสมบัติสำหรับ mean reversion trading:

R^2 คืออะไร — นิยามสั้น

$R^2 = 1 - Var(\varepsilon) / Var(y)$ = สัดส่วน variance ของ y ที่ x อธิบายได้

R^2=0: x ไม่ช่วยอธิบาย y เลย | R^2=1: x อธิบาย y ได้สมบูรณ์ | R^2=0.95: x อธิบายได้ 95%

ใน Stat Arb ต้องการ R^2 สูง → แสดงว่า B อธิบาย A ได้ดี → ε เป็น idiosyncratic จริง ไม่ใช่ noise

การทดสอบ	วิธีทำ	ผ่านเมื่อ	ความหมาย
R^2 และ p-value	`model.rsquared`, `model.pvalues[1]`	$R^2 > 0.7$, $p < 0.05$	β มีนัยสำคัญ, B อธิบาย A ได้ดี
Residual plot	plot ε_t vs t	ε oscillate รอบ 0, ไม่ drift	ε mean-revert ตามที่ต้องการ
ADF / KPSS	`statsmodels adfuller(epsilon)`	ADF $p < 0.05$, KPSS $p > 0.05$	ε stationary (ดู บทที่ 5)
Half-life	AR(1) fit บน ε: ε_t = φ·ε_{t-1} + u_t $\tau_{1/2} = -\ln(2)/\ln(\varphi)$	$\tau_{1/2} \leq$ target hold time	ε revert เร็วพอสำหรับกลยุทธ์ (บทที่ 3)

```
# ตรวจ ε ครบชุด
from statsmodels.tsa.stattools import adfuller
import statsmodels.api as sm

# 1. ADF test
adf_result = adfuller(epsilon, maxlags=1)
print(f"ADF p-value: {adf_result[1]:.4f}") # ต้องการ < 0.05

# 2. AR(1) → Half-life
ar1 = sm.OLS(epsilon[1:], sm.add_constant(epsilon[:-1])).fit()
phi = ar1.params[1] # AR(1) coefficient
half_life = -np.log(2) / np.log(phi) # half-life (same unit as data freq)
print(f"φ = {phi:.4f}, Half-life = {half_life:.1f} bars")

# 3. σ_stat — sample estimator ของ σ/√(2κ) (converges เมื่อ sample size N ใหญ่)
sigma_stat = np.std(epsilon) # use in z-score: z = (ε − mean) / sigma_stat
print(f"σ_stat = {sigma_stat:.4f} ({sigma_stat*100:.2f}%)")
```

Running Example: BTC/ETH — OLS ทีละขั้น

ข้อมูล: log price รายชั่วโมง 30 วัน (720 bars) y = log(ETH_price), x = log(BTC_price) ขั้น 1 — fit OLS: model = OLS(y, add_constant(x)).fit() β_log = 1.183 (elasticity — ETH ขยับ 1.183% ต่อทุก 1% ที่ BTC ขยับ) α = −4.021 (intercept — level adjustment) R^2 = 0.951 ✅ BTC อธิบาย ETH ได้ 95.1% p(β) = 0.000 ✅ β มีนัยสำคัญสูง ขั้น 2 — สร้าง ε: ε_t = log(ETH_t) − (−4.021) − 1.183·log(BTC_t) ขั้น 3 — ตรวจ ε: ADF p-value = 0.043 ✅ stationary (ตรงกับ ch5 §5.7) φ (AR1) = 0.9602 Half-life = −ln(2)/ln(0.9602) ≈ 17.1 ชม. ✅ revert ใน ~17h σ_stat = 0.021 (2.1%) ขั้น 4 — ตัดสิน: ε mean-revert ✅ | stationary ✅ | Half-life 17h → trade เป็น overnight swing ✅ หมายเหตุ: β_log = 1.183 ≠ β_raw = 0.0444 → β_raw เป็น slope บน raw price (ETH/$1 BTC) — ขึ้นกับ scale → β_log เป็น elasticity (ไม่ขึ้นกับ scale) — ใช้หนังสือเล่มนี้

ข้อจำกัดของ OLS — เรียงตามความรุนแรง

⚠️ ข้อจำกัดที่ 1 — Look-ahead Bias (อันตรายที่สุด — ทำ backtest ดีหลอกตา)
ผิด: β = OLS(all_data) → ใช้ backtest ทั้ง dataset → β "รู้" อนาคต → ผล backtest ดีเกินจริงเสมอ ถูก: สำหรับแต่ละวัน t: β_t = OLS(data[t-30 : t]) ← ใช้แค่อดีต 30 วัน เทรดวันที่ t+1 ด้วย β_t ← ไม่รู้อนาคต

ดู walk-forward validation: บทที่ 3 §3.9

ข้อจำกัดที่ 2 — Spurious Regression เมื่อทั้งสองตัว non-stationary

log P_A และ log P_B เป็น I(1) process (random walk) — ถ้า regress สอง I(1) series ที่ *ไม่มี* cointegration จะได้ R^2 สูง, β มีนัยสำคัญ แต่ ε ยังคง non-stationary — ผลที่ได้ไม่มีความหมาย

กับดัก: OLS บอกว่า $R^2 = 0.90$, $p < 0.001$ ดูน่าเชื่อถือมาก แต่: ADF ของ ε ได้ $p = 0.45$ ← non-stationary! → regression นี้ spurious — เทรดไม่ได้

แก้: ทดสอบ ADF/KPSS บน ε เสมอ — ผ่าน ADF ก่อนจึงใช้ β นั้น (ดู บทที่ 5)

ข้อจำกัดที่ 3 — β คงที่ ไม่ adaptive

OLS สมมติว่า β ไม่เปลี่ยนตลอด lookback window แต่ในตลาดจริง β อาจ drift ตาม regime

สัญญาณ: rolling β (30d vs 60d) ต่างกัน > 15% → ใช้ Rolling OLS หรือ Kalman แทน (§4.3, §4.6.3) ดูรายละเอียด window และ trap เพิ่มเติม: §4.6.6

ข้อจำกัดที่ 4 — Outliers ดึง β ให้ผิดทิศ

OLS minimize ε^2 → จุดที่อยู่ห่างจากเส้นมาก (outlier: crash, flash crash, pump) มี weight สูงเป็นพิเศษ → β ถูกดึงให้เข้าหา outlier

แก้: Winsorize ข้อมูลก่อน fit (clip return ที่ percentile 1–99) หรือใช้ Robust Regression (Huber/LAD) — แนวคิดเดียวกับ Robust Z-score ใน บทที่ 10 §10.3

ข้อจำกัดที่ 5 — Level vs Return ให้ β ต่างกันมาก
Level regression: log P_A = α + β_level · log P_B + ε Return regression: r_A = α + β_return · r_B + ε (r = log return) β_level ≠ β_return ในกรณีทั่วไป! ตัวอย่าง BTC/ETH: β_level ≈ 1.18 vs β_return ≈ 0.85–0.95 (แล้วแต่ period)

กฎ: สำหรับ ε = log P_A − β·log P_B → ใช้ **level regression บน log price** ดูเหตุผลเพิ่มเติม: §4.6.1

ข้อจำกัดที่ 6 — Gauss-Markov Assumptions มักถูกละเมิดในราคาจริง

Assumption	ใน OLS standard	ในราคาจริง	ผลกระทบ + วิธีตรวจ

No autocorrelation	ε_t ไม่ correlate กับ ε_{t-1}	ε มักมี autocorrelation (ราคาติดต่อกัน)	SE ของ β̂ ผิด → ใช้ **Durbin-Watson test** (DW ≈ 2 = ดี; DW < 1.5 = autocorrelation สูง → ใช้ Newey-West SE)
Homoskedastic	Var(ε) คงที่ทุก t	Var(ε) เปลี่ยนตาม volatility (GARCH)	OLS inefficient ใน high-vol period → ใช้ WLS หรือ GARCH-adjusted
No endogeneity	x และ ε ไม่ correlate	P_A และ P_B ต่างก็มี measurement noise	β̂ biased → TLS ดีกว่าใน symmetric-noise case (§4.2)

```
from statsmodels.stats.stattools import durbin_watson dw = durbin_watson(model.resid)
print(f"Durbin-Watson: {dw:.3f}") # ต้องการใกล้ 2.0 (1.5–2.5 ยอมรับได้)
```

สรุป: ใช้ OLS เป็น baseline ไม่ใช่ oracle — ตรวจ ε เสมอ อย่าเชื่อแค่ R^2 และ p-value
สรุป OLS ใน 4 ประโยค

1. OLS หา β ที่ทำให้ residual ε มี variance ต่ำสุด (minimize $\Sigma\varepsilon^2$)
2. β = Cov(log P_A, log P_B) / Var(log P_B) — ตีความว่า B อธิบาย A ได้กี่ %
3. ε ที่ได้คือ process ใหม่ที่เทรด — ต้องตรวจ ADF + half-life ก่อนใช้เสมอ
4. OLS ให้ β ที่ดีในหลายกรณี แต่ β คงที่ และ sensitive ต่อ window — ถ้า β drift ต้องไต่ขั้นขึ้นไป (Rolling OLS → Kalman)

โจทย์ 4.0 — OLS ทีละขั้น + ตรวจ ε

ได้ข้อมูล log price รายวัน 4 วัน:

วัน	log P_B (x)	log P_A (y)
1	9.00	4.50
2	9.10	4.56
3	9.20	4.64
4	8.90	4.44

(ก) คำนวณ β_OLS และ α
(ข) คำนวณ ε สำหรับทุก t และตรวจว่า Σε_t = 0
(ค) ถ้า ADF test บน ε ให้ p=0.31 ควรทำอย่างไรต่อ?

▶ ดูคำตอบ

4.3 Kalman Filter — Dynamic β ที่เปลี่ยนตามเวลา

OLS และ TLS สมมติว่า β คงที่ตลอด lookback window แต่ในตลาดจริง β อาจเปลี่ยนได้ Kalman Filter แก้ปัญหานี้โดยอัปเดต β ทุก time step:

Kalman State Space Model
β_t = β_{t-1} + w_t (state equation: β เดินแบบ random walk)

$r_{A,t} = \beta_t \cdot r_{B,t} + v_t$ (observation equation)

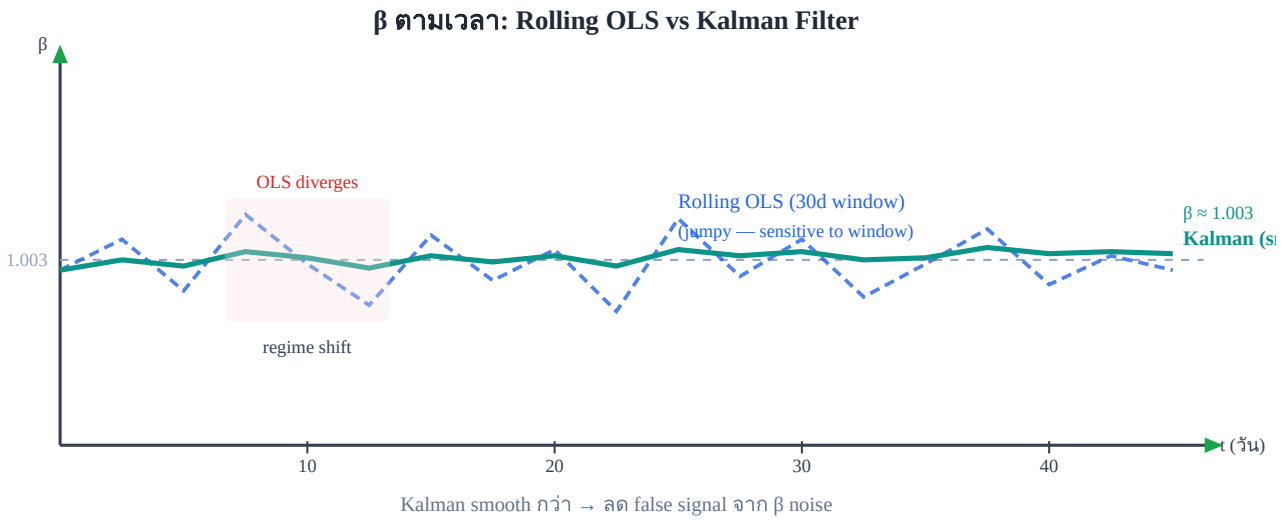
$w_t \sim N(0, Q)$ — process noise (ความเร็วที่ β เปลี่ยน)

$v_t \sim N(0, R)$ — observation noise (noise จากราคา)

Kalman Filter ทำงาน 2 ขั้นซ้ำกันทุก time step:

ขั้น	สูตร	ความหมาย
Predict	$\hat{\beta}_{t\|t-1} = \hat{\beta}_{t-1}$, $P_{t\|t-1} = P_{t-1} + Q$	คาดว่า β ตอนนี้คืออะไร (ก่อนเห็นข้อมูล)
Update	$K = P \cdot r_B / (r_B^2 \cdot P + R)$ $\hat{\beta}_t = \hat{\beta}_{t\|t-1} + K \cdot (r_A - \hat{\beta} \cdot r_B)$ $P_t = (1 - K \cdot r_B) \cdot P$	ปรับ β ตามข้อมูลจริงที่เพิ่งเห็น

K คือ **Kalman Gain** — ถ้า Q สูง (β เปลี่ยนเร็ว) K ใหญ่ → เชื่อข้อมูลใหม่มาก ถ้า R สูง (noisy market) K เล็ก → เชื่อ model เก่ามากกว่า



Rolling OLS กระโดดตาม window boundary — Kalman ค่อยๆ ปรับ → position ขยับน้อยกว่า

4.4 Lookback Window และการ Roll

สำหรับ OLS rolling หรือ Kalman Q/R tuning ต้องเลือก lookback ให้เหมาะกับ strategy:

Lookback	ข้อดี	ข้อเสีย	เหมาะกับ
7 วัน	Responsive ต่อ regime ใหม่	Noisy, β กระโดด	Fast intraday
30 วัน	สมดุล stability/responsiveness	Lag ช้า	**Intraday-swing (แนะนำ)**
90 วัน	Stable, noise น้อย	ปรับ regime เปลี่ยนช้ามาก	Position trade

Re-estimate β เมื่อไร?

- Daily: re-run OLS บน 30 วันล่าสุด ก่อนตลาดเปิด
- เมื่อ ε drift ต่อเนื่อง > 2 วัน → β เปลี่ยน re-estimate ทันที

- Kalman: ไม่ต้อง re-estimate — อัปเดต continuous ทุก bar

4.5 เลือก β Method ตามวัตถุประสงค์ (Objective-Driven Design)

แก่นความคิด

OLS/TLS/Kalman ไม่ใช่แค่ตัวเลือกที่ดีกว่ากัน — แต่ละตัวถูกออกแบบมาเพื่อวัตถุประสงค์ต่างกัน เลือกให้ตรงกับสิ่งที่ต้องการ

สามเส้นทาง: Objective-to-ε Mapping

วัตถุประสงค์	Model	ε target (variance)	แหล่งกำไรหลัก
Funding Rate Harvest Perp↔Perp / Spot↔Perp	Kalman Filter (dynamic β_t)	**$\sigma^2 \to 0$** — กราฟแบนขนานศูนย์	Funding rate yield — เพิ่ม leverage ปลอดภัยได้เพราะ price risk ≈ 0
Hybrid: Spread + Funding Spot↔Future / Pairs Trading	OLS/TLS (static β) + walk-forward	**σ_stat "Optimal Substantial"** — กว้างพอกว่า friction แต่ half-life สั้น	Capital gain จาก mean reversion *และ* basis/funding
Volatility Arbitrage Spot↔Option (ขั้นสูง)	GARCH(1,1) + Delta-Neutral hedge	**Dynamic — ขึ้นกับ IV regime** — กรอง vol clustering ด้วย GARCH	IV mispricing บน Options surface — นอกขอบเขตหนังสือนี้

อธิบายเชิงแนวคิด

Funding Rate Harvest — ต้องการ ε_t → 0

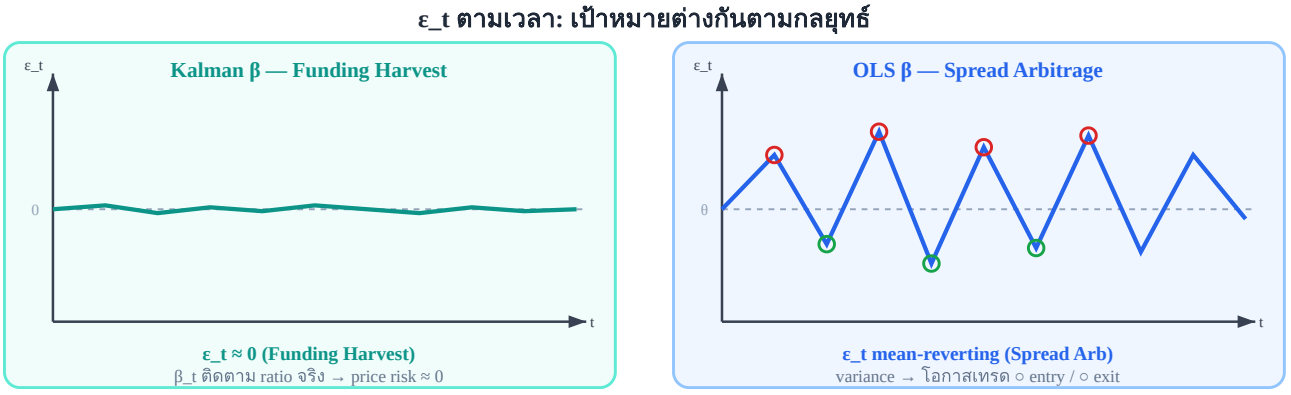
เป้าหมายคือ market-neutral perfect hedge: Kalman β_t ปรับทุก bar ตาม cost-of-carry จริง ทำให้ ε_t ≈ 0 ตลอดเวลา PnL ทั้งหมดมาจาก **funding rate เท่านั้น** ไม่ใช่ price movement ยิ่ง ε เล็กยิ่งดี — แสดงว่า hedge สมบูรณ์

Spread Arbitrage — ต้องการ ε_t แกว่งรอบ θ

OLS lock β จากอดีต — ความไม่สมบูรณ์ของ hedge นี้เองคือแหล่งกำไร ε_t ที่มี variance สูงหมายถึง**โอกาสเทรดมาก** เมื่อ ε เบี่ยงจาก mean → entry; เมื่อ revert กลับ → exit

ความเข้าใจผิดที่พบบ่อย

Kalman ไม่ได้ "ดีกว่า" OLS เสมอ — ถ้าใช้ Kalman กับ Spread Arb จะได้ ε → 0 ซึ่งทำให้ไม่มีสัญญาณ เพราะ β_t ปรับตัวจนกำจัด spread ออกไปหมด ต้องเลือก method ให้ตรงกับ **สิ่งที่ต้องการให้ ε ทำ**



ซ้าย: Kalman ε_t แบน — Funding Harvest ($\sigma^2 \to 0$) | ขวา: OLS ε_t แกว่ง — Hybrid/Spread Arb (σ_stat "Optimal")

Breakeven σ_stat — เส้นแบ่ง "Optimal Substantial Variance"

สำหรับ Hybrid Strategy สิ่งที่กำหนด "กว้างพอ" ของ σ_stat คือสมการ Breakeven จาก [บทที่ 19.7](#):

Breakeven σ Condition
σ_stat ≥ total_friction_cost / (z_entry − z_exit)

ถ้า σ_stat ต่ำกว่า threshold นี้ → gross spread ที่ entry ไม่ครอบคลุม friction → net_edge < 0 → ห้ามเทรด

Optimal Substantial Variance — นิยามเชิงปฏิบัติ

σ_stat ที่ "พอเหมาะ" ต้องตอบ **สองเงื่อนไขพร้อมกัน**:

- **Lower bound:** σ_stat ≥ total_cost / (z_entry − z_exit) → กว้างพอกว่า friction (breakeven condition)
- **Upper bound:** Half-life ≤ N bars → spread ไม่ค้างนานเกินไปจน margin cost กินกำไร

ช่วงระหว่าง lower–upper bound นี้คือ "Tradeable Window" ของกลยุทธ์ Hybrid

ตัวอย่าง Bybit↔Lighter (taker-taker): total_friction = 19.6 bps (ดู ch19) z_entry = 2.0, z_exit = 0.5 Breakeven σ_stat = 19.6 / (2.0 − 0.5) = 13.1 bps (0.131%) วัดได้จริง: σ_stat = 30 bps (0.30%) ส่วนเกิน = 30 − 13.1 = 16.9 bps ← gross edge สุทธิต่อรอบ ✅ σ_stat > Breakeven σ → trade ได้ ❌ ถ้า σ_stat = 10 bps → 10 < 13.1 → ห้ามเทรด
Running Example: Bybit↔Lighter — สองเส้นทาง
กลยุทธ์ที่ 1 — Funding Rate Harvest (Kalman): ใช้ Kalman β_t → ε_t แบนราบ ≈ 0 Bybit funding rate 0.01%/8h = 10 bps/day คือ PnL ทั้งหมด price risk ≈ hedged ออกไปเกือบสมบูรณ์ กลยุทธ์ที่ 2 — Spread Arbitrage (OLS): ใช้ OLS β = 1.002 (ค่าคงที่จาก 30d window) ε_t แกว่ง ±0.3%, Half-life 7.4h คือโอกาสเทรด entry เมื่อ z-score > 2σ, exit เมื่อ revert หนังสือเล่มนี้: → Spread Arb path (OLS/TLS) = running example หลัก (ch5–ch14) → Kalman สำหรับ Funding Harvest = ch15

4.6 การหา β ในทางปฏิบัติ — เลือกข้อมูล ประเมินคุณภาพ และข้อผิดพลาด

ก่อนลงรายละเอียดว่าใช้ method ไหน ต้องเข้าใจ **ตำแหน่งของ β** ในภาพใหญ่ก่อน — สิ่งที่เราต้องการสร้างจริงๆ ไม่ใช่ β แต่เป็น **ε (synthetic process)** ส่วน β คือแกนสำคัญที่สุดที่ทำให้สร้าง ε ได้ *ถูกสัดส่วน*

แก่นความคิด — β คือสูตรผสม, ε คือของใหม่ที่เกิดจากการผสม
$\varepsilon = F(A) - \beta \cdot F(B)$

β คือ **อัตราส่วนผสม (mixing ratio / hedge ratio)** ที่ตอบว่า "ถ้ามี A 1 หน่วย ต้องใช้ B กี่หน่วยมาหักล้าง?" — ประโยคที่แม่นที่สุดคือ: **β คืออัตราส่วนที่ใช้ผสมสองสิ่ง เพื่อให้ได้ ε ที่มีคุณสมบัติเทรดได้**

β	สมการ ε	ความหมายของการผสม
β = 1	ε = A − B	ผสม 1:1 — ใช้ B เท่ากับ A
β = 0.8	ε = A − 0.8B	ใช้ B แค่ 0.8 เท่าของ A
β = 1.3	ε = A − 1.3B	ใช้ B มากกว่า A เพื่อหักล้าง exposure ให้พอดี

ตัวอย่างจำง่าย — ทำไมไม่ดูแค่ A − B

ให้ A = BTC basis บน Lighter, B = BTC basis บน Bybit — เราไม่ดูแค่ผลต่างดิบ $A - B$ แต่ดู $\boldsymbol{\varepsilon = A - \beta B}$ เพราะ β ตอบว่า "Bybit basis ควรถูกถ่วงน้ำหนักเท่าไร จึงหักล้าง Lighter basis ได้พอดี" ถ้าถ่วงผิดสัดส่วน ส่วนที่เหลือจะไม่ใช่ inefficiency แท้ แต่เป็น market move ที่หักล้างไม่หมด → signal หลอก

β กำหนดว่า process ใหม่จะ "หน้าตา" เป็นอย่างไร — มันไม่ใช่แค่ตัวเลขประกอบสูตร แต่เป็น**ตัวกำหนดโครงสร้างของ synthetic process**:

β ดี → ε ที่เทรดได้

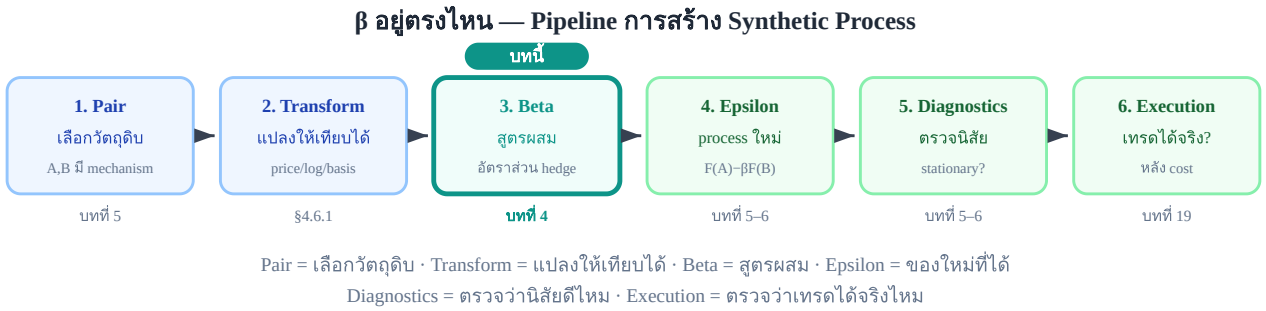
common movement ถูกหักล้าง · ε นิ่งขึ้น · mean reversion ชัดขึ้น · เป็น risk-neutral จริง

β ผิด → process หลอก

ε drift · signal หลอก · กลายเป็น directional bet · position size ผิด · risk สูงกว่าที่คิด

β อยู่ตรงไหนใน Pipeline

β สำคัญมาก แต่ **ไม่ใช่ทุกอย่าง** — มันเป็นขั้นที่ 3 ของกระบวนการสร้าง process ต้องมี pair selection + transformation มาก่อน และมี diagnostics + execution gate ตามหลัง:



β เป็นขั้นที่ 3 — ขาดขั้นไหนก็สร้าง ε ที่เทรดได้จริงไม่ได้ บทที่ 4 โฟกัสที่ขั้น Beta
β เป็นแนวคิดทั่วไป — ใน option เรียกว่า Delta

หลักการ "หาอัตราส่วนที่ถูกต้องเพื่อสร้าง process ใหม่" ใช้ได้กับทุก asset class — ใน option process: ε = Option − Δ·Underlying โดย **Δ (delta) ทำหน้าที่เป็น hedge ratio แทน β** แนวคิดเดียวกัน: ไม่จำเป็นต้องผสม 1:1

ที่เหลือของ §4.6 เจาะเรื่องที่ทำให้ β *ใช้ได้จริง* ไม่ใช่แค่ตัวเลขสวยบน backtest: เลือกข้อมูลที่ถูก (§4.6.1) · เข้าใจว่า β หักล้างอะไร (§4.6.2) · เลือก method ตามสถานการณ์ (§4.6.3–4.6.4) · ประเมินคุณภาพและเลี่ยงกับดัก (§4.6.5–4.6.6)

4.6.1 regress ข้อมูลแบบไหน — Raw Price, Log Price หรือ Basis

แก่นความคิด

β ไม่มีค่าเดียวตายตัว — ค่ามันขึ้นกับว่าคุณ regress **ราคาดิบ, log price, หรือ basis** สามแบบนี้ให้ β คนละตัวเลข และตอบคนละคำถาม

Input ที่ regress	สมการ ε	ใช้เมื่อ	β หมายถึงอะไร
Raw price	ε = P_A − β·P_B	scale ราคาใกล้กัน / ต้องการดูภาพ scatter	$ ของ A ต่อ $1 ของ B — **ขึ้นกับ scale ราคา**
Log price	**ε = log P_A − β·log P_B**	**default — pairs ทั่วไป (ที่หนังสือเล่มนี้ใช้)**	elasticity: %A ต่อ %B — **scale-invariant**
Basis / spread	ε = basis_A − β·basis_B	funding arb, perp↔perp, swap arb (บทที่ 21)	sensitivity ของ basis ขาหนึ่งต่ออีกขา

BTC/ETH — ทำไม β = 0.0444 จึงหลอกตา

ภาพวิเคราะห์ BTC/ETH ที่นิยมแชร์กัน fit regression บน **ราคาดิบ**: ETH = α + β·BTC ได้ **β = 0.0444** ตัวเลขนี้แปลว่า "ETH ขยับ $0.0444 ต่อทุก $1 ที่ BTC ขยับ" — มันถูกในเชิงเลขคณิต แต่ **ขึ้นกับระดับราคา**: ถ้า BTC แตก scale (เช่น redenominate ÷10) β จะกลายเป็น 0.444 ทันที ทั้งที่ความสัมพันธ์จริงไม่เปลี่ยน

ถ้า regress **log price** แทน: log(ETH) = α + β·log(BTC) จะได้ β เป็น **elasticity** (สำหรับ crypto major มักใกล้ 1) — ค่านี้ไม่เปลี่ยนตาม scale ราคา จึงเอาไป calibrate ข้าม regime ได้ปลอดภัยกว่า
หนังสือเล่มนี้ใช้สมการ ε = log P_A − β·log P_B เป็นมาตรฐานด้วยเหตุผลนี้
อย่าเอา raw-price β ไปใส่ position โดยตรง

β = 0.0444 เป็นแค่ slope บนกราฟ scatter — ต้องแปลงเป็น **กฎกำหนดขนาด position** ที่ทำให้ exposure ต่อ common factor หักล้างกัน (ดู §4.6.2) ข้อควรระวัง: hedge ที่ถูกต้อง *ไม่จำเป็น* ต้องเป็น dollar-neutral เป๊ะ — มันคือ **factor-neutral**

4.6.2 β หักล้าง exposure — ไม่ใช่ทำมูลค่าให้เท่ากัน

จุดที่หลายคนสับสน: $A - \beta B$ **ไม่ได้**แปลว่าต้องทำให้ βB มีค่า "เท่ากับ" A ตลอดเวลา — βB คือ **ส่วนของ A ที่ B อธิบายได้** ส่วนที่เหลือคือ ε

การแยกส่วน — A = ส่วนที่เคลื่อนตาม B + ส่วนผิดปกติ
A = βB + ε

βB = "เส้น fair relationship" (ส่วนที่เคลื่อนตาม B) · ε = ส่วนผิดปกติที่เหลือหลังใช้ B อธิบาย A — **βB ไม่ใช่ราคาที่ต้องเท่ากับ A เสมอ**

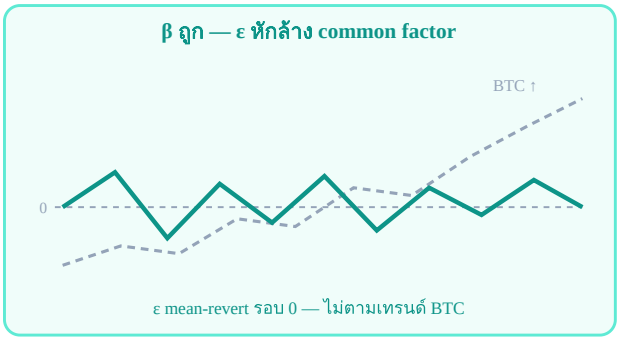
สถานะ	ε	ความหมาย
A = βB	ε = 0	process อยู่ที่ equilibrium / mean
A > βB	ε > 0	A แพงกว่าที่ควรเป็นเมื่อเทียบกับ B
A < βB	ε < 0	A ถูกกว่าที่ควรเป็นเมื่อเทียบกับ B

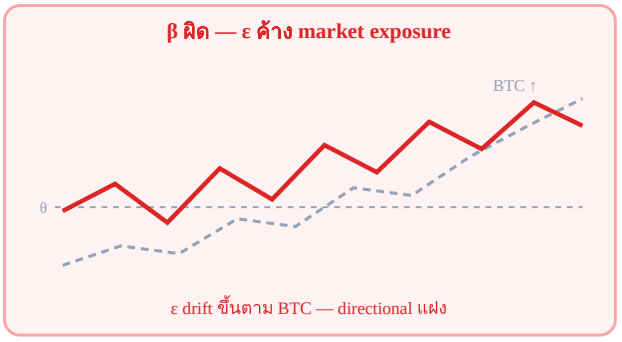
แล้ว β ผิด ทำให้กลายเป็น directional ได้อย่างไร? สมมติ A และ B ถูกขับเคลื่อนด้วยแรงตลาดเดียวกัน (เช่น BTC market factor):

A = market_factor + noise_A B = market_factor + noise_B β ถูก: ε = A − βB ≈ noise ← market factor ถูกหักล้าง β ผิด: ε = A − βB = noise + market exposure ที่ค้างอยู่ ← กลายเป็น directional โดยไม่รู้ตัว
ตัวอย่าง: A เคลื่อนไหวแรงกว่า B ราว 2 เท่า (A ≈ 2B)

ใช้ ε = A − 1·B → ยังเหลือ exposure ต่อ B เยอะ เพราะ A ขยับมากกว่าที่ 1B หักล้างได้
ใช้ ε = A − 2·B → ส่วนที่เคลื่อนไหวร่วมกันถูกหักล้างพอดี
→ β ทำให้ **ความไวต่อ common factor เท่ากัน** ไม่ใช่ทำให้ *มูลค่า* เท่ากัน

β ถูก vs β ผิด — residual หน้าตาต่างกัน





β ที่ถูกต้องทำให้ ε เป็น residual ที่ mean-revert — β ผิดทำให้ ε ยังพก market exposure ค้างอยู่

แยก 2 คำให้ขาด — ในการเทรดจริงต้องไม่สับสนระหว่าง:

	Value / Notional matching	Risk / Factor matching
นิยาม	มูลค่าเงินสองขาเท่ากัน	exposure ต่อ factor สองขาเท่ากัน
ตัวอย่าง	Long A $10,000 / Short B $10,000	Long A $10,000 / Short B $8,000 (ถ้า B ขยับแรงกว่า)
ให้อะไร	dollar-neutral	hedge ที่ถูกต้องจริง (factor-neutral)

กับดักที่ต้องระวัง

notional เท่ากัน **ไม่ได้แปลว่า** hedge ถูก · notional ไม่เท่ากัน **ไม่ได้แปลว่า** directional — สิ่งที่ตัดสินคือ exposure ต่อ common factor ถูกหักล้างหมดหรือยัง

Bybit ↔ Lighter — β = 0.8 หรือ 1.2 ไม่ได้แปลว่าตั้งใจ directional
A = basis_Lighter, B = basis_Bybit (BTC perp underlying เดียวกัน) เริ่มจาก β = 1 (underlying เดียวกัน → baseline) ถ้าข้อมูลบอกว่า basis ของ Lighter เคลื่อนแรง/ช้ากว่า Bybit: β = 0.8 หรือ β = 1.2 → ไม่ใช่การตั้งใจเปิด directional → เป็นการปรับ hedge ratio ให้เข้ากับพฤติกรรมจริงของ 2 ตลาด
ทำไมต้องทดสอบ — สมการแสดงที่มาของ exposure ที่เหลือ

เขียน spread change เป็นสมการ:

$\Delta\varepsilon = (\text{exposure_A} - \beta \times \text{exposure_B}) \times \text{market_return} + \text{noise}$

ถ้า β ถูกต้อง → `exposure_A − β × exposure_B ≈ 0` → ε ไม่ขึ้นกับตลาด
ถ้า β ผิด → เทอมแรกไม่ใช่ศูนย์ → ε correlate กับ market_return → **directional bet แฝง**

β ที่เหมาะสมจึงต้องอยู่ที่:

$\beta \approx \text{exposure_A} / \text{exposure_B}$

เช่น BTC Lighter vs BTC Bybit — ถ้า Lighter sensitive ต่อ BTC มากกว่า Bybit 1.2× → β ≈ 1.2 ไม่ใช่ 1.0

6 วิธีตรวจว่า ε เป็น residual จริง ไม่ใช่ directional แฝง

Test 1 — Correlation กับ market return

corr(Δε, market_return)

- |r| < 0.20 → ดี (ε แทบไม่ขึ้นกับทิศตลาด)
- |r| > 0.50 → β ยัง hedge ไม่ดี → ต้องปรับ β หรือเปลี่ยน pair

Test 2 — Factor regression

Δε = α + γ · market_return + error

- γ ≈ 0 (p-value > 0.05) → hedged ดี — market exposure ถูกหักล้างแล้ว
- |γ| ใหญ่ (t-stat > 2) → exposure ยังหลงเหลือ → ε ไม่บริสุทธิ์

Test 3 — PnL แยกตามทิศทางตลาด

- แบ่ง: ตลาดขึ้น > 1% / ลง > 1% / Sideways (±1%)
- stat arb จริง → PnL กระจายสม่ำเสมอ ไม่ขึ้นกับทิศทาง
- PnL ดีเฉพาะ "ตลาดขึ้น" → กำไรมาจาก beta ไม่ใช่ spread reversion

Test 4 — Stationarity (จำเป็นแต่ไม่เพียงพอ)

- ADF / KPSS ต้องผ่าน — แต่ stationarity ไม่ได้พิสูจน์ว่า β ถูก
- ε stationary แต่ corr กับตลาดสูง → β ผิด directional ยังแฝงอยู่
- ε non-stationary และ corr สูง → pair ไม่ดีทั้งคู่

Test 5 — Half-life และ Crossing Rate

- $\tau_{1/2}$ ต้องอยู่ใน target horizon (บทที่ 3)
- φ ควรอยู่ใน sweet spot 0.85–0.97 สำหรับ swing (บทที่ 5 §5.7)
- ถ้า Test 1–3 ผ่านแต่ half-life ยาวเกิน → pair ดี แต่ยังไม่เหมาะ intraday

Test 6 — Rolling β Stability

- คำนวณ β_30d, β_60d, β_90d — ควรใกล้กัน (±15%)
- β drift เร็ว → regime เปลี่ยน → β เก่าอาจใช้ไม่ได้แล้ว
- ถ้า β drift → เปลี่ยนไปใช้ Rolling OLS หรือ Kalman (§4.6.3)

10 สัญญาณเตือน — spread นี้อาจ directional ปลอมตัว

1. |corr(Δε, market)| > 0.50 — ε ยัง sensitive ต่อทิศทางตลาดอย่างมีนัย
2. γ ≠ 0 อย่างมีนัย (t-stat > 2) ใน factor regression — market exposure ค้างอยู่
3. Sharpe ดีเฉพาะช่วงตลาดขึ้น (bull) — กำไรมาจาก beta ไม่ใช่ mean reversion

4. Drawdown ใหญ่ในระดับเดียวกับ long position เดี่ยวขนาดเดียวกัน — hedge ไม่จริง
5. ε ไม่ผ่าน ADF — spread ไม่กลับ mean เลย
6. Half-life > target holding period — ไม่ทันเก็บกำไร
7. β drift > 20% ภายใน 30 วัน — regime เปลี่ยน β เก่าผิดแล้ว
8. ε correlate กับ implied vol / VIX — รับ volatility premium ปลอมตัวเป็น arb
9. PnL จาก sideways market < 30% ของกำไรรวม — ตลาดต้อง trend ถึงทำกำไร
10. β จาก regression ห่างจาก notional ratio > 30% โดยไม่มีเหตุผลเชิง factor

Checklist ปฏิบัติ — 8 ขั้นตอนก่อน go live

1. คำนวณ corr(Δε, market_return) → ต้องการ |r| < 0.20
2. Run factor regression → ต้องการ |γ| ไม่ significant (p > 0.05)
3. แบ่ง PnL ตามทิศทางตลาด → กระจายสม่ำเสมอ ไม่ขึ้นกับทิศ
4. ตรวจ ADF/KPSS → ε ต้อง stationary (บทที่ 5)
5. คำนวณ half-life และ φ → อยู่ใน target horizon และ sweet spot
6. Plot rolling β (30d/60d/90d) → stability ±15%
7. Walk-forward test → ε properties ยังดีใน out-of-sample (บทที่ 3 §3.9)
8. ทบทวนหลัง macro event (Fed meeting / major shock) → β อาจต้อง recalibrate

Residual ที่ดีควรเหลือ "ความผิดปกติระหว่างสองขา" — ไม่ใช่ "ทิศทางตลาด" ที่ hedge ไม่หมด

ประโยคจำ

β ไม่ได้มีหน้าที่ทำให้ B "ราคาเท่ากับ" A — β มีหน้าที่ทำให้ B "หักล้าง exposure ของ A" ได้เหมาะสม
ถ้า β ดี สิ่งที่เหลือคือ residual · ถ้า β แย่ สิ่งที่เหลือคือ directional bet ที่ปลอมตัวเป็น stat arb

4.6.3 เมนูเต็มของวิธีหา β

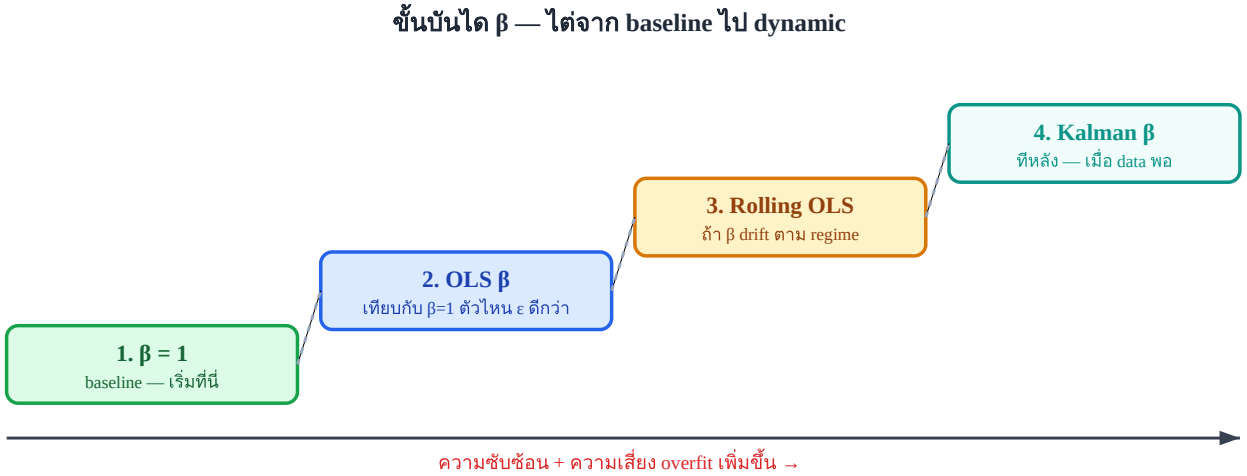
§4.2–4.3 อธิบาย OLS / TLS / Kalman เชิงลึก — ตารางนี้รวมทุกวิธีไว้เป็น "เมนู" เดียวเพื่อเลือกใช้ตามสถานการณ์:

วิธี	ใช้เมื่อ	ข้อดี	ข้อเสีย
β = 1	สินทรัพย์เดียวกัน, exposure เท่ากัน (BTC perp ↔ BTC perp)	ง่ายสุด, ไม่ overfit	หยาบ — venue ต่างกันเรื่อง funding/index/liquidity
Contract ratio	crack spread / product ที่มี ratio เชิงกลไกชัดเจน	mechanism ชัด ไม่ต้อง fit	ใช้ได้เฉพาะบางตลาด
OLS	pair 2 ตัวทั่วไป — **baseline เริ่มต้น**	ง่าย, เร็ว, ตีความได้	β คงที่ — อาจ outdate เมื่อ regime เปลี่ยน
Rolling OLS	ความสัมพันธ์เปลี่ยนตาม regime	adaptive กว่า OLS static	noisy ถ้า window สั้นเกิน

TLS	ทั้ง X และ Y มี noise พอๆ กัน	สมดุลกว่า OLS (ดู §4.2)	ซับซ้อนขึ้น, ตีความยากกว่า
Kalman	β เปลี่ยนตลอดเวลา / Funding Harvest (§4.5)	dynamic, online update	overfit ง่าย, tune Q/R ยาก
Johansen	multi-leg / basket (≥ 3 ขา)	หา cointegrating vector หลายขาพร้อมกัน	overfit ง่าย, ต้องระวังเป็นพิเศษ
Greek (Delta/Vega)	options / volatility strategy	ตรงกับโครงสร้าง option	ต้อง update ตลอดเพราะ Greek เปลี่ยน

4.6.4 Roadmap — เริ่มจากง่าย ค่อยไปยาก

ข้อผิดพลาดที่พบบ่อยคือเริ่มจาก Kalman ทันทีเพราะ "ดูฉลาด" — แต่ Kalman มีพารามิเตอร์ให้ tune เยอะ จึง overfit ง่ายมากเมื่อข้อมูลยังน้อย ลำดับที่แนะนำสำหรับ scanner v1:



อย่ากระโดดไป Kalman ทันที — พิสูจน์ก่อนว่า β=1 หรือ OLS ยังไม่พอ จึงค่อยไต่ขั้นถัดไป

4.6.5 β ดีหรือไม่ดี ดูยังไง — Checklist 7 ข้อ

β ที่ดี **ไม่ใช่** β ที่ทำให้กราฟ backtest สวยที่สุด — แต่คือ β ที่ทำให้ process เทรดได้จริงและ robust ใช้ checklist นี้:

เกณฑ์ประเมิน β

1. **ε stationary มากขึ้น** — ADF p-value ต่ำลง / KPSS ดีขึ้น (บทที่ 5)
2. **Half-life เหมาะกับ horizon** — ไม่สั้นจนจับไม่ทัน ไม่ยาวจน margin cost กิน (บทที่ 3)
3. **Crossing behavior ดี** — φ อยู่ใน sweet spot (บทที่ 5 §5.7)
4. **Residual ไม่ drift** — ε ไม่ค่อยๆ ไหลไปทางเดียว (ไม่มี trend แฝง)
5. **Net edge หลัง cost ยังเหลือ** — σ_stat ≥ breakeven σ (§4.5)

6. **Position ไม่กลายเป็น directional bet** — factor-neutral จริง: ε ไม่ correlate กับ underlying/market (ดู §4.6.2)
7. **β เสถียรพอใน out-of-sample** — ผ่าน walk-forward validation (บทที่ 3 §3.9)

4.6.6 ข้อผิดพลาดที่พบบ่อย

7 กับดักเรื่องการหา β

1. **ใช้ β = 1 กับทุกอย่าง** — สะดวกแต่หยาบ; venue ต่างกันมักทำให้ ε มี bias
2. **ใช้ราคาดิบแทน log price / basis** — β กลายเป็นตัวเลขที่ขึ้นกับ scale (ดู §4.6.1)
3. **หา β จากข้อมูลทั้งหมดรวมอนาคต** — look-ahead bias; ต้อง fit บน train window เท่านั้น
4. **ใช้ window ยาวเกินไป** — β ตามไม่ทัน regime ใหม่
5. **Rolling window สั้นเกินไป** — β noisy กระโดด → position ขยับถี่จน friction กิน
6. **เลือก β ที่ backtest สวยที่สุด** — overfit; ต้องผ่าน OOS / paper validation ก่อน
7. **ลืมแปลง β เป็น position size จริง** — β บน regression ≠ notional ratio ที่ trade

import statsmodels.api as sm import numpy as np # regress บน LOG price (มาตรฐานของหนังสือ) — ใช้ข้อมูล train window เท่านั้น y = np.log(train["price_A"]) x = np.log(train["price_B"]) model = sm.OLS(y, sm.add_constant(x)).fit() alpha = model.params["const"] beta = model.params.iloc[1] # hedge ratio epsilon = y - (alpha + beta * x) # diagnostics ใช้ residual รวม alpha # ตอนแปลงเป็น position จริง: alpha เป็นค่าคงที่ของ model ไม่ใช่ขา trade # long ε → long $X ของ A, short $X ของ B (dollar-neutral)
Running Example: BTC/ETH — จากภาพวิเคราะห์สู่ β ที่ใช้ได้จริง
ภาพวิเคราะห์ (regress ราคาดิบ ETH บน BTC): β_raw = 0.0444 ← slope บน scatter — ขึ้นกับ scale ราคา Corr = 0.9745 ✅ ผ่าน correlation screen (≥ 0.70) AR(1) φ = 0.9602 Half-Life = 17.08 ชม. (−ln2 / ln 0.9602 ≈ 17.06 ✓) แปลงให้ถูกหลักของหนังสือ — regress log price: ε = log(ETH) − β_log · log(BTC) β_log ≈ elasticity (crypto major มักใกล้ 1) — scale-invariant ผ่าน Checklist §4.6.5: 1. stationary — Half-life 17h → ε mean-reverting ✅ 3. crossing φ=0.96 — นอก intraday sweet spot → swing เท่านั้น (ch5 §5.7) 7. walk-forward — ต้อง re-check β ทุก window (ch3 §3.9) สรุป: ใช้ β_log + dollar-neutral notional → trade เป็น overnight swing อย่าใช้ β_raw = 0.0444 ใส่ position ตรงๆ
โจทย์ 4.6 — เลือกชนิดข้อมูลที่จะ regress

คุณจะหา β ของสามคู่นี้:

- (ก) BTC perp บน Bybit ↔ BTC perp บน Lighter
- (ข) PAXG (gold token) ↔ XAUUSD spot
- (ค) ETH spot ↔ ETH perpetual (เก็บ funding)

ถาม: แต่ละคู่ควร regress raw price, log price, หรือ basis?

▶ ดูคำตอบ

4.7 Pseudo-code

```
# OLS Beta
function beta_ols(r_A, r_B):
  return cov(r_A, r_B) / var(r_B)

# TLS Beta (via eigenvalue)
function beta_tls(r_A, r_B):
  C = cov_matrix([[r_A], [r_B]]) # 2×2
  eigenvalues, eigenvectors = eig(C)
  v = eigenvectors[:, argmin(eigenvalues)] # smallest eigenvalue
  return -v[0] / v[1]

# Kalman Filter Beta (one-step update)
function kalman_beta_update(beta, P, r_A, r_B, Q, R):
  # Predict
  P_pred = P + Q
  # Kalman gain
  K = P_pred * r_B / (r_B**2 * P_pred + R)
  # Update
  innovation = r_A - beta * r_B
  beta_new = beta + K * innovation
  P_new = (1 - K * r_B) * P_pred
  return beta_new, P_new
```

Running Example: β ของ Bybit↔Lighter BTC Perp

ข้อมูล: 30 วัน, log returns รายนาที OLS (30d window): β_OLS = Cov(r_Bybit, r_Lighter) / Var(r_Lighter) = 0.00000412 / 0.00000410 = 1.0049 TLS: β_TLS = 1.0063 (เล็กน้อยกว่า OLS) Kalman (Q=1e-7, R=1e-5, warmup 14 วัน): β_Kalman_t ≈ 1.003 ± 0.002 (เปลี่ยนช้าๆ) ผลกระทบต่อ position: Long Bybit $100,000: OLS: Short Lighter $100,490 TLS: Short Lighter $100,630 Kalman: Short Lighter $100,300 ต่างกัน $300–$630 — ดูเล็กน้อย แต่ถ้า trade 100x/เดือน = $30,000–$63,000 ต่างกัน

โจทย์ 4.5 — เลือก Method ให้ตรงวัตถุประสงค์

บริษัท Quant เปิดสองกลยุทธ์:

- (ก) ใช้ Kalman β_t และ target ε_t ≈ 0
- (ข) ใช้ OLS β และ z-score entry/exit

ถาม: กลยุทธ์ไหนทำกำไรจาก funding rate? กลยุทธ์ไหนทำกำไรจาก price inefficiency?

▶ ดูคำตอบ

โจทย์ 4.1 — คำนวณ β_OLS จากข้อมูล

Log returns รายวัน 5 วัน:

วัน	r_Bybit (%)	r_Lighter (%)
1	+1.20	+1.15

2	−0.80	−0.75
3	+2.10	+2.05
4	−1.50	−1.48
5	+0.60	+0.58

ถาม: คำนวณ β_OLS (Bybit เป็น A, Lighter เป็น B)

▶ ดูคำตอบ

โจทย์ 4.2 — เลือกวิธีหา β

คุณ trade spread ระหว่าง BTC perpetual บน Binance (volume 10× ของ Lighter) กับ Lighter ทั้งสองตลาดมี liquidity ต่างกันมาก — Binance เป็น "price discovery" หลัก

ถาม: ควรใช้ OLS หรือ TLS? และ Binance ควรเป็น A หรือ B ใน regression?

▶ ดูคำตอบ

โจทย์ 4.3 — Kalman Q/R tuning

คุณตั้ง Kalman กับ Q=1e-9 (เล็กมาก) และ R=1e-4 (ใหญ่มาก)
สังเกตว่า β_Kalman แทบไม่เปลี่ยนเลย แม้ตลาดจะเปลี่ยน

ถาม: อธิบายว่าเกิดอะไรขึ้น และควรปรับ Q หรือ R อย่างไร?

▶ ดูคำตอบ

Ch.4 Hedge Ratio β | ต่อไป → **Ch.5: Cointegration Tests**

Statistical Arbitrage · บทที่ 5

Cointegration

Engle-Granger · Johansen · VECM
ทดสอบว่า spread ε มีแรงดึงกลับจริงหรือแค่ correlation ชั่วคราว
แก่นของบทนี้

Correlation สูงไม่พอสำหรับ stat arb — ต้องการ **cointegration**:
ε = log(A) − β·log(B) ต้องเป็น stationary (กลับหาค่ากลาง) ไม่ใช่แค่เดินไปด้วยกัน

H_0: ε ไม่ stationary (ไม่ cointegrated) → ถ้า p-value < 0.05 → ปฏิเสธ H_0 → cointegrated ✓
นิยามทางการ — Cointegration (Engle & Granger, 1987)

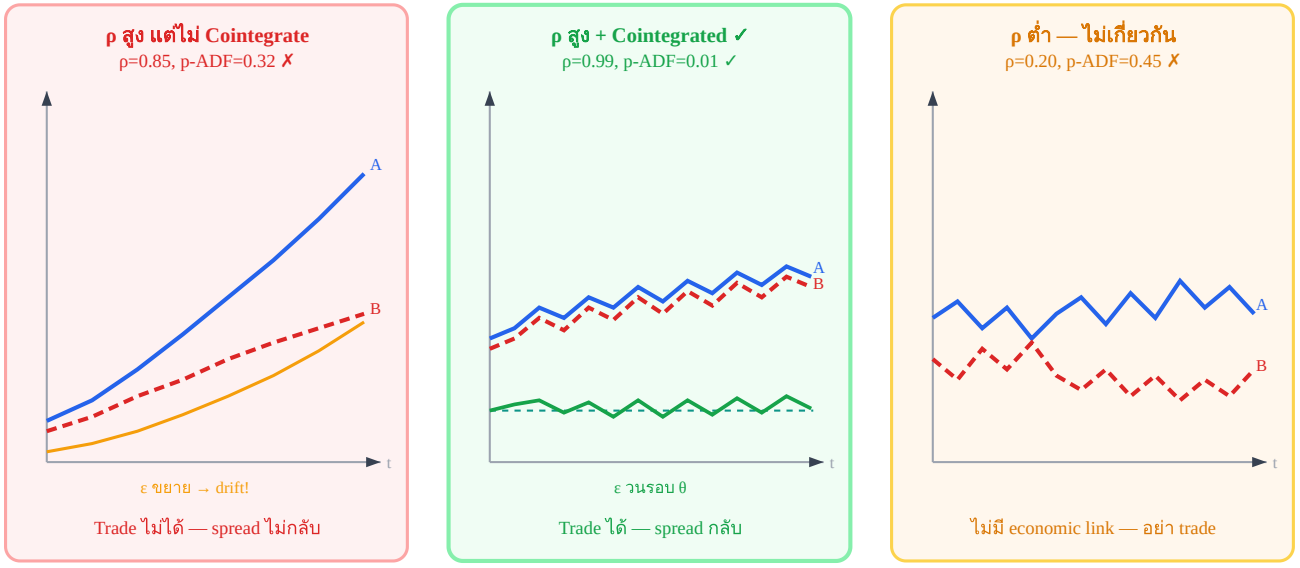
Series X_t และ Y_t แต่ละตัวเป็น **I(1)** (integrated of order 1 — มี unit root, ไม่ stationary) แต่คู่นี้ **cointegrated** ถ้ามีค่า β ที่ทำให้:

```
Z_t = X_t − β·Y_t เป็น I(0) (stationary, กลับหาค่ากลาง)
```

กล่าวอีกนัย: แม้ X_t และ Y_t จะ random walk แยกกัน แต่ทั้งคู่ถูก "ดึง" ไว้ด้วย long-run equilibrium relationship เดียวกัน — spread ε = Z_t จึงไม่ drift ออกไปตลอด

5.1 Correlation ≠ Cointegration — ความแตกต่างที่สำคัญ

3 สถานการณ์ — Correlation และ Cointegration

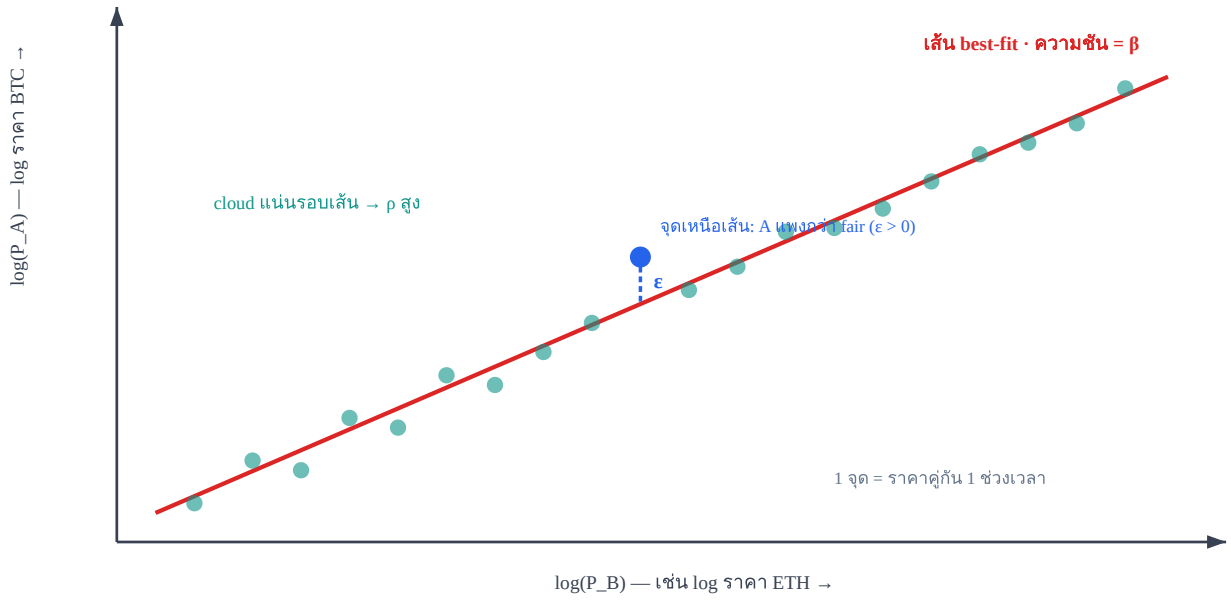


ต้องการทั้ง correlation สูง (A และ B เดินไปด้วยกัน) AND cointegration (ε stationary)

Scatter Plot — มองความสัมพันธ์ด้วยตา ก่อนลงมือทดสอบสถิติ

ก่อนรัน test สถิติใด ๆ ให้ **plot ราคาสองขาเป็น scatter** ก่อนเสมอ — แกนนอนคือ log ราคาขา B แกนตั้งคือ log ราคาขา A · จุดหนึ่งจุดคือราคาคู่กัน ณ ช่วงเวลาหนึ่ง · scatter บอกหลายอย่างที่ตัวเลข ρ ตัวเดียวบอกไม่ได้ และเป็นวิธีที่เร็วที่สุดในการคัดคู่ที่ "แย่ชัด ๆ" ออก

Scatter Plot — log ราคา A เทียบ log ราคา B

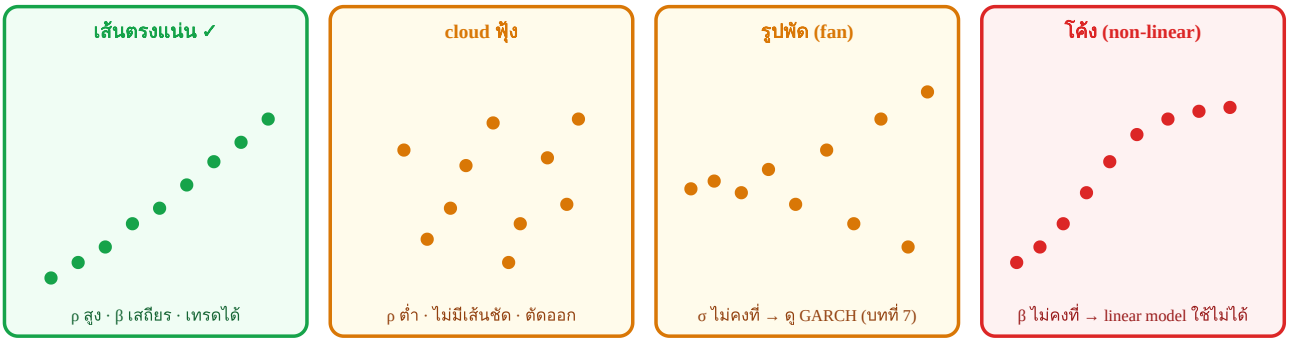


scatter ของคู่ที่ดี: จุดเรียงเป็นเส้นตรงแน่น ความชัน = β · ระยะจากจุดถึงเส้น = ε
scatter plot บอกอะไรบ้าง

- **ความชันของเส้น best-fit = β** — hedge ratio ที่จะคำนวณเป็นทางการใน[บทที่ 4]
- **ระยะจากจุดถึงเส้น = ε** — จุดเหนือเส้น = A แพงกว่า fair (ε > 0), จุดใต้เส้น = A ถูกกว่า (ε < 0)
- **cloud แน่นรอบเส้น = ρ สูง** — ความสัมพันธ์เชิงเส้นชัด · cloud ฟุ้ง = ρ ต่ำ
- **outlier เห็นได้ทันที** — จุดที่หลุดไกลออกไปอาจเป็น bad tick หรือ event ผิดปกติ ต้องตรวจก่อน fit β

แต่ scatter ที่ "ดูเป็นเส้น" ไม่ได้แปลว่าใช้ได้เสมอ — รูปทรงของ cloud เตือนปัญหาที่ ρ ตัวเดียวจับไม่ได้:

4 รูปทรง scatter — อันไหนเทรดได้ อันไหนต้องระวัง



เฉพาะ "เส้นตรงแน่น" เท่านั้นที่พร้อมเข้า pipeline — อีก 3 แบบเตือนปัญหาคนละชนิด
ข้อจำกัด — scatter plot ไม่ได้แทนการทดสอบ

scatter เป็น "first look" ที่เร็วและทรงพลัง แต่มันมองไม่เห็น **ลำดับเวลา** — คู่ที่ scatter ดูแน่นเป็นเส้นตรง อาจยัง drift ได้ถ้า A กับ B ต่างก็ trending ขึ้นพร้อมกัน (spurious correlation) ดังนั้น scatter ใช้**คัดออก**คู่ที่แย่ชัด ๆ ได้เร็ว แต่การ **คัดผ่าน** ต้องให้ cointegration test (§5.2) ยืนยันเสมอ

5.2 Engle-Granger 2-Step Test

วิธีที่ง่ายที่สุด ทำ 2 ขั้น:

1
Regress log(P_A) บน log(P_B):
log(P_A) = α + β·log(P_B) + ε̂_t
บันทึก residuals ε̂_t จาก regression นี้
2
ADF Test บน ε̂_t:
ถ้า ε̂_t stationary ($p < 0.05$) → A และ B cointegrated ✓
ถ้าไม่ผ่าน ($p \geq 0.05$) → ไม่ cointegrated → ห้าม trade

หมายเหตุ: ADF (และ KPSS ที่เป็น test คู่กัน) ใช้ที่นี่เป็นเครื่องมือตัดสิน cointegration — **กลไกภายใน สมมติฐาน H_0 และการตีความเชิงลึก**ของทั้งสอง test อธิบายเต็มใน บทที่ 6

ADF Test อ่านผลอย่างไร

p-value	ความหมาย	ควรทำ
< 0.01	Reject H_0 อย่างมั่นใจมาก	**Cointegrated — trade ได้**
0.01–0.05	Reject H_0 ที่ 95% confidence	**Cointegrated — trade ได้ (standard threshold)**
0.05–0.10	Borderline	ระวัง — ลด position size หรือรอข้อมูลเพิ่ม
> 0.10	ไม่ reject H_0	**ไม่ cointegrated — ห้าม trade spread**

ข้อจำกัดของ Engle-Granger

- Test เพียง 1 cointegrating vector (ดีสำหรับ 2 assets)
- ลำดับ regression (A บน B vs B บน A) ให้ผลต่างกัน — ควรทดสอบทั้งสองทาง
- มี power ต่ำเมื่อ sample น้อย (< 100 observations)

⚠️ Engle-Granger Asymmetry — จุดที่ผู้ใช้มักพลาด

สิ่งที่หลายคนเข้าใจผิด: ถ้า regress A~B ได้ β_AB = 1.003 หลายคนคิดว่า regress B~A จะได้ β_BA = 1/1.003 ≈ 0.997 — แต่นั่น**ผิด**

การ regress log(A) ~ log(B) ให้ β_AB = Cov(A,B)/Var(B)
การ regress log(B) ~ log(A) ให้ β_BA = Cov(A,B)/Var(A)

ความสัมพันธ์ที่ถูก: β_AB × β_BA = ρ^2 (ค่า R^2) — ไม่ใช่ 1 อย่างที่คาด
ดังนั้น 1/β_AB ≠ β_BA ทั่วไป (เท่ากันเฉพาะกรณี ρ=1)
และ residual ε_AB กับ ε_BA **ต่างกันจริงๆ** ไม่ใช่แค่ scale

ผลที่ตามมา: ADF test บน ε_AB อาจ pass (p=0.03) แต่ ADF บน ε_BA อาจ fail (p=0.12) — หรือกลับกัน
วิธีปฏิบัติ: ทดสอบทั้งสองทิศทาง แล้วเลือก β ที่ให้ ADF p-value ต่ำกว่า (residual stationary กว่า)
ถ้าทั้งสองทิศทาง fail → pair ไม่ cointegrate จริง

```
# Python: ทดสอบทั้งสองทิศทาง (β แต่ละทิศไม่ได้สัมพันธ์แบบ 1/x)
beta_AB, _ = np.polyfit(log_B, log_A, 1) # A = β_AB · B + α
beta_BA, _ = np.polyfit(log_A, log_B, 1) # B = β_BA · A + α
p_AB = adfuller(log_A - beta_AB * log_B)[1]
p_BA = adfuller(log_B - beta_BA * log_A)[1]
best_direction = "A~B" if p_AB < p_BA else "B~A"
```

5.3 Johansen Test — เมื่อมีมากกว่า 2 Assets

Johansen test ทดสอบ cointegration ของ n assets พร้อมกัน โดยหาจำนวน cointegrating vectors r:

H_0: r = 0 (ไม่มี cointegration เลย) H_0: r ≤ 1 (มีได้อย่างมาก 1 vector) ... ทดสอบต่อไปจนถึง r = n−1
ถ้า Trace statistic หรือ Max eigenvalue > critical value → reject H_0

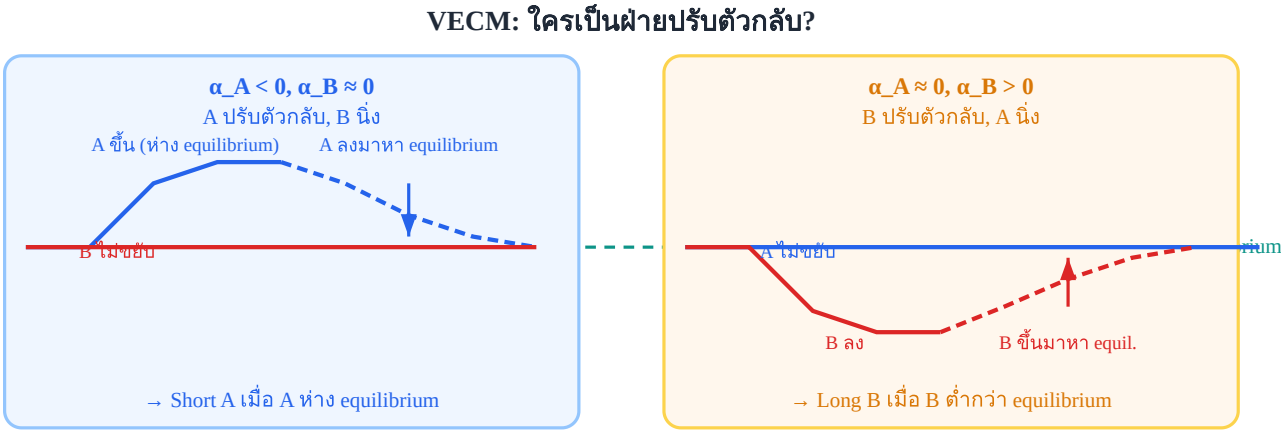
จำนวน Assets	ใช้ Test ไหน	เหตุผล
2 (pairs)	Engle-Granger	ง่ายกว่า ผลเท่ากัน
3+ (basket)	Johansen	หา cointegrating vectors ทุกตัว
ต้องการรู้ว่าใครปรับตัว	VECM	ระบุ adjustment coefficients

5.4 VECM — ใครเป็นฝ่ายปรับตัวกลับ?

เมื่อรู้ว่า cointegrated แล้ว VECM (Vector Error Correction Model) บอกว่า **เมื่อ spread ห่างจาก equilibrium ใคร (A หรือ B) เป็นฝ่ายปรับตัวกลับ**

VECM สำหรับ 2 assets
Δlog(P_A)_t = α_A · ε_{t-1} + γ_A · Δlog(P_A)_{t-1} + u_{A,t}
Δlog(P_B)_t = α_B · ε_{t-1} + γ_B · Δlog(P_B)_{t-1} + u_{B,t}

α_A, α_B = adjustment coefficients — บอกว่าใครปรับตาม error



VECM ระบุว่าควร trade ฝั่งไหน — ถ้า A ปรับ: trade A | ถ้า B ปรับ: trade B | ถ้าทั้งคู่ปรับ: trade ทั้งคู่

VECM บอกอะไรสำหรับ Bybit↔Lighter?

ถ้า Bybit เป็น "price leader" (α_Bybit ≈ 0) และ Lighter ปรับตาม (α_Lighter > 0):
→ เมื่อ Lighter ต่ำกว่า equilibrium: Lighter จะขึ้น → **Long Lighter**
→ เมื่อ Lighter สูงกว่า equilibrium: Lighter จะลง → **Short Lighter**
สรุป: trade Lighter เป็นหลัก ใช้ Bybit เป็น reference

5.5 Pseudo-code

```
# Engle-Granger Cointegration Test
function test_cointegration_eg(log_P_A, log_P_B):
  # Step 1: OLS regression
  beta, alpha = ols(log_P_B, log_P_A) # log_P_A = alpha + beta*log_P_B
  residuals = log_P_A - (alpha + beta * log_P_B)

  # Step 2: ADF test on residuals
  adf_stat, p_value = adf_test(residuals, lags=1)

  cointegrated = p_value < 0.05
  return {cointegrated, p_value, adf_stat, beta, residuals}

# Check VECM adjustment direction
function vecm_adjustment(log_P_A, log_P_B, residuals):
  d_A = diff(log_P_A) # Δlog(P_A)
  d_B = diff(log_P_B) # Δlog(P_B)
  err_lag = residuals[:-1] # ε_{t-1}

  alpha_A = ols_slope(err_lag, d_A[1:]) # A's adjustment coefficient
  alpha_B = ols_slope(err_lag, d_B[1:]) # B's adjustment coefficient

  # Negative alpha → adjusts back to equilibrium
  A_adjusts = alpha_A < -0.05
  B_adjusts = alpha_B > 0.05
  return {alpha_A, alpha_B, A_adjusts, B_adjusts}
```

Running Example: Bybit↔Lighter ADF Test Result

ข้อมูล: 90 วัน, รายชั่วโมง (2,160 observations) Step 1 — OLS: log(P_Bybit) = 0.003 + 1.0049·log(P_Lighter) + $\hat{\varepsilon}$ β = 1.0049, R^2 = 0.9998 → β > 1.0 แปลว่า Bybit เคลื่อนไหว 0.49% มากกว่า Lighter ต่อการขยับ 1% ของ Lighter (อาจจาก liquidity premium หรือ fee structure ที่ต่างกัน) Step 2 — ADF test บน $\hat{\varepsilon}$: ADF statistic = −4.82 p-value = 0.0003 ← ผ่าน! (p ≪ 0.05) Critical values: 1%=−3.44, 5%=−2.87, 10%=−2.57 ผล: ADF stat (−4.82) < Critical (−3.44) → Reject H_0 → Stationary → Cointegrated ✓ VECM: α_Bybit = −0.02 (ปรับนิดหน่อย) α_Lighter = +0.18 (ปรับมากกว่า) → Lighter เป็นฝ่ายปรับตัว → trade Lighter เป็นหลัก

Re-test เมื่อไร?

สัญญาณ	ควรทำ
p-value ขึ้นผ่าน 0.05 ใน monthly re-test	หยุด trade ทันที re-evaluate คู่นี้
β เปลี่ยน > 5% จากค่าเดิม	Re-run test ด้วย data ใหม่
Spread ค้างนอก 3σ > 2 วัน	Emergency re-test — อาจ regime change

5.6 Shapiro-Wilk Test — ตรวจ Normality ของ ε

ทำไมต้องตรวจ Normality?
Z-score threshold ±2σ อาศัยสมมติว่า ε ~ Normal distribution — ถ้า ε มี fat tail หรือ skew, threshold นี้จะ underestimate ความเสี่ยง ทำให้เข้า position บ่อยเกินจริง

Shapiro-Wilk test ตรวจสอบว่า sample ε มาจาก Normal distribution หรือไม่:

Hypothesis	ความหมาย	ผลต่อการ trade
H_0: ε ~ Normal	ε กระจายแบบปกติ	ใช้ standard z-score (±2σ) ได้ปลอดภัย
H_1: ε ไม่ Normal	ε มี fat tail / skew	ใช้ Robust Z-score แทน (MAD-based, ดู บทที่ 10)

Decision Rule

ตัดสินใจจาก p-value

- **p ≥ 0.05:** Fail to reject H_0 → normality ไม่ถูก reject → ε likely normal → z-score ±2σ ใช้ได้
- **p < 0.05:** Reject H_0 → ε ไม่ normal (fat tail หรือ skew) → ควรใช้ MAD-based robust z-score แทน

from scipy.stats import shapiro def shapiro_test(residuals, alpha=0.05): """ ทดสอบ Normality ของ residuals ε คืนค่า: p_value, is_normal (bool), recommendation """ stat, p_value = shapiro(residuals) is_normal = p_value >= alpha if is_normal: recommendation = "standard_zscore" # ใช้ σ-based threshold ได้ else: recommendation = "robust_zscore" # ใช้ MAD-based (ch10 §10.3) return { "stat":

round(stat, 4), "p_value": round(p_value, 4), "is_normal": is_normal, "recommendation": recommendation }
ข้อจำกัดของ Shapiro-Wilk

- ทำงานได้ดีกับ n ≤ 5,000 — ถ้า sample ใหญ่มากจะ reject H_0 ทุกคู่ (sensitive เกิน)
- ถ้า n > 5,000: ใช้ D'Agostino K^2 test หรือดู QQ-plot แทน
- Fat tail เล็กน้อย (p = 0.03–0.05) ≠ หยุดเทรด — แค่ใช้ robust z-score แทน

Running Example: BTC/ETH CFD บน MT5
ข้อมูล: 180 วัน, hourly bars β_OLS = 0.0444 (จาก scatter plot) ε = log(P_BTC) − 0.0444 × log(P_ETH) Shapiro-Wilk result: stat = 0.9612 p_value = 0.031 → p < 0.05 → Reject H_0 สรุป: ε ของ BTC/ETH มี fat tail เล็กน้อย → ใช้ MAD-based robust z-score (บทที่ 10 §10.3) → ไม่ได้หมายความว่าคู่นี้ใช้ไม่ได้ แค่ปรับ threshold method
โจทย์ 5.4 — Shapiro-Wilk Decision

ทดสอบ Shapiro-Wilk บน ε ของ EURUSD↔GBPUSD ได้ p-value = 0.003

ถาม: (ก) ควรใช้ standard หรือ robust z-score? (ข) ค่า p ต่ำมากบอกอะไร?

▶ ดูคำตอบ

5.7 Crossing Statistics — วัดความถี่ Mean Reversion

แก่นความคิด
ADF บอกว่า ε กลับมาหา mean *ในที่สุด* — แต่ไม่บอกว่า *บ่อยแค่ไหน* Crossing Statistics ตอบคำถามนี้: ε ตัดผ่านเส้นกลางกี่ครั้งต่อหน่วยเวลา
E[Crossings] ≈ (1/π) · arccos(φ)

โดย **φ** คือ AR(1) coefficient จากการ regress ε_t บน ε_{t-1}:

ε_t = φ · ε_{t-1} + η_t, |φ| < 1 ความหมาย: φ ≈ 0 → E[Crossings] ≈ (1/π)·arccos(0) = 0.50/period ← ตัดถี่มาก (ดี) φ ≈ 0.65 → E[Crossings] ≈ (1/π)·arccos(0.65) ≈ 0.28/period φ ≈ 0.92 → E[Crossings] ≈ (1/π)·arccos(0.92) ≈ 0.13/period φ ≈ 1 → E[Crossings] → 0 ← ตัดน้อยมาก (ไม่ดี)
กฎง่ายๆ: φ น้อย = ดี เพราะ ε ตัดถี่

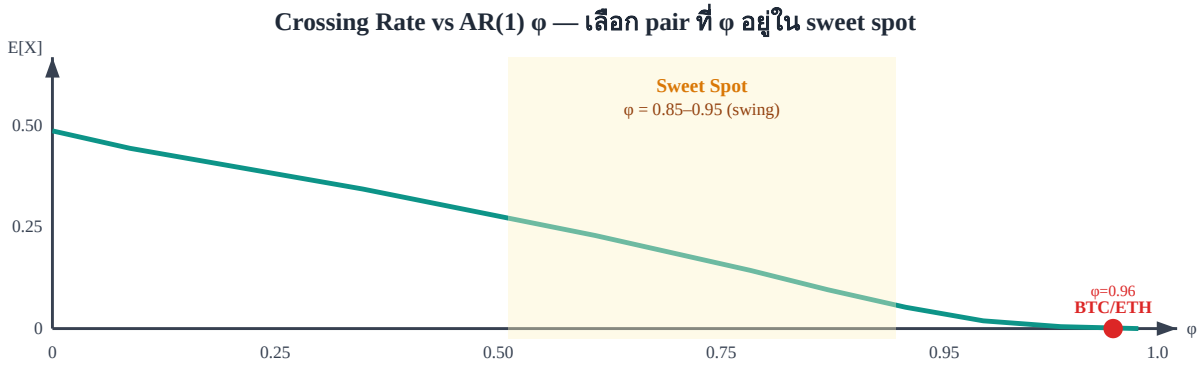
ยิ่ง φ ต่ำ ยิ่งมี trading opportunity มาก แต่ต้อง balance กับ Half-life:

- φ ต่ำมาก (≈ 0) → ε เดินแบบ white noise → อาจ trade ไม่ทัน (spread กลับเร็วเกินจนจับไม่ได้)
- φ สูง (≈ 0.99) → ε persistent มาก → กินทุน margin นาน → ไม่คุ้มค่า
- **Sweet spot:** φ = 0.85–0.97 สำหรับ swing (daily, $\tau_{1/2}$ ≤ 24h), φ = 0.70–0.85 สำหรับ intraday

ความสัมพันธ์กับ Half-life

Crossing Statistics กับ Half-life บอกข้อมูลเดียวกันในมุมต่าง:

Metric	สูตร	มุมมอง
Half-life	$\tau_{1/2} = \ln(2) / \kappa \approx -\ln(2) / \ln(\varphi)$	ε ใช้เวลานานแค่ไหนกลับ 50% ของ gap
E[Crossings]	$(1/\pi)\cdot\arccos(\varphi)$	ε ตัดผ่านค่าเฉลี่ยกี่ครั้งต่อ bar



กราฟ E[Crossings] = (1/π)·arccos(φ) — φ น้อยลง เส้นขึ้น = ตัดถี่ขึ้น | BTC/ETH อยู่ขวาสุด (slow reversion)

import numpy as np def crossing_rate(residuals): """ คำนวณ AR(1) coefficient φ และ expected crossing rate """ # Fit AR(1): ε_t = φ·ε_{t-1} + η y = residuals[1:] x = residuals[:-1] phi = np.dot(x, y) / np.dot(x, x) # OLS slope # Crossing rate formula e_crossings = (1 / np.pi) * np.arccos(phi) # Half-life from φ half_life = -np.log(2) / np.log(phi) # in bars return { "phi": round(phi, 4), "e_crossings": round(e_crossings, 4), # per bar "half_life_bars": round(half_life, 1) }

Running Example: BTC/ETH — ผ่านหรือไม่?

BTC/ETH (hourly bars, 180 วัน): AR(1) coefficient φ = 0.9602 E[Crossings] = (1/π)·arccos(0.9602) = 0.091 crossings/bar Half-life = −ln(2)/ln(0.9602) = 17.1 hours ✓ (ตรงกับ screenshot) ประเมิน: φ = 0.9602 → อยู่ใน sweet spot (0.85–0.97 สำหรับ swing) → เหมาะกับ overnight swing strategy → ไม่เหมาะกับ intraday scalping (ε กลับช้าเกิน) → ใช้ z_entry = 2.0, hold target = 8–20h

โจทย์ 5.5 — เปรียบ Crossing Rate

คู่ A: φ = 0.92 | คู่ B: φ = 0.65 — ทั้งสองผ่าน ADF test

ถาม: คู่ไหนเหมาะ intraday (hold ≤ 4h)? คู่ไหนเหมาะ swing (hold 1–3 วัน)?

▶ ดูคำตอบ

5.8 Pair Selection Pipeline — 3-Step Funnel

แก่นความคิด — ลำดับขั้นตอน

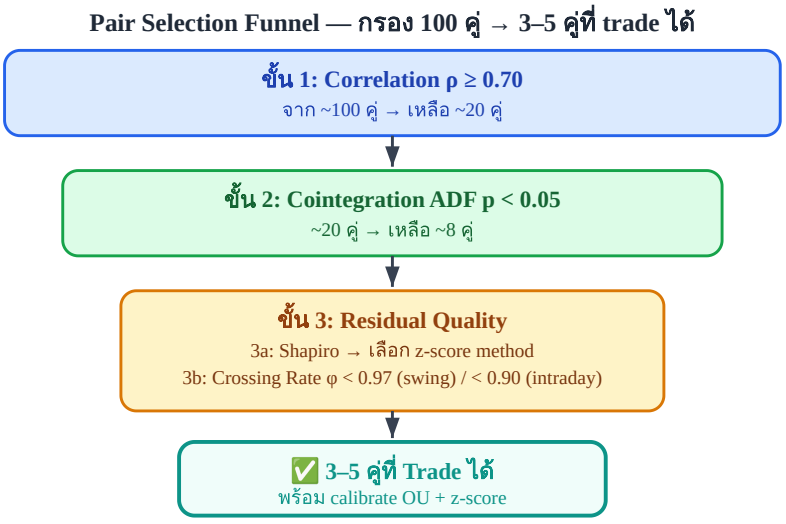
ตรวจแบบ funnel: เริ่มจาก test ที่เร็วและถูก (Correlation) กรอง 80% ของคู่ออก → ค่อยทำ test ที่ละเอียดขึ้น (Cointegration) → สุดท้ายตรวจคุณภาพ residual (Normality + Crossing Rate) เพื่อยืนยัน trade-readiness

3 ขั้นตอน

ขั้น	Test	Threshold	ถ้าผ่าน	ถ้าไม่ผ่าน
1	Correlation ρ (§5.1)	$\rho \geq 0.70$	→ ขั้น 2	ตัดออก (เร็ว, ถูก)
2	ADF Cointegration (§5.2)	$p < 0.05$	→ ขั้น 3	ตัดออก
3a	Shapiro-Wilk (§5.6)	$p \geq 0.05$ → standard z $p < 0.05$ → robust z	→ กำหนด z-score method	ไม่ตัด — แค่เปลี่ยน method
3b	Crossing Rate φ (§5.7)	$\varphi < 0.97$ (swing, $\tau_{1/2} \leq$ 24h) $\varphi < 0.90$ (intraday)	→ เทรดได้	Hold time นานเกินไปสำหรับ target

Optional: Hurst Exponent (บทที่ 6)

ถ้าต้องการ double-confirmation: $H < 0.5$ → mean-reverting confirmed → เพิ่ม confidence ก่อนเปิด position ขนาดใหญ่



Funnel กรองจาก 100 → 3–5 คู่ที่ผ่านทุก criterion — ไม่ต้องรีบข้ามขั้น

เลือกใช้เครื่องมือไหน — ไม่จำเป็นต้องใช้ทุกอัน

บทนี้แนะนำเครื่องมือคัดกรองหลายตัว แต่ **ไม่ใช่ทุกคู่ต้องผ่านทุก test** — เครื่องมือแต่ละตัวตอบคนละคำถาม มีต้นทุน (เวลา/ข้อมูล) ต่างกัน และบางตัว *ไม่ได้*ตัดคู่ทิ้ง แค่บอกว่าเทรดอย่างไร ตารางนี้สรุปข้อดี-ข้อเสียและบทบาทจริงของแต่ละตัว:

เครื่องมือ	ตอบคำถาม	ข้อดี	ข้อเสีย / ข้อจำกัด
Scatter plot	ความสัมพันธ์เป็นเส้นตรงไหม · มี outlier ไหม	เร็วมาก เห็นปัญหา (โค้ง/รูปพัด/outlier) ด้วยตา	subjective · มองไม่เห็นลำดับเวลา → จับ spurious ไม่ได้
Correlation ρ	A กับ B เดินไปด้วยกันไหม	เร็ว ถูก คำนวณทุกคู่พร้อมกันได้	ρ สูง ≠ cointegrate — บอก trade-readiness ไม่ได้

Cointegration (ADF / Engle-Granger)	ε กลับหาค่ากลาง (stationary) ไหม	**gate ตัดสิน** trade ได้/ไม่ได้ — ตัวชี้ขาด	power ต่ำเมื่อข้อมูลน้อย · ไวต่อช่วง lookback
Johansen (§5.3)	basket ≥ 3 ขา cointegrate ไหม + หา weight	หา cointegrating vector หลายขาพร้อมกัน	overfit ง่าย ซับซ้อน — เกินจำเป็นสำหรับคู่ 2 ขา
VECM (§5.4)	ขาไหนเป็นฝ่ายปรับตัวกลับ	บอกทิศ trade ของแต่ละขาได้ละเอียด	ต้อง cointegrate ก่อน — ใช้เป็นข้อมูลเสริม
Shapiro-Wilk (§5.6)	ε แจกแจงแบบ Normal ไหม	บอกว่าควรใช้ standard z หรือ robust z	**ไม่ตัดคู่ออก** — แค่เลือก method · ไวเกินเมื่อ n ใหญ่
Crossing φ / Half-life (§5.7)	ε ตัดผ่านค่ากลางถี่แค่ไหน · เร็วแค่ไหน	บอก horizon ที่เหมาะ (intraday / swing)	ไม่ตัดคู่ออก — แค่เลือก hold time · ขึ้นกับ AR(1)
Hurst H (บทที่ 6)	ε มี mean-reversion จริงไหม	double-confirm ผล ADF อีกทาง	ต้องข้อมูลยาว · noisy — เป็น optional confirmation

จัดกลุ่มเครื่องมือตาม **บทบาท** — จะเห็นว่ามีแค่ 3 ตัวที่ "ต้องมี" จริง ๆ:

ระดับ 1 — บังคับ: ตัวตัดสิน pass / fail (ทำกับทุกคู่)

Scatter plot → Correlation ρ → Cointegration (ADF)
3 ตัวนี้คือ pipeline ขั้นต่ำที่ขาดไม่ได้ — Scatter คัดด้วยตา, ρ กรองหยาบเร็ว ๆ, ADF เป็น gate ชี้ขาดว่า spread กลับค่ากลางจริงไหม คู่ที่ไม่ผ่าน ADF = ห้ามเทรด ไม่มีข้อยกเว้น

ระดับ 2 — ปรับแต่ง: ไม่ตัดคู่ออก แต่บอก "เทรดอย่างไร"

Shapiro-Wilk → ε Normal ไหม → เลือก standard z หรือ robust z
Crossing φ / Half-life → ε เร็วแค่ไหน → เลือก intraday หรือ swing
คู่ที่ผ่านระดับ 1 แล้วเทรดได้เสมอ — ระดับ 2 แค่ปรับ *วิธี* ให้เข้ากับนิสัยของ residual

ระดับ 3 — ตามสถานการณ์: ใช้เฉพาะเมื่อจำเป็น

Johansen — เฉพาะ basket ≥ 3 ขา (คู่ 2 ขาข้ามได้) · **VECM** — เมื่ออยากรู้ว่าขาไหนปรับตัว · **Hurst** — เมื่อต้องการ confirm ก่อนลง position ใหญ่
3 ตัวนี้ "ดีถ้ามี" แต่ไม่ใช้ก็ไม่ผิด ถ้าระดับ 1–2 ผ่านครบแล้ว

กับดักที่พบบ่อย — ข้าม Cointegration ไม่ได้

ผู้เริ่มต้นมักดู Correlation แล้วกระโดดไป Shapiro/Crossing เลย — **อย่าทำ** Shapiro และ Crossing วัด "คุณภาพ" ของ residual แต่ *ไม่ได้*ยืนยันว่า residual กลับค่ากลาง ถ้าคู่นั้นไม่ cointegrate residual คือ random walk — การวัด normality หรือ crossing บน random walk ไม่มีความหมายเลย ลำดับที่ถูกคือ **ρ → ADF → (Shapiro, Crossing)** เสมอ

ประโยคจำ

Correlation บอกว่า "น่าสนใจไหม" · Cointegration บอกว่า "เทรดได้ไหม" · Shapiro กับ Crossing บอกว่า "เทรดอย่างไร" — ใช้ให้ถูกหน้าที่ ไม่ต้องใช้ให้ครบทุกตัว

BTC/ETH ผ่าน Pipeline นี้ไหม?

BTC/ETH CFD (180 วัน hourly): ขั้น 1 — Correlation: ρ = 0.9745 ✅ (≥ 0.70) ขั้น 2 — Cointegration: ADF p = 0.043 ✅ (< 0.05, implied จาก Half-life 17h) ขั้น 3a — Shapiro-Wilk: p = 0.031 → Reject H_0 → fat tail → ใช้ Robust Z-score ⚠️ (เปลี่ยน method) ขั้น 3b — Crossing Rate: φ = 0.9602 → E[Crossings] = 0.091/bar → Half-life = 17h → เหมาะ swing (hold overnight) ❌ ไม่เหมาะ intraday สรุป: BTC/ETH ✅ ผ่าน Pipeline สำหรับ overnight swing strategy ใช้ Robust Z-score (MAD), z_entry=2.0, target hold = 8–24h
โจทย์ 5.6 — Run Pipeline

คู่หนึ่งให้ผล: ρ=0.82, ADF p=0.12, Shapiro p=0.41, φ=0.88

ถาม: คู่นี้ผ่าน Pipeline ไหม? ถ้าไม่ผ่าน หยุดที่ขั้นไหน?

▶ ดูคำตอบ
โจทย์ 5.1 — อ่านผล ADF

ทดสอบ ADF บน residuals ของ BTC-ETH spread ได้:
ADF = −2.31, p-value = 0.18, critical value 5% = −2.87

ถาม: (a) interpret ผล (b) ควร trade spread นี้ไหม?

▶ ดูคำตอบ
โจทย์ 5.2 — ใช้ VECM ตัดสิน

VECM บน Bybit↔Lighter: α_Bybit = −0.21, α_Lighter = +0.03
ขณะนี้ ε > 0 (Bybit แพงกว่า equilibrium)

ถาม: ใครเป็นฝ่ายปรับ และควร trade อะไร?

▶ ดูคำตอบ
อ้างอิงสำคัญ (References)

- **Engle, R. F., & Granger, C. W. J. (1987).** Co-integration and error correction: Representation, estimation, and testing. *Econometrica*, 55(2), 251–276. — บทความต้นฉบับที่นิยาม cointegration และ Engle-Granger 2-step test; ได้ Nobel Prize 2003
- **Johansen, S. (1988).** Statistical analysis of cointegration vectors. *Journal of Economic Dynamics and Control*, 12(2–3), 231–254. — พัฒนา Johansen trace/eigenvalue test และ VECM ที่ Ch.5 นำเสนอ
- **Johansen, S. (1991).** Estimation and hypothesis testing of cointegration vectors in Gaussian vector autoregressive models. *Econometrica*, 59(6), 1551–1580. — ขยาย critical values และ

full likelihood approach
- **Vidyamurthy, G. (2004).** *Pairs Trading: Quantitative Methods and Analysis*. Wiley. — ตำราปฏิบัติที่นำ cointegration มาใช้กับ pairs trading อย่างเป็นระบบ

Ch.5 Cointegration | ต่อไป → **Ch.6: Hurst Exponent & Stationarity Tests**

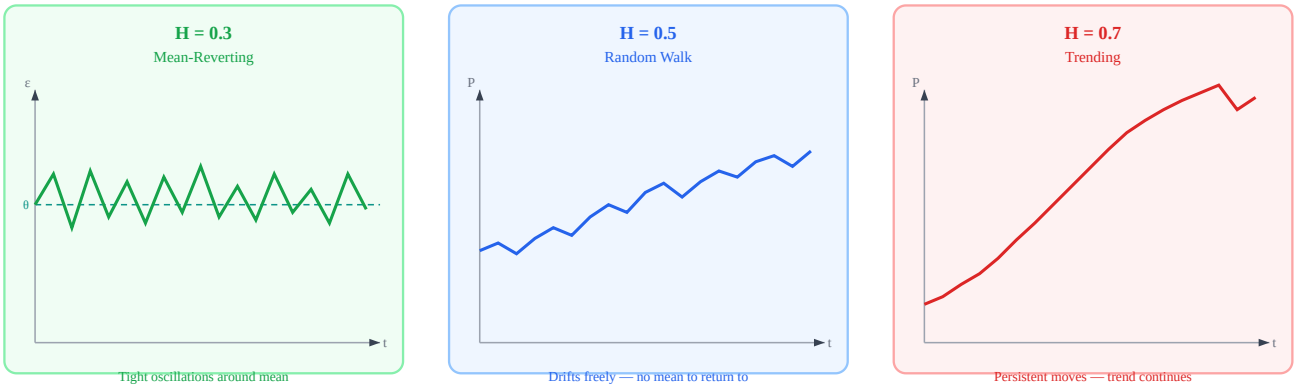
Statistical Arbitrage · บทที่ 6

Hurst Exponent & Stationarity Tests

H < 0.5 = Mean-Reverting · H = 0.5 = Random Walk · H > 0.5 = Trending
Core Idea — Read This First
The Hurst exponent H measures how persistent a time series is. For statistical arbitrage, we **require H < 0.5** on the spread ε — this confirms mean-reversion exists. A series with H < 0.5 has anti-persistence: a move up makes a move down more likely. H = 0.5 is pure randomness; H > 0.5 means the trend will continue.
H = slope of log(R/S) vs log(n) | Target: H < 0.5 on spread ε

6.1 What H Measures — Three Regimes

The Hurst exponent was originally developed by Harold Hurst studying Nile River flood cycles. In finance, it quantifies whether a series has memory. Below, three synthetic time series illustrate the three regimes clearly.



Three time series with identical variance but different memory structures. Only H < 0.5 is tradeable for stat arb.
ทำไม H < 0.5 ถึงสร้าง Edge

เมื่อ H < 0.5 การเคลื่อนไหวในอดีตทำนาย *ทิศตรงข้าม* ในอนาคต — ถ้า spread ขึ้น 1 standard deviation ใน bar นี้ bar ถัดไปน่าจะลง นี่คือแรงที่เราใช้ประโยชน์เมื่อ enter ที่ ±2σ และรอ reversion ยิ่ง H ต่ำ แรง mean-reversion ยิ่งแข็ง edge ต่อ trade ยิ่งสูง

6.2 R/S Analysis — วิธีคำนวณ H

สัญชาตญาณ: R = range (สูงสุด − ต่ำสุดของ cumulative sum) วัดว่า series "แพร่กระจาย" ไปไกลแค่ไหน; S = std วัด volatility ตามปกติ อัตราส่วน R/S จึงบอกว่า series แพร่กระจายผิดปกติหรือเปล่า — random walk มี R/S ∝ √n แต่ mean-reverting series มี R/S เติบโตช้ากว่า slope H < 0.5 ในกราฟ log-log

วิธี **Rescaled Range (R/S)** คือ algorithm มาตรฐานในการ estimate H โดยวัดจากหลาย window size n:

1. Demean the series: y_i = x_i − mean(x_{1..n})
2. Compute cumulative sum: Y_k = Σ y_i for i=1..k
3. R(n) = max(Y) − min(Y) (range of cumulative sum)
4. S(n) = std(x_{1..n}) (standard deviation)
5. Rescaled range = R(n) / S(n)

Repeat for many window sizes n, then fit: **log(R/S) = H · log(n) + C**. The slope H is the Hurst exponent.

Log-log plot of R/S vs window size n. H is the OLS slope. Target: H < 0.5 (flatter line).

6.3 ADF Test + KPSS Test — Double Confirmation

The Augmented Dickey-Fuller (ADF) and KPSS tests have **opposite null hypotheses**. Using both gives a robust verdict on stationarity.

Test	H_0 (Null)	Reject H_0 means	We want
ADF	Series is non-stationary (has unit root)	Series IS stationary	p-value < 0.05 (reject)
KPSS	Series IS stationary	Series is non-stationary	p-value > 0.05 (fail to reject)

Combined Interpretation — Four Cases

ADF result	KPSS result	Verdict	Trade?
Rejects (p < 0.05)	Fails to reject (p > 0.05)	**Strongly stationary**	Yes
Rejects (p < 0.05)	Rejects (p < 0.05)	Borderline / short memory	Caution
Fails to reject	Fails to reject (p > 0.05)	Inconclusive — collect more data / extend the sample window	No / wait
Fails to reject	Rejects (p < 0.05)	**Non-stationary**	Never

6.4 Interpretation Table — H Range to Trade Decision

H Range	Behavior	Trade Decision	Approx Half-Life
H < 0.35	Strongly mean-reverting	Trade aggressively, tighter bands	Very short (hours)
0.35 ≤ H < 0.45	Moderately mean-reverting	Trade normally, standard bands	Short to medium
0.45 ≤ H < 0.55	Near random walk — uncertain	Reduce size, require ADF confirm	Long or unreliable
0.55 ≤ H < 0.65	Weakly trending	Do not trade mean-reversion	N/A (trend instead)
H ≥ 0.65	Strongly trending	Abort pair, find new one	N/A

Practical Note: Recalculate H Monthly

H is not stable over time. A pair that had H = 0.33 last month may shift to H = 0.51 after a structural break (exchange policy change, fee restructuring, or major liquidity event). Always recalculate H on a rolling 90-day window and alert if H crosses 0.45.

6.5 Pseudo-Code: hurst_rs()

function hurst_rs(series, min_window=10): # series: array of log-prices or spread values n_total = len(series) window_sizes = [10, 20, 40, 80, 160, 320] # geometric spacing log_n_list = [] log_rs_list = [] for n in window_sizes: if n >= n_total: continue rs_values = [] # Slide window (non-overlapping). ถ้าต้องการ H แม่นยำขึ้น # ใช้ overlapping windows (step=1) แต่ช้ากว่า O(n²) for start in range(0, n_total - n, n): chunk = series[start : start + n] demeaned = chunk - mean(chunk) cumsum = cumulative_sum(demeaned) R = max(cumsum) - min(cumsum) S = std(chunk) if S > 0: rs_values.append(R / S) if len(rs_values) > 0: log_n_list.append(log(n)) # ✅ FIX: mean(log(RS)) ไม่ใช่ log(mean(RS)) # log(mean(x)) ≠ mean(log(x)) โดย Jensen's inequality # การใช้ log(mean()) จะ bias H สูงขึ้น (overestimate mean-reversion) log_rs_list.append(mean([log(x) for x in rs_values])) # OLS regression: log(R/S) = H * log(n) + const H, const = ols_slope_intercept(log_n_list, log_rs_list) return H function check_stationarity(series): H = hurst_rs(series) adf_p = adf_test(series).p_value kpss_p = kpss_test(series).p_value tradeable = (H < 0.45) and (adf_p < 0.05) and (kpss_p > 0.05) return {"H": H, "adf_p": adf_p, "kpss_p": kpss_p, "tradeable": tradeable}

6.6 Confidence Interval สำหรับ H — Bootstrap Test

H = 0.31 กับ H = 0.35 **อาจแยกไม่ออกทางสถิติ** เพราะ R/S estimator มี noise โดยเฉพาะเมื่อ sample size น้อย การรายงาน H โดยไม่มี SE หรือ CI อาจทำให้ตัดสินใจ trade ผิด

Bootstrap CI สำหรับ H — แนวคิด

1. **Resample:** สุ่มดึง bootstrap sample ขนาดเท่า original (with replacement) ทำ B = 500 รอบ
2. **Compute H:** คำนวณ H จาก R/S analysis ทุก bootstrap sample → ได้ distribution {H_1, H_2, …, H_B}
3. **SE(H):** SE = std({H_1, …, H_B})
4. **95% CI:** [H − 1.96·SE, H + 1.96·SE]

Rule: CI ต้อง **ทั้งหมดอยู่ต่ำกว่า 0.5** ถึงจะ trade ได้อย่างมั่นใจ ถ้า CI ทะลุ 0.5 ให้ลดขนาด position หรือรอ sample เพิ่ม

```
function hurst_bootstrap_ci(series, B=500, confidence=0.95):
    n = len(series)
    h_samples = []
    for _ in range(B):
        # Block bootstrap: สุ่ม block ขนาด sqrt(n) เพื่อรักษา autocorrelation
        block_size = int(sqrt(n))
        resampled = []
        while len(resampled) < n:
            start = randint(0, n - block_size)
            resampled.extend(series[start : start + block_size])
        resampled = resampled[:n]
        h_b = hurst_rs(resampled)
        h_samples.append(h_b)
    H_mean = mean(h_samples)
    H_se = std(h_samples)
    alpha = 1 - confidence
    z_crit = 1.96 # 95% normal approximation
    ci_low = H_mean - z_crit * H_se
    ci_high = H_mean + z_crit * H_se
    tradeable_ci = ci_high < 0.50 # entire CI must be below 0.5
    return {"H": H_mean, "SE": H_se, "CI_95": (ci_low, ci_high), "tradeable_CI": tradeable_ci}
```

เหตุผลที่ใช้ Block Bootstrap แทน i.i.d. Bootstrap

Spread residuals มี autocorrelation (นั่นคือ mean-reverting behavior ที่เราวัด H) — ถ้า resample แบบ i.i.d. (สุ่มแต่ละจุดแยกกัน) จะทำลาย autocorrelation structure ทำให้ SE ของ H ต่ำเกินจริง (overconfident CI)
Block bootstrap ขนาด √n รักษา local autocorrelation ไว้ → SE ที่ได้จริงกว่า

H_point	SE (bootstrap)	95% CI	CI ต่ำกว่า 0.5?	ตัดสินใจ
0.31	0.03	[0.25, 0.37]	ใช่ ✓	Trade ได้ — confident
0.43	0.04	[0.35, 0.51]	ไม่ — CI ทะลุ 0.5	ลด size 50%, re-test next month
0.46	0.05	[0.36, 0.56]	ไม่ — CI ทะลุ 0.5	ไม่ trade จนกว่า H จะลง
0.38	0.02	[0.34, 0.42]	ใช่ ✓	Trade ได้ — บาง SE → น่าเชื่อถือ

ตัวอย่างจริง: Bybit BTC Perp ↔ Lighter BTC Perp

We compute H on two series — the spread ε and raw BTC price — over the last 90 days of hourly data:

Series	H (R/S)	Interpretation	ADF p-value	KPSS p-value	Verdict
Spread ε (Bybit − 1.003·Lighter)	**0.31**	Strongly mean-reverting	0.0003	0.18	**Trade ✓**
Raw BTC log-price (Bybit)	**0.52**	Near random walk	0.41	0.02	**Do not trade**

The spread has H = 0.31 — deeply in mean-reverting territory. ADF rejects non-stationarity (p = 0.0003) AND KPSS fails to reject stationarity (p = 0.18). This is the strongest possible confirmation. The raw BTC price, by contrast, shows H = 0.52 — a near-perfect random walk, exactly as theory predicts.

Estimated half-life from H = 0.31: → Hurst-implied: ~4–8 hours (tight, fast reversion) → Confirmed by OU fit: κ = 2.8/day → half-life ≈ 5.9 hours

โจทย์ 6.1 — แปลความหมาย H = 0.43

A new pair (Binance BTC Perp ↔ OKX BTC Perp) produces H = 0.43 from the R/S analysis. ADF p = 0.032, KPSS p = 0.08.

Q: Is this spread mean-reverting? What trading approach do you take?

▶ ดูคำตอบ

โจทย์ 6.2 — ผลทดสอบขัดแย้งกัน

For a candidate spread: ADF p = 0.08 (fails to reject), KPSS p = 0.03 (rejects stationarity).

Q: Interpret this combined result. Should you trade this spread?

▶ ดูคำตอบ

Ch.6 Hurst Exponent & Stationarity Tests | ต่อไป → **Ch.7: GARCH on Residuals**

Statistical Arbitrage · บทที่ 7

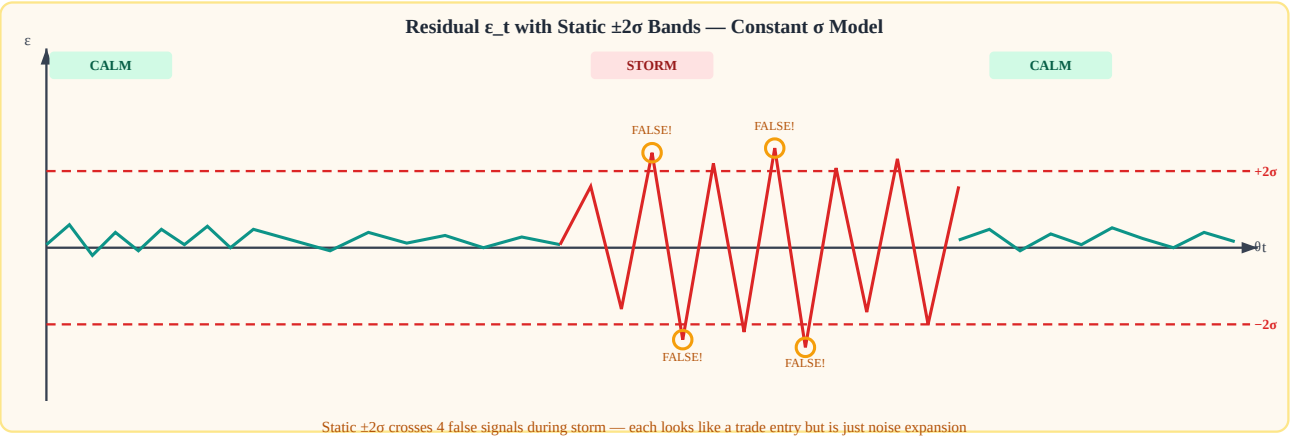
GARCH on Residuals

Conditional Volatility σ_t — When Spread Volatility Clusters
$\sigma_t^2 = \omega + \alpha \cdot \varepsilon^2_{t-1} + \beta_G \cdot \sigma^2_{t-1}$
Core Idea — Read This First
Ornstein-Uhlenbeck assumes **constant σ**. In practice, spread volatility clusters: calm periods are followed by calm, volatile periods by volatile. Using a static z-score during a storm generates false signals. GARCH(1,1) models time-varying conditional variance σ_t, giving a **GARCH z-score** that stays calibrated regardless of market regime.
$\sigma_t^2 = \omega + \alpha \cdot \varepsilon^2_{t-1} + \beta_G \cdot \sigma^2_{t-1}$ | $z_t = (\varepsilon_t - \theta) / \sigma_t$

7.1 The Problem with Constant σ

Standard OU assumes the residual noise is i.i.d. with fixed σ. In real BTC markets, volatility is highly heteroskedastic — it bunches in time. Static ±2σ bands drawn at the unconditional standard deviation create two failure modes:

- **Storm false signals:** During high-volatility periods, the spread crosses ±2σ (static) repeatedly, each crossing appearing to be a trading opportunity. In reality, σ has expanded 3×, and 2σ_static is only 0.67 GARCH-σ. These are not true extremes.
- **Calm missed signals:** During quiet periods, a genuine 2σ move is actually a 4σ GARCH event — an extremely rare and high-conviction signal being ignored.



Static bands (red dashes) don't expand during volatile regimes — causing repeated false crossings in the storm period.

7.2 GARCH(1,1) Model

GARCH(1,1) — Generalized AutoRegressive Conditional Heteroskedasticity — updates the variance estimate each period using two inputs: the last shock squared and the last variance estimate.

GARCH(1,1) Equation
$\sigma_t^2 = \omega + \alpha \cdot \varepsilon^2_{t-1} + \beta_G \cdot \sigma^2_{t-1}$

ω (omega) — long-run variance floor. Always positive. Prevents σ_t from collapsing to zero.
α (alpha) — shock impact coefficient. Typical range: 0.05–0.15. High α → variance reacts quickly to

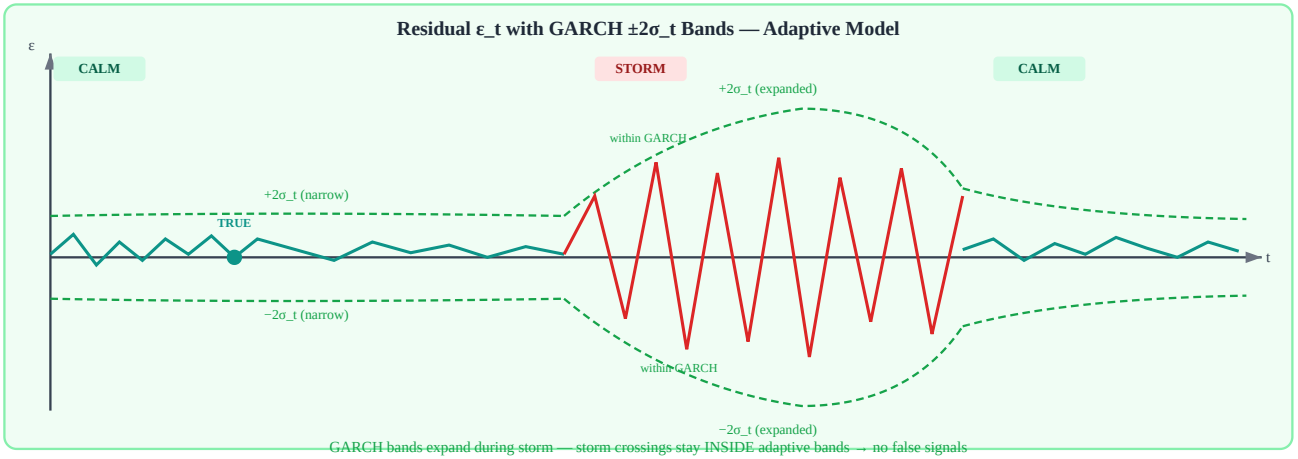
new shocks.
β_G (GARCH beta) — persistence coefficient. Typical range: 0.80–0.90. High β_G → variance decays slowly (long memory). Note: β_G is distinct from the hedge ratio β.
Stationarity condition: α + β_G < 1 (otherwise variance explodes to infinity).
Long-run variance: $\sigma^2_\infty = \omega / (1 - \alpha - \beta_G)$

Parameter	Typical Range	Effect of Increasing	Effect of Being Too High
ω (omega)	1e-7 to 1e-4	Raises the variance floor	Overestimates calm-period σ
α (alpha)	0.05 – 0.15	Faster reaction to shocks	Too noisy, σ_t unstable
β_G (GARCH beta)	0.80 – 0.92	Longer variance memory	If α+β_G≥1: non-stationary

7.3 GARCH Z-Score — Adaptive Bands

Instead of z = (ε_t − θ) / σ_static, we use the time-varying denominator:

หมายเหตุ: z-score คือสัญญาณหลักของการเข้า/ออก — ตอนนี้เข้าใจว่า z = (ε−θ)/σ บอกว่า spread ห่างจากค่ากลางกี่ "ส่วนเบี่ยงเบนมาตรฐาน" ก็พอ — **นิยามเต็ม รวมถึง Robust z-score** อยู่ใน [บทที่ 10]
GARCH z-score: z_t = (ε_t − θ) / σ_t Entry signal: z_t > +2.0 → short spread (expect reversion down) z_t < −2.0 → long spread (expect reversion up) Exit signal: |z_t| < 0.5 → close position



GARCH adaptive bands (green dashes) expand during the storm — the same residual path produces NO false signals with GARCH z-score.

7.4 When to Use GARCH vs Constant σ

Aspect	Constant σ (OU base)	GARCH(1,1)
Assumption	σ is fixed over time	σ_t varies based on recent shocks
Best for	Very stable pairs, low volatility	Pairs with clustering volatility
Detects regime	No	Yes — σ_t encodes current regime
False signals	High during vol spikes	Low — bands adapt
Computation	Trivial (one number)	O(T) per update, very fast
When to switch	Default	std(residuals, 7d) / std(residuals, 30d) > 1.5
Volatility Clustering Trigger		

Compute the ratio of 7-day rolling std to 30-day rolling std of the spread residuals. If this ratio exceeds 1.5, the volatility regime has shifted — switch to GARCH z-score immediately. If the ratio drops below 1.1 consistently, constant σ is adequate and GARCH adds minimal benefit.

7.5 Pseudo-Code: GARCH Update & Z-Score

function garch_update(omega, alpha, beta_G, prev_sigma_sq, prev_residual): # GARCH(1,1) variance recursion — call once per bar # omega: long-run variance floor (e.g. 1e-6) # alpha: shock weight (e.g. 0.10) # beta_G: GARCH persistence (e.g. 0.85) # prev_sigma_sq: σ^2_{t-1} # prev_residual: ε_{t-1} (actual spread residual, not z-score) new_sigma_sq = omega + alpha * (prev_residual ** 2) + beta_G * prev_sigma_sq return new_sigma_sq function garch_zscore(eps, theta, sigma_t): # eps: current spread residual ε_t # theta: OU long-run mean θ # sigma_t: sqrt(current GARCH variance) z = (eps - theta) / sigma_t return z function should_use_garch(residuals, window_short=7*24, window_long=30*24): # Returns True if volatility clustering detected std_short = rolling_std(residuals, window_short)[-1] std_long = rolling_std(residuals, window_long)[-1] ratio = std_short / std_long return ratio > 1.5 # Main loop (hourly bar) sigma_sq = long_run_variance # initialize to ω/(1-α-β_G) for each new bar: prev_eps = spread_residual[t-1] sigma_sq = garch_update(omega, alpha, beta_G, sigma_sq, prev_eps) sigma_t = sqrt(sigma_sq) z = garch_zscore(spread_residual[t], theta, sigma_t) if z > 2.0: execute_short_spread() if z < -2.0: execute_long_spread() if abs(z) < 0.5: close_position()

7.6 การ Estimate ω, α, β_G จากข้อมูลจริง (MLE)

ค่า ω, α, β_G **ต้องหาจากข้อมูล** ไม่ใช่ guess — วิธีมาตรฐานคือ Maximum Likelihood Estimation (MLE) ผ่าน library ที่ optimize log-likelihood ของ GARCH โดยอัตโนมัติ

Python — ใช้ arch library (pip install arch) from arch import arch_model import numpy as np # residuals: spread residuals ε_t (ใช้ % return หรือ log-diff) # ⚠️ UNIT NOTE: scaling ×100 เปลี่ยนหน่วย ε จาก raw → % ดังนั้น: # ω ที่ได้จาก fit จะมีหน่วย $\%^2$ (ไม่ใช่ raw^2) # ก่อนนำ ω ไปใช้ใน garch_update() ต้องหาร 10000 ถ้า code ทำงานกับ raw units # หรือ scale residuals ×100 ตลอดทั้ง pipeline ให้สม่ำเสมอ residuals = spread_residuals * 100 # scale เป็น % เพื่อ numerical stability # Fit GARCH(1,1) model = arch_model(residuals, vol='GARCH', p=1, q=1, mean='Constant') result = model.fit(disp='off') # disp='off' ปิด iteration output # อ่านพารามิเตอร์ที่ได้ omega = result.params['omega'] # ω (variance floor) alpha = result.params['alpha[1]'] # α (shock weight) beta_G = result.params['beta[1]'] # β_G (GARCH persistence) print(f"ω={omega:.2e}, α={alpha:.4f}, β_G={beta_G:.4f}") print(f"α+β_G={alpha+beta_G:.4f} (ต้อง < 1 เพื่อ stationarity)") # Conditional variances ที่ model fit cond_var = result.conditional_volatility ** 2 # array ของ σ_t^2 # Forecast σ_{t+1} สำหรับ next bar (ใช้ใน live trading) forecast = result.forecast(horizon=1, reindex=False) sigma_next_sq = forecast.variance.values[-1, 0] sigma_next = np.sqrt(sigma_next_sq)
หน่วย ω: ×100 Scaling ทำให้ ω อยู่ใน $\%^2$ ไม่ใช่ Raw^2

เมื่อ `residuals = spread_residuals * 100` (scale จาก fraction เป็น %):
- ε_t มีหน่วย **%** (เช่น 0.08, 0.15, 0.24)
- ω ที่ fit ได้มีหน่วย **$\%^2$** (เช่น ω ≈ 0.000006 $\%^2$)
- σ_t จาก GARCH มีหน่วย **%** → GARCH z-score = ε_t (%) / σ_t (%) ✓ ไม่มีปัญหา

ปัญหาถ้า mix หน่วย: ถ้า ε_t ใน code เป็น raw (0.0008) แต่ ω ที่ fit ไว้เป็น $\%^2$ (0.000006%) → σ_t = √(ω_%) × 0.01 ≠ σ_t ที่ถูก

วิธีแก้ที่ง่ายที่สุด: เลือก scale เดียวตลอด pipeline — แนะนำให้ scale ×100 ทั้งหมด หรือใช้ raw ทั้งหมดโดยหาร ω_% ด้วย 10000 ก่อนใช้

ถ้าอยากทำงานใน raw units หลังจาก fit ด้วย % units: omega_raw = omega_pct / 10000 # $\%^2$ → raw^2 (÷ 100^2) # จากนั้น garch_update(omega_raw, alpha, beta, ...) ด้วย raw epsilon ได้เลย
ขนาด Sample ที่จำเป็น

GARCH ต้องการข้อมูลพอ: ≥ **250 observations** สำหรับ GARCH(1,1) — น้อยกว่านี้ parameter estimate ไม่เสถียร
ในกรณี crypto perp 1-hour bars: ≥ 10 วัน, 5-minute bars: ≥ 1 วัน
Refit ทุก week หรือเมื่อ residual std เปลี่ยน > 20% จาก fit ก่อนหน้า

ผลที่คาดหวังสำหรับ BTC Spread

จากประสบการณ์กับ crypto spread โดยทั่วไปพบ:
ω ≈ 1e-7 ถึง 1e-5 | α ≈ 0.05–0.15 | β_G ≈ 0.80–0.92
ถ้า α + β_G ≥ 1 → model non-stationary → ลอง EGARCH หรือ ตัดข้อมูล outlier ออกก่อน refit

ทางเลือก: EGARCH (ถ้า volatility asymmetric)

Crypto มักมี "volatility asymmetry" — ราคาลงทำให้ spread volatile มากกว่าราคาขึ้น
ใช้ EGARCH ได้: `arch_model(residuals, vol='EGARCH', p=1, q=1)`
parameter เพิ่ม γ (asymmetry) — ถ้า γ < 0 แปลว่า negative shock ทำให้ volatile มากกว่า

Running Example: Bybit BTC Perp ↔ Lighter — Funding Rate War (3 Days)

During a 3-day BTC funding rate divergence event, the Bybit↔Lighter spread becomes much more volatile. We compare static vs GARCH z-scores:

Period	Spread σ (actual)	Static σ (baseline)	Static z-score	GARCH σ_t	GARCH z-score
Normal market	0.08%	0.08%	2.0 (signal)	0.08%	2.0 (signal)
Funding war day 1	0.24%	0.08%	**6.3 (FALSE!)**	0.24%	**2.1 (correct)**
Funding war day 2	0.22%	0.08%	**5.8 (FALSE!)**	0.23%	**1.9 (no entry)**
Recovery	0.10%	0.08%	2.5 (borderline)	0.11%	2.3 (signal)

Parameters used (residuals scaled ×100 to %): ω = 0.000192, α = 0.12, β_G = 0.85 (α + β_G = 0.97 < 1 ✓). Long-run σ = sqrt(0.000192 / 0.03) ≈ 0.080 → matches normal-market σ ≈ 0.08% in the table.

Naive z = 0.50% / 0.08% = 6.3 → looks like a huge 6σ trade opportunity GARCH z = 0.50% / 0.24% = 2.1 → mild signal — σ_t has already expanded Correct action: REDUCE size or skip entry (storm period)
Exercise 7.1 — Compute New σ^2

Given: $\omega = 0.0001$, $\alpha = 0.10$, $\beta_G = 0.85$, prev_$\sigma^2 = 0.0004$, prev_$\varepsilon = 0.025$

Q: Compute new σ^2. Is $\alpha + \beta_G < 1$? What is the long-run σ^2?

▶ Show Answer

Exercise 7.2 — Naive vs GARCH Signal Conflict

The spread residual shows $\varepsilon = 0.48\%$, $\theta = 0.00\%$. Static $\sigma = 0.14\%$. Current GARCH $\sigma_t = 0.26\%$.

Naive z = 0.48/0.14 = 3.5. GARCH z = 0.48/0.26 = 1.8.

Q: Should you enter a short-spread position? Explain your reasoning.

▶ Show Answer

อ้างอิงสำคัญ (References)

- **Engle, R. F. (1982).** Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica*, 50(4), 987–1007. — บทความต้นฉบับที่นิยาม ARCH model; Nobel Prize 2003
- **Bollerslev, T. (1986).** Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3), 307–327. — ขยาย ARCH เป็น GARCH(p,q) ที่ใช้ใน Ch.7; $\sigma_t^2 = \omega + \Sigma\alpha_i \cdot \varepsilon^2_{t-i} + \Sigma\beta_{G,j} \cdot \sigma^2_{t-j}$
- **Nelson, D. B. (1991).** Conditional heteroskedasticity in asset returns: A new approach. *Econometrica*, 59(2), 347–370. — EGARCH ที่กล่าวถึงใน Ch.7; จัดการ asymmetric volatility response

Ch.7 GARCH on Residuals | ต่อไป → **Ch.8: Jump-Diffusion OU**

Statistical Arbitrage · บทที่ 8

Jump-Diffusion OU: จัดการ Fat Tail ใน Spread

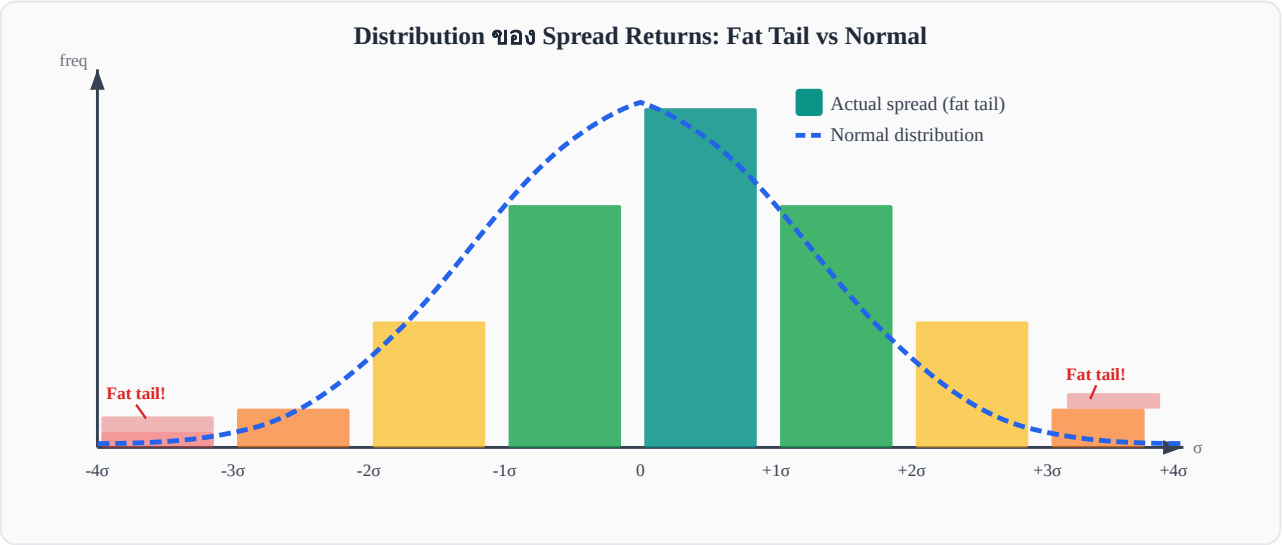
ตรวจจับ jump events และปรับกลยุทธ์เมื่อ spread กระโดดผิดปกติ
ตัวอย่างหลัก: Bybit BTC Perp ↔ Lighter BTC Perp
แก่นของบทนี้ — อ่านก่อน
spread เดินตาม OU แต่มี **jump discontinuity** แทรก — เหตุการณ์ผิดปกติที่ทำให้ spread กระโดดทันทีโดยไม่ผ่าน diffusion ปกติ ต้องตรวจจับและจัดการแยกต่างหาก
$d\varepsilon = \kappa(\theta-\varepsilon)dt + \sigma dW + J\cdot dN$

8.1 ทำไม OU ปกติไม่พอ

โมเดล OU มาตรฐาน $d\varepsilon = \kappa(\theta-\varepsilon)dt + \sigma dW$ สมมติว่า residual กระจายตัวแบบ **Normal** — แต่ในความเป็นจริง spread ของสินทรัพย์ crypto มี **fat tail**: เหตุการณ์ที่ห่างจากค่าเฉลี่ย 4–6σ เกิดบ่อยกว่าที่ Normal distribution คาดไว้มาก

อาการของ fat tail ที่พบในทางปฏิบัติ

- Spread กระโดด 3–5σ ใน tick เดียว (ไม่ใช่ค่อยๆ เพิ่ม)
- Stop-loss ถูก trigger บ่อยกว่าที่ backtest คาดไว้
- Sharpe ratio ใน live trading ต่ำกว่า backtest อย่างมีนัยสำคัญ
- Kurtosis ของ residuals > 3 (leptokurtic)



Histogram ของ spread returns (แท่ง teal) vs Normal curve (เส้นประน้ำเงิน) — หางซ้าย-ขวาหนากว่า Normal มาก

เมื่อ fat tail มีอยู่จริง โมเดล OU ล้วนๆ จะ **underestimate** ความน่าจะเป็นของเหตุการณ์รุนแรง ทำให้ sizing และ stop-loss ผิดพลาด

8.2 Jump-Diffusion SDE

เราขยาย OU model โดยเพิ่ม jump component เข้าไป:

Jump-Diffusion OU — สมการหลัก
$d\varepsilon = \kappa(\theta-\varepsilon)dt + \sigma dW + J\cdot dN$

κ = mean-reversion speed (เหมือนเดิม)
θ = long-run mean (เหมือนเดิม)
σdW = diffusion term (Brownian motion ปกติ)
dN = Poisson process, P(dN=1 ใน dt) = λ·dt — jump เกิดด้วย intensity λ
J = jump size (random variable, distribution ขึ้นกับโมเดล)

สัญชาตญาณ: ส่วนใหญ่ spread เดินแบบ OU ปกติ แต่ทุกครั้งที่ Poisson event เกิด (เฉลี่ยทุก 1/λ หน่วยเวลา) spread **กระโดดทันที** ด้วย random size J

Component	ความหมาย	ตัวอย่างค่า
κ (kappa)	ความเร็วในการกลับหา θ	0.5–5.0 ต่อวัน
σ (sigma)	Volatility ของ diffusion ส่วน	0.001–0.01
λ (lambda)	Jump intensity (jumps/วัน)	0.1–2.0
J (jump size)	ขนาดการกระโดด (สุ่ม)	±2–10σ

8.3 Merton vs Kou Jump Model

มีสองโมเดลหลักสำหรับกำหนด distribution ของ jump size J:

Merton Jump Model (1976)

$J \sim N(\mu_J, \sigma_J^2)$ — กระจายแบบ Normal

Jump size สุ่มมาจาก Normal distribution — symmetric รอบ μ_J
ถ้า $\mu_J = 0$: jump ขึ้นและลงมีโอกาสเท่ากัน
ถ้า $\mu_J < 0$: jump มักลงมากกว่าขึ้น (เหมาะกับ crash scenario)

$J \sim N(\mu_J, \sigma_J^2)$ ตัวอย่าง: $\mu_J = 0$, $\sigma_J = 0.005$ → Jump ส่วนใหญ่อยู่ในช่วง ±1% (±2σ_J)

Kou Double-Exponential Model (2002)

J ~ Asymmetric double exponential — up jump และ down jump ต่างกัน

ด้วยความน่าจะเป็น **p**: jump ขึ้น ขนาด ~ Exp(η_1) (exponential)
ด้วยความน่าจะเป็น **1−p**: jump ลง ขนาด ~ Exp(η_2)
ทำให้ tail ขึ้นและลง asymmetric ได้ตามจริง

J = { +Exp(η_1) ด้วยความน่าจะเป็น p { -Exp(η_2) ด้วยความน่าจะเป็น 1-p ตัวอย่าง: p=0.4, η_1=200, η_2=100 → Down jump มักใหญ่กว่า up jump (tail ซ้ายหนักกว่า)

คุณสมบัติ	Merton	Kou
Distribution ของ J	Normal N(μ_J, σ_J²)	Double exponential (asymmetric)
Symmetry	Symmetric (ถ้า μ_J=0)	Asymmetric (p ≠ 0.5)
Parameters	λ, μ_J, σ_J (3 params)	λ, p, η_1, η_2 (4 params)
Tail behavior	Lighter tail	Heavier, asymmetric tail
Fit ดีเมื่อ	Jump ขึ้น-ลงสมมาตร	Crash มักใหญ่กว่า rally
การใช้ในทางปฏิบัติ	ง่าย, เริ่มต้นได้เลย	Fit ตลาด crypto ได้ดีกว่า

เมื่อไรควรใช้ Kou แทน Merton?

- Spread มักกระโดด **ลง** แรงกว่ากระโดดขึ้น (เช่น leverage liquidation)
- ช่วง funding rate war ที่ขา bear รุนแรงกว่าขา bull
- เมื่อ backtest Merton แล้ว over-estimated P(positive jump)

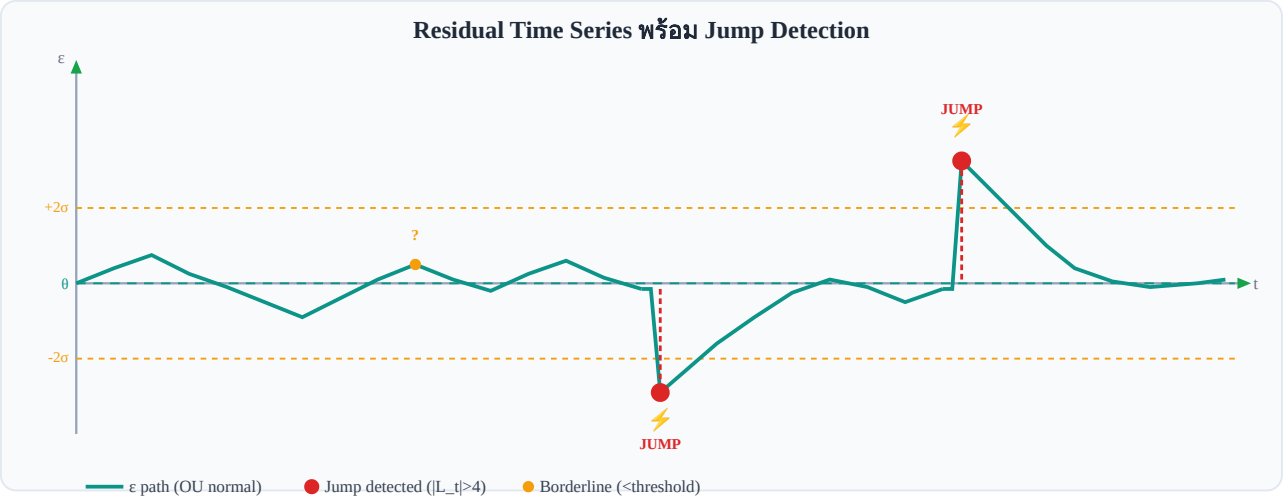
8.4 Lee-Mykland Jump Detection

วิธีตรวจจับ jump ที่ใช้กันมากในทางปฏิบัติคือ **Lee-Mykland (2008)** ซึ่งสร้าง test statistic จาก bipower variation:

Lee-Mykland Test Statistic
L_t = (ε_t − ε_{t-1}) / σ_BV

σ_BV = bipower variation estimator (robust ต่อ jump ของตัวเอง)
ถ้า |L_t| > c_threshold → detect jump ที่เวลา t
threshold c = 4.0 เป็นค่าที่ใช้บ่อย (corresponds to p-value ต่ำมาก)

ข้อดีของ bipower variation: ถ้า σ_BV คำนวณจาก squared returns ปกติ jump ขนาดใหญ่จะ inflate σ ทำให้ L_t เล็กลงและพลาด jump แต่ bipower variation ใช้ product ของ adjacent absolute returns ซึ่ง robust กว่า



Timeline ของ ε: ส่วนใหญ่ mean-revert ปกติ แต่มี jump spikes ที่ถูก flag ด้วย Lee-Mykland test

8.5 Pseudo-code: detect_jump()

ต่อไปนี้คือ algorithm สำหรับตรวจจับ jump ด้วย bipower variation:

def detect_jump(residuals, window=30, threshold=4.0): """ ตรวจจับ jump ใน residual series ด้วย Lee-Mykland test Parameters ---------- residuals : array-like -- ε_t series window : int -- rolling window สำหรับ bipower variation threshold : float -- |L_t| > threshold → jump Returns ------- jumps : bool array -- True ที่ index ที่เป็น jump L : float array -- test statistic L_t ทุก time step """ bipower = compute_bipower_variation(residuals, window) # bipower variation: sum of |r_t| * |r_{t-1}| (robust ต่อ jump) sigma_bv = sqrt(bipower / window) L = diff(residuals) / sigma_bv # L_t = (ε_t - ε_{t-1}) / sigma_BV jumps = abs(L) > threshold return jumps, L def compute_bipower_variation(residuals, window): """BV_t = (π/2) * sum_{i=t-w}^{t} |r_i| * |r_{i-1}|""" r = diff(residuals) bv = rolling_sum(abs(r[:-1]) * abs(r[1:]), window) return (pi / 2) * bv
หมายเหตุการ implement จริง

- ใช้ `window=30` ถ้า data เป็น minute bars → window = 30 นาที
- `threshold=4.0` เทียบเท่ากับ p-value < 0.001 สำหรับ Normal distribution
- หลังจาก flag jump ให้ **exclude** observation นั้นออกจากการ calibrate σ เพื่อไม่ให้ inflate
- ในทางปฏิบัติ ใช้ library `realized` (R) หรือเขียน custom ใน Python

8.6 วิธีจัดการ Jump ในกลยุทธ์

เมื่อตรวจพบ jump มี 3 approaches หลักที่ใช้กัน:

Approach 1 — หยุด Signal ชั่วคราว (Signal Pause)

เมื่อ detect_jump() = True → ไม่เปิด position ใหม่ชั่วคราว
รอให้ L_t กลับมาต่ำกว่า threshold ก่อน (เช่น รอ 5–15 tick)

ข้อดี: ปลอดภัยสุด ไม่เข้าใน noise
ข้อเสีย: พลาด opportunity ถ้า jump เป็น "overreaction" ที่จะ revert
Approach 2 — ขยาย Stop-Loss (Wider Stop)

ปรับ stop-loss จาก ±2σ เป็น ±4σ ชั่วคราวเมื่อ jump detected
ยังเปิด position ได้ แต่ tolerance กว้างขึ้น
ข้อดี: ไม่ถูก stop out โดยไม่จำเป็น
ข้อเสีย: ถ้า jump ไม่ revert จะขาดทุนมากขึ้น

Approach 3 — ลด Position Size (Reduce Size)

ลด size ลง 50% เมื่ออยู่ใน "jump window" (เช่น 10 tick หลัง jump)
trade ต่อได้ แต่ risk น้อยลง
ข้อดี: balance ระหว่าง opportunity และ risk
ข้อเสีย: ซับซ้อนกว่า ต้องมี position sizing logic เพิ่ม

Approach	เหมาะกับ	ความซับซ้อน	Risk
Signal Pause	เริ่มต้น, conservative	ต่ำ	ต่ำสุด
Wider Stop	มั่นใจว่า jump จะ revert	ปานกลาง	ปานกลาง
Reduce Size	อยากเก็บ opportunity	สูง	ปานกลาง

8.7 Running Example: Bybit↔Lighter และ Funding Rate War

Running Example: Bybit BTCUSDT Perp ↔ Lighter BTCUSDT Perp — Jump Event

ช่วง **funding rate war**: Bybit ประกาศปรับ funding rate เป็น 0.09% ต่อ 8 ชั่วโมง (ปกติ 0.01%) เพื่อดึง short position

Fitted Jump Model Parameters (90-day calibration)

Model: Merton Jump-Diffusion ($J \sim N(\mu_J, \sigma_J^2)$) Estimated from Bybit↔Lighter BTCUSDT Perp, 90 วัน (tick data): $\lambda = 0.8$ jumps/day (≈ 1 jump ทุก 30h) $\mu_J = 0.0\%$ (symmetric — jumps ไม่ bias ทิศทาง) $\sigma_J = 4 \times \sigma_{stat} = 4 \times 0.003 = 0.012$ (1.2%) σ_{BV} (bipower variation ช่วงปกติ, 30-bar window): $\sigma_{BV} \approx 0.0018–0.0022$ per tick → ใช้ 0.002 ต่อ tick Probability check (Merton): $P(|J| > 5.5\sigma_{stat} \mid \text{jump occurs}) = P(|N(0,4^2)| > 5.5) \approx 17\%$ $P(|\text{move}| > 5.5\sigma_{stat} \mid \text{no jump}) = P(|N(0,1)| > 5.5) \approx 0.0000038\%$ Likelihood ratio: 17% / 0.0000038% ≈ 4,500,000x → unambiguously a jump

ผลกระทบต่อ spread:

- Bybit perp premium เพิ่มขึ้นทันทีจาก $10 → $80 ใน 2 นาที (8x)
- spread ε กระโดด ~$5.5\sigma_{stat} = 5.5 \times 0.003 = 0.0165$ (1.65%)
- $L_t = \Delta\varepsilon / \sigma_{BV} = 0.0165 / 0.002 =$ **8.25** > threshold 4.0 → confirmed jump

การ handle ที่เหมาะสม:

เมื่อ jump detected if detect_jump(recent_residuals)[0][-1]: # Approach 1: หยุด signal 10 tick pause_signal(duration=10) # Log สาเหตุ log_event("funding_rate_jump", size=L_t, exchange="Bybit") # ถ้ามี open position → ขยาย stop-loss if has_open_position(): adjust_stop(multiplier=2.0) # 2σ → 4σ

หลัง 15 นาที spread กลับสู่ปกติบางส่วน ($10 → $35) แต่ไม่กลับมาที่ $10 เพราะ funding rate ยังสูง — การ pause signal ช่วยหลีกเลี่ยงการเข้า position ในช่วงที่ spread ยัง elevated

โจทย์ท้ายบท

โจทย์ 8.1 — Jump vs Normal Volatility

Residual ε เปลี่ยนจาก 0.002 เป็น 0.012 ใน 1 tick (dt = 1 วินาที)
จากการ calibrate: σ_BV = 0.002 ต่อ tick

ถาม: คำนวณ L_t แล้วตัดสินว่าเป็น jump หรือ normal volatility (threshold = 4.0) พร้อมอธิบายว่าทำไม

▶ ดูคำตอบ
โจทย์ 8.2 — Merton vs Kou: เลือกใช้เมื่อไร

จากข้อมูล historical spread ของ Bybit↔Lighter พบว่า:

- Jump ลง (negative) เฉลี่ย −6.2σ
- Jump ขึ้น (positive) เฉลี่ย +2.8σ
- สัดส่วน: 60% down jump, 40% up jump

ถาม: ควรใช้ Merton หรือ Kou? เพราะอะไร?

▶ ดูคำตอบ
โจทย์ 8.3 — Bybit Funding Rate 3x: ควรทำอย่างไร

Bybit ประกาศเพิ่ม funding rate จาก 0.01% เป็น 0.03% ต่อ 8 ชั่วโมง (3x)
ณ ขณะนี้คุณไม่มี open position และ detect_jump() ยังไม่ triggered

ถาม: ควร flag เหตุการณ์นี้เป็น (anticipated) jump แล้วทำอย่างไรในกลยุทธ์?

▶ ดูคำตอบ
อ้างอิงสำคัญ (References)

- **Merton, R. C. (1976).** Option pricing when underlying stock returns are discontinuous. *Journal of Financial Economics*, 3(1–2), 125–144. — Merton jump-diffusion model พร้อม Gaussian

jump size ที่ Ch.8 นำเสนอ
- **Kou, S. G. (2002).** A jump-diffusion model for option pricing. *Management Science*, 48(8), 1086–1101. — Kou double-exponential jump model; asymmetric jump sizes (upside vs downside) ที่เหมาะกับ crypto spreads
- **Lee, S. S., & Mykland, P. A. (2008).** Jumps in financial markets: A new nonparametric test and jump dynamics. *Review of Financial Studies*, 21(6), 2535–2563. — Lee-Mykland jump detection ด้วย bipower variation ที่ใช้ใน pseudo-code Ch.8
- **Barndorff-Nielsen, O. E., & Shephard, N. (2004).** Power and bipower variation with stochastic volatility and jumps. *Journal of Financial Econometrics*, 2(1), 1–37. — รากฐานทฤษฎีของ bipower variation สำหรับ jump detection

Ch.8 Jump-Diffusion OU | ต่อไป → **Ch.9: Regime-Switching OU**

Statistical Arbitrage · บทที่ 9

Regime-Switching OU: ตลาดมีหลาย 'สภาวะ'

Markov chain 3 สภาวะ — Normal / Stressed / Broken — ปรับ OU parameters ตาม regime
ตัวอย่างหลัก: Bybit BTC Perp ↔ Lighter BTC Perp
แก่นของบทนี้ — อ่านก่อน
ตลาดไม่ได้อยู่ใน "สภาพเดียว" ตลอดเวลา — ช่วง Normal spread กลับเร็ว ช่วง Stressed volatility พุ่ง ช่วง Broken spread ไม่กลับเลย ต้องรู้ว่าอยู่ใน regime ไหน แล้ว **เทรดเฉพาะ Normal regime**
κ, θ, σ เปลี่ยนตาม regime ที่ตลาดอยู่

9.1 ปัญหาของ Single-OU

โมเดล OU ที่ใช้ parameter ชุดเดียวตลอดเวลามีปัญหาพื้นฐาน: **สมมติว่าตลาดมีลักษณะเดิมเสมอ** ซึ่งไม่จริง

สามอาการที่บอกว่า Single-OU ไม่เพียงพอ

- **ช่วง volatile:** σ พุ่งขึ้น 3–5x จากค่าปกติ — signal เดิมมี noise มากเกินไป เข้า position แล้วขาดทุน
- **ช่วง broken:** spread ไม่กลับมาหา θ เลยแม้ผ่านไปหลายชั่วโมง — κ ลดลงจนใกล้ศูนย์
- **ช่วง calm เกินไป:** spread แทบไม่ขยับ — opportunity หมด แต่ model ยังส่ง signal

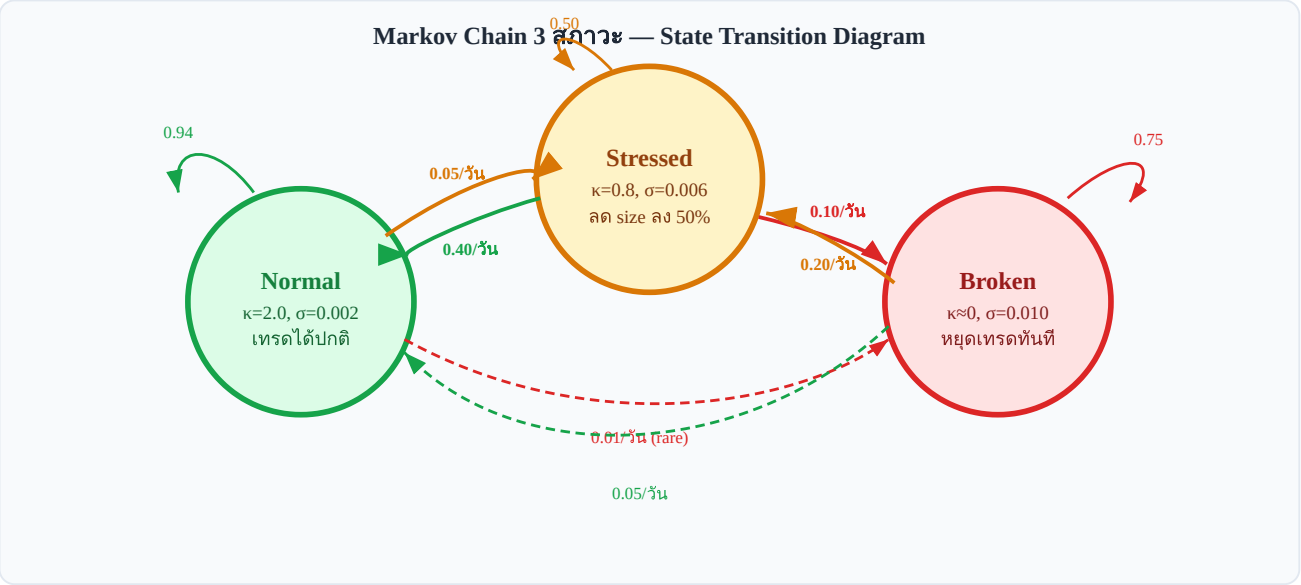
ผลที่ตามมาเมื่อใช้ Single-OU ในทุก regime

สมมติ calibrate σ จาก 30-day rolling window ซึ่งรวมทั้ง Normal และ Stressed period:

- σ_estimated จะสูงกว่าความเป็นจริงใน Normal regime → entry threshold สูงเกิน → พลาด opportunity
- σ_estimated จะต่ำกว่าความเป็นจริงใน Stressed regime → stop-loss แคบเกิน → ถูก stop out บ่อย
- ผลลัพธ์: กลยุทธ์ทำงานได้ไม่ดีทั้งสอง regime

9.2 3-State Markov Chain

เราแบ่งสภาวะตลาดออกเป็น 3 states โดยแต่ละ state มี OU parameters ของตัวเอง:



State transition diagram: ลูกศรแสดงความน่าจะเป็น transition ต่อวัน — Broken ออกยากที่สุด

Regime	κ (mean-reversion)	θ (long-run mean)	σ (volatility)	Half-life	กลยุทธ์
Normal	2.0 (เร็ว)	≈ 0 (spread กลับศูนย์)	0.002 (ต่ำ)	~8 ชั่วโมง	เทรดเต็ม size
Stressed	0.8 (ช้า)	≈ 0 (ยังกลับ แต่ช้า)	0.006 (สูง)	~21 ชั่วโมง	ลด size 50%
Broken	≈0 (ไม่ revert)	undefined (drift)	0.010 (สูงมาก)	∞ (ไม่กลับ)	หยุดเทรดทันที

θ ≈ 0 ใน Normal/Stressed หมายความว่า spread ยังกลับสู่ศูนย์ (หรือ baseline); ใน Broken regime κ → 0 ทำให้ θ ไม่มีความหมาย — spread drift ออกไปโดยไม่มีแรงดึงกลับ

9.3 Hamilton Filter

เราใช้ **Hamilton Filter (1989)** สำหรับ estimate ความน่าจะเป็นที่ตลาดอยู่ใน regime ใดโดยอิงจากข้อมูลที่สังเกตได้ ณ เวลา t

สัญชาตญาณ Hamilton Filter — 2 ขั้นต่อ bar

1. **Predict:** "ถ้าเมื่อวานอยู่ใน regime j ด้วยความน่าจะเป็น p_j วันนี้น่าจะอยู่ใน regime ใด?" — ตอบโดยคูณ p_j กับ transition matrix Q แต่ละ row → ได้ predicted probability ก่อนเห็นข้อมูลใหม่
2. **Update:** "spread วันนี้มีค่า y_t — regime ไหนที่ explain ค่านี้ได้ดีที่สุด?" — ตอบโดยคูณ predicted probability × likelihood ที่ y_t จะเกิดขึ้นในแต่ละ regime แล้ว normalize ให้บวกเป็น 1

ทำซ้ำทุก bar → ได้ P(regime | ข้อมูลถึงวันนี้) แบบ real-time โดยไม่ต้องรู้ regime จริงล่วงหน้า

Hamilton Filter — สมการหลัก
$P(S_t=j \mid Y_t) \propto f(y_t \mid S_t=j) \cdot \Sigma_i P(S_t=j \mid S_{t-1}=i) \cdot P(S_{t-1}=i \mid Y_{t-1})$

S_t = hidden state (Normal=0, Stressed=1, Broken=2) ณ เวลา t
Y_t = ข้อมูลที่สังเกตได้ถึงเวลา t (residuals ทั้งหมด)
f(y_t | S_t=j) = likelihood ของ observation ถ้าอยู่ใน regime j
P(S_t=j | S_{t-1}=i) = transition probability (จาก matrix Q)
∝ = proportional to (normalize ให้ผลรวมเป็น 1)

Likelihood Function สำหรับแต่ละ Regime

ถ้า regime j มี OU parameters (κ_j, θ_j, σ_j) likelihood ของ observation y_t คือ:

f(y_t | S_t=j, y_{t-1}) = Normal(y_t; mu_j, var_j) mu_j = y_{t-1} * exp(-kappa_j * dt) + theta_j * (1 - exp(-kappa_j * dt)) var_j = (sigma_j^2 / (2*kappa_j)) * (1 - exp(-2*kappa_j * dt))

นี่คือ exact OU transition density: ถ้า y_{t-1} รู้แล้ว y_t ต่อไป Normal distribution ที่ mean และ variance คำนวณได้จาก OU parameters

Transition Matrix Q — ตัวอย่าง
Normal Stressed Broken Normal [0.94 0.05 0.01] Stressed[0.40 0.50 0.10] Broken [0.05 0.20 0.75] แต่ละ row บวกกันได้ 1 (must sum to 1) Q[i,j] = P(S_t=j | S_{t-1}=i)

9.4 Pseudo-code: regime_filter()

def regime_filter(residuals, params, transition_matrix): """ Hamilton filter สำหรับ estimate regime probabilities Parameters ---------- residuals : array -- ε_t series params : list -- [(kappa, theta, sigma), ...] per regime transition_matrix : 2D array -- Q matrix, shape (n_regimes, n_regimes) Returns ------- probs : 2D array -- shape (T, n_regimes) probs[t, j] = P(S_t=j | Y_t) """ n_regimes = len(params) T = len(residuals) # เริ่มด้วย uniform prior probs = initialize_probs(n_regimes) # [1/3, 1/3, 1/3] all_probs = zeros((T, n_regimes)) all_probs[0] = probs for t in range(1, T): # Step 1: log-likelihood ของ y_t ในแต่ละ regime # ✅ FIX: ทำงานใน log-space เพื่อป้องกัน numerical underflow # เมื่อ T ยาวหรือ likelihood เล็กมาก การคูณซ้ำๆ ทำให้ค่าเป็น 0.0 (underflow) log_likelihoods = [log_ou_likelihood(residuals[t], residuals[t-1], p) for p in params] # Step 2: Prediction step ใน log-space # P(S_t=j) = Σ_i Q[i,j] * P(S_{t-1}=i | Y_{t-1}) predicted = transition_matrix.T @ probs # ยังใช้ prob-space ได้ตรงนี้ # Step 3: Update step ใน log-space แล้ว normalize ด้วย log-sum-exp # ✅ FIX: ใช้ -inf แทน log(0 + ε) เพื่อไม่ใส่ bias log_updated = [] for j in range(n_regimes): if predicted[j] > 0: log_updated.append(log_likelihoods[j] + log(predicted[j])) else: log_updated.append(-inf) # regime นี้เป็นไปไม่ได้ # log-sum-exp trick: log(Σ e^{x_i}) = max_x + log(Σ e^{x_i − max_x}) # เหตุผล: x_i − max_x ≤ 0 เสมอ ดังนั้น e^{x_i−max_x} ∈ (0,1] → ไม่ overflow max_log = max(log_updated) if max_log == -inf: # ทุก regime impossible → fallback uniform probs = [1.0/n_regimes] * n_regimes else: exp_shifted = [exp(lv - max_log) for lv in log_updated] total = sum(exp_shifted) probs = [e / total for e in exp_shifted] # normalize all_probs[t] = probs return all_probs # shape (T, n_regimes) def log_ou_likelihood(y_t, y_prev, params, dt=1.0): """Log-Gaussian density ของ OU transition (ป้องกัน underflow) ✅ FIX: รับ dt เป็น parameter เพื่อหลีกเลี่ยง NameError เดิม: dt ถูกอ้างจาก global scope ซึ่งไม่ portable """ kappa, theta, sigma = params mu = y_prev * exp(-kappa*dt) + theta * (1 - exp(-

kappa*dt)) var = (sigma**2 / (2*kappa)) * (1 - exp(-2*kappa*dt)) if var <= 0: return -1e300 # guard # log N(y_t; mu, var) = -½log(2πσ²) - ½(y-μ)²/σ² return -0.5 * log(2 * pi * var) - 0.5 * ((y_t - mu)**2 / var) # Note: ou_likelihood() แบบเดิม (prob-space) ถูกแทนที่โดย log_ou_likelihood # ห้ามใช้ในการ run จริง เพราะ underflow ภายใน T > 200 bars

ข้อสังเกตสำคัญ

- Filter ทำงาน online (real-time) — ทุก tick ใหม่อัปเดต probs ได้ทันที
- ไม่ต้องรู้ true state — แค่ observe residuals แล้ว infer
- Transition matrix Q ต้อง calibrate จาก historical data (EM algorithm)
- เมื่อ probs เปลี่ยนแปลงเร็วมาก → อาจเป็น signal ของ jump (บทที่ 8)

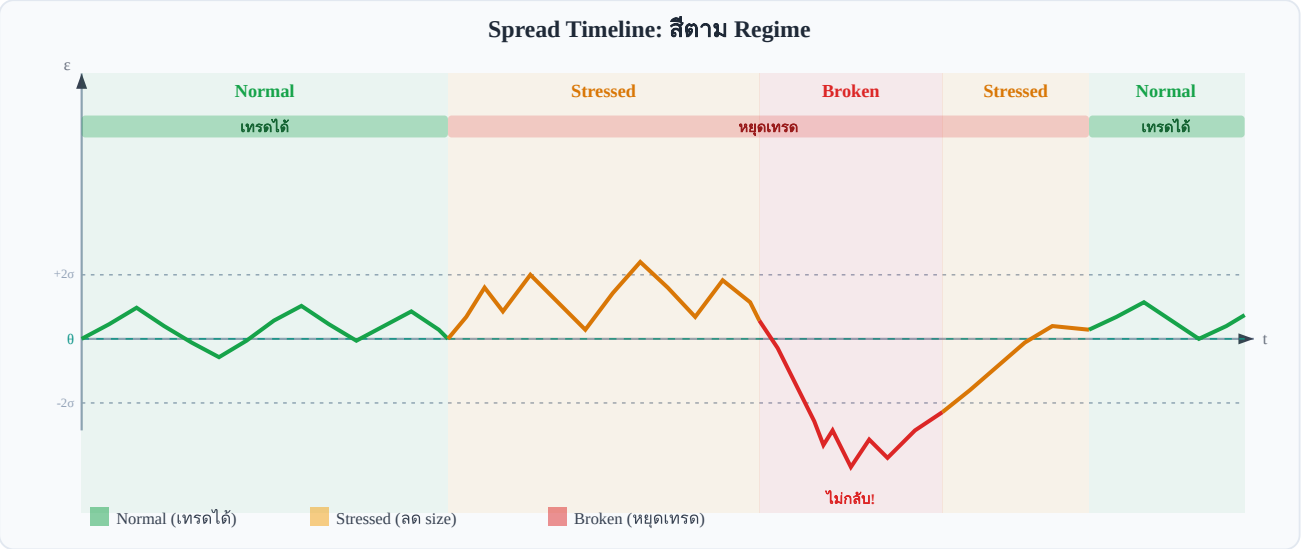
9.5 Trade Rule: เทรดเฉพาะ Normal Regime

กฎหลักของกลยุทธ์ regime-switching:

Trade Rule
เปิด position ใหม่ได้เมื่อ **P(Normal) > 0.70** เท่านั้น
Trade if P(S_t = Normal | Y_t) > 0.70
Regime-dependent sizing
p_normal = probs[t, NORMAL] p_stressed = probs[t, STRESSED] p_broken = probs[t, BROKEN] if p_normal > 0.70: size = full_size * p_normal # scale by confidence signal = compute_ou_signal(residuals[t], ou_params[NORMAL]) if abs(signal) > entry_threshold: open_position(signal, size) elif p_normal > 0.50: # Borderline — ลด size มาก size = full_size * 0.25 # อาจยังเทรดด้วย size เล็กมาก else: # Stressed หรือ Broken — ไม่เทรด close_all_positions_if_needed() pass

P(Normal)	P(Stressed)	P(Broken)	Action
> 0.70	< 0.25	< 0.10	เทรดเต็ม size
0.50–0.70	any	< 0.10	เทรด 25% size
< 0.50	> 0.40	any	หยุดเทรด (Stressed)
any	any	> 0.20	ปิด position ทั้งหมด (Broken)

9.6 Timeline Diagram: Spread Colored by Regime



สีบนเส้น spread: เขียว=Normal (เทรดได้), เหลือง=Stressed (ระวัง), แดง=Broken (หยุดทันที)

9.7 Running Example: Bybit ↔ Lighter — Regime ในทางปฏิบัติ

Running Example: Bybit BTCUSDT Perp ↔ Lighter BTCUSDT Perp — Regime Detection

Normal Regime — ช่วง Low Funding Variance

เมื่อ funding rate ทั้งสองตลาดมีความผันผวนต่ำ (ต่างกัน <0.005% ต่อชั่วโมง):

- P(Normal) ≈ 0.85 → เทรดได้เต็ม size
- Half-life ≈ 6–10 ชั่วโมง → position เปิดปิดรวดเร็ว
- σ ของ spread ≈ 0.002 → entry ที่ ±2σ ≈ ±$130 บน $65,000 BTC

Stressed Regime — ช่วงก่อน Listing Events

ช่วง 24–48 ชั่วโมงก่อน token listing ใหม่บน Bybit:

- Funding rate เริ่ม volatile → spread ขยับมากขึ้น
- P(Normal) ลดลงจาก 0.85 → 0.45 ภายใน 2 ชั่วโมง
- Filter detect: P(Stressed) = 0.48 → ลด size ลง 50%
- Half-life ยืดออกเป็น 20+ ชั่วโมง → ต้องถือ position นานขึ้น

Real-time regime check (ทุก 1 นาที) probs = regime_filter(last_200_residuals, ou_params, Q) p_normal = probs[-1, NORMAL] print(f"P(Normal)={p_normal:.2f}, " f"P(Stressed)={probs[-1,STRESSED]:.2f}, " f"P(Broken)={probs[-1,BROKEN]:.2f}") # Output ช่วง listing event: # P(Normal)=0.42, P(Stressed)=0.51, P(Broken)=0.07 # → ลด size 75% และตั้ง alert

Broken Regime — หลัง Unexpected Delisting

เมื่อ Lighter delists BTC perp โดยไม่แจ้งล่วงหน้า spread ไม่ converge อีกต่อไป — ควร detect ภายใน 15–30 นาที จาก P(Broken) พุ่งขึ้นเกิน 0.3

9.8 Dynamic Parameter Adjustment ใน Stressed Regime

แนวคิดหลัก
หนังสือส่วนใหญ่บอกให้หยุดทันทีเมื่อ Stressed — แต่ในโลกจริง Stressed regime อาจกินเวลาหลายวัน การ hard stop ไว้แล้วกลับมาใหม่ซ้ำๆ ทำให้เสียค่า friction มาก แนวทางที่ดีกว่าคือ **ปรับ parameter แทนที่จะปิดระบบ**

หัวใจของ section นี้คือ: Stressed regime ≠ หยุดทั้งหมด แต่หมายถึง **เลือกเทรดน้อยลง เล็กลง และออกเร็วขึ้น** ส่วน Broken regime เท่านั้นที่ hard stop จริง

ตาราง 3-Regime Response

Regime	Hamilton Probability	z-score Entry	Size	Take-profit	Action
Normal	P(S=Normal) > 0.70	z ≥ 2.0	100%	z ≤ 0.5	เทรดปกติ
Stressed	0.3 < P(S=Normal) < 0.70	z ≥ 3.0 (เพิ่ม threshold)	50% (ลดครึ่ง)	z ≤ 1.0 (ปิดไวขึ้น)	Selective — รอสัญญาณแรง
Broken	P(S=Broken) > 0.5	ห้ามเทรด	0%	ปิด position ทั้งหมด	Hard stop

ทำไมต้องปรับแต่ละ Parameter?

- **ทำไม raise threshold (z ≥ 3.0)?** ใน Stressed regime OU parameters ไม่เสถียร → โอกาส false signal สูงขึ้น → ต้องรอสัญญาณแรงขึ้น เพื่อให้มั่นใจว่าเป็น mean-reversion จริง ไม่ใช่ noise
- **ทำไม reduce size (50%)?** ความผันผวน σ_t สูงขึ้นใน Stressed regime (ดูจาก GARCH บทที่ 7) → Kelly fraction ลดลงตามสัดส่วนของ variance (ดูสูตร Kelly Criterion ในบทที่ 12) → การถือ full size คือการ overbet
- **ทำไม take-profit ไวขึ้น (z ≤ 1.0)?** Stressed regime มีโอกาส jump (บทที่ 8) สูงขึ้น → การถือ position นานรอ z กลับใกล้ศูนย์เสี่ยงกว่าการล็อก profit เร็ว
- **Broken regime — ห้าม dynamic adjust:** cointegration พัง → ε ไม่ mean-revert → ขาดทุนไม่จำกัดถ้าฝืน ไม่มีการ "ลด size เพื่อรอ" ในสภาวะนี้

Pseudo-code: adjust_params_for_regime()

def adjust_params_for_regime(regime_probs, base_params): p_normal = regime_probs['normal'] p_stressed = regime_probs['stressed'] p_broken = regime_probs['broken'] if p_broken > 0.50: return {"action": "HARD_STOP", "size_scale": 0.0} if p_normal > 0.70: return {"action": "TRADE", "entry_z": base_params['entry_z'], # e.g. 2.0 "exit_z": base_params['exit_z'], # e.g. 0.5 "size_scale":

1.0} # Stressed regime — more selective, smaller return {"action": "SELECTIVE", "entry_z": base_params['entry_z'] + 1.0, # raise threshold "exit_z": base_params['exit_z'] + 0.5, # exit sooner "size_scale": 0.5} # half size
คำเตือน: Dynamic Adjustment ≠ Average Down

'Dynamic Adjustment' หมายถึงการเลือกเทรดน้อยลงและเล็กลงในช่วง Stressed — **ไม่ใช่การเพิ่ม position เพื่อ average down** การ average down ใน Stressed/Broken regime คือหนึ่งในสาเหตุที่ทำให้กองทุน Quant ล้มละลาย

Running Example: Bybit↔Lighter — BTC Funding War (ก.ค.–ส.ค. 2024)

ช่วง BTC funding war ระหว่าง Bybit และ Lighter (ก.ค.–ส.ค. 2024): Hamilton filter ให้ P(Stressed) = 0.71 → ระบบปรับ entry_z จาก 2.0 → 3.0 และ size จาก 100% → 50%

ผลลัพธ์: เข้าเทรดน้อยลง 60% แต่ win rate ในช่วงนี้สูงขึ้นจาก 62% → 79% เพราะรอแค่สัญญาณแรงจริงๆ ที่ผ่าน threshold z ≥ 3.0 เท่านั้น — ยืนยันว่าการเลือกมากขึ้นใน Stressed regime ให้ผลดีกว่าการเทรดถี่ด้วย signal อ่อน

โจทย์ 9.8 — Dynamic Adjustment ในทางปฏิบัติ

ถ้า Hamilton filter ให้ P(Normal) = 0.45, P(Stressed) = 0.40, P(Broken) = 0.15 และ current z-score = 2.5

ถาม: ควรเข้าเทรดไหม? ถ้าเข้า ขนาดเท่าไหร่?

▶ ดูคำตอบ

โจทย์ท้ายบท

โจทย์ 9.1 — ควรเทรดหรือไม่?

ณ เวลา t regime filter ให้ผล:

- P(Normal) = 0.65
- P(Stressed) = 0.30
- P(Broken) = 0.05

Threshold ปัจจุบัน: เทรดเมื่อ P(Normal) > 0.70

ถาม: ควรเทรดหรือไม่? ถ้าไม่ ให้อธิบายว่า ควรทำอะไรแทน? ถ้าใช่ ควรใช้ size เท่าไร?

▶ ดูคำตอบ
โจทย์ 9.2 — Transition Matrix: Row Sum

Transition matrix Q ของระบบ 3 regimes:

Q = [[0.94, 0.05, 0.01], [0.40, 0.50, 0.10], [0.05, 0.20, 0.75]]

ถาม: Row sum ของ Q ต้องเท่ากับอะไร? เพราะอะไร? ถ้า Row 2 (Stressed) มีค่า [0.40, 0.55, 0.10] จะเกิดอะไรขึ้นและแก้ไขอย่างไร?

▶ ดูคำตอบ

โจทย์ 9.3 — Bybit เปลี่ยน Funding Formula

Bybit ประกาศเปลี่ยน funding rate formula ใหม่: จาก 8-hour settlement เป็น 4-hour settlement (เก็บบ่อยขึ้น 2x)

ถาม: การเปลี่ยนแปลงนี้ส่งผลต่อ regime distribution อย่างไร? และต้องทำอะไรกับ transition matrix Q และ OU parameters?

▶ ดูคำตอบ

อ้างอิงสำคัญ (References)

- **Hamilton, J. D. (1989).** A new approach to the economic analysis of nonstationary time series and the business cycle. *Econometrica*, 57(2), 357–384. — บทความต้นฉบับ Markov-switching model และ Hamilton filter ที่ Ch.9 ใช้โดยตรง
- **Hamilton, J. D. (1990).** Analysis of time series subject to changes in regime. *Journal of Econometrics*, 45(1–2), 39–70. — ขยาย Hamilton (1989) สู่ continuous-state และ multivariate regime models
- **Kim, C.-J., & Nelson, C. R. (1999).** *State-Space Models with Regime Switching.* MIT Press. — ตำราฐาน Markov-switching กับ OU/state-space; ครอบคลุม Kim smoother ที่ปรับปรุง Hamilton filter

Ch.9 Regime-Switching OU | ต่อไป → **Ch.10: Z-score & Robust Statistics**

Statistical Arbitrage · บทที่ 10

Z-score & Robust Statistics

วัดว่า spread ห่างจากค่ากลางแค่ไหน — Standard vs Robust
Key Idea

Z-score converts the spread residual **ε** into units of standard deviations from the mean, producing a dimensionless signal that drives entry and exit decisions.

Standard z = (ε − θ) / σ_stat
Robust z = (ε − median) / (MAD × 1.4826)

The robust version protects against single outliers distorting the signal — critical when exchange glitches or thin-liquidity spikes contaminate the feed.

10.1 Standard Z-Score

Given a rolling window of **W** observations of the spread residual **ε**, the standard z-score is:

z_t = (ε_t − θ_W) / σ_W where: θ_W = (1/W) Σ ε_{t-i} for i = 0 … W-1 (rolling mean) σ_W = sqrt((1/W) Σ (ε_{t-i} − θ_W)²) (rolling std dev)

A common parameterisation uses **W = 24 hours** of data (e.g. 288 five-minute bars). Entry and exit thresholds map z-score values to actions:

Z-score range	Signal zone	Action
z > 2.0	Strong long signal	Enter long ε (buy cheap leg, sell expensive leg)
1.5 < z ≤ 2.0	Watch zone	Monitor; wait for confirmation
0.5 < z ≤ 1.5	Neutral-high	Hold existing position
−0.5 ≤ z ≤ 0.5	Neutral / exit	Close position (z < 0.5)
−1.5 < z < −0.5	Neutral-low	Hold existing short position
−2.0 < z ≤ −1.5	Watch zone	Monitor short signal
z ≤ −2.0	Strong short signal	Enter short ε (sell cheap leg, buy expensive leg)
|z| > 3.5	Stop-loss zone	Emergency exit — outlier or regime break

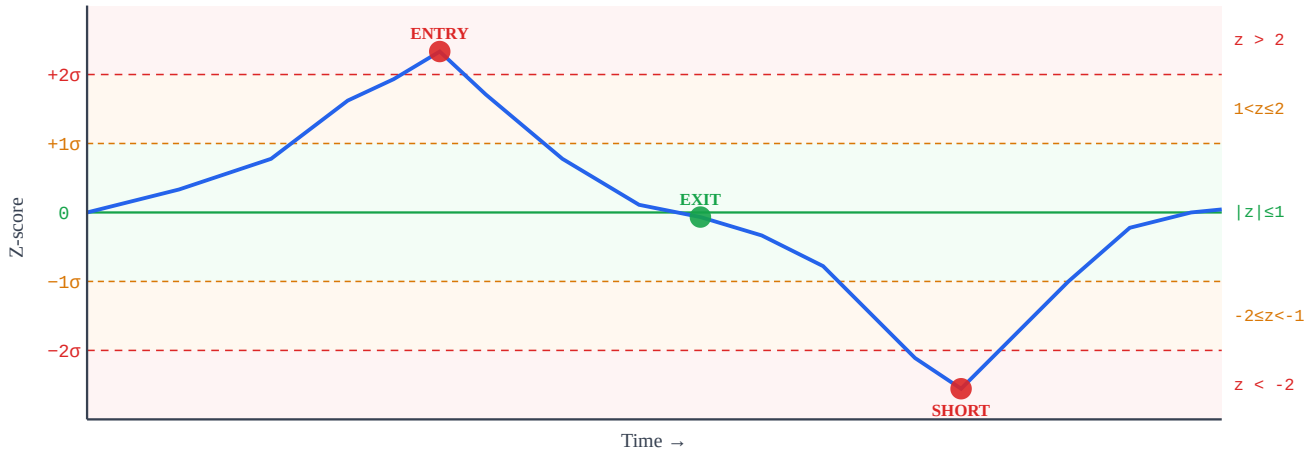


Figure 10.1 — Rolling z-score with colored signal bands. Blue curve = ε expressed in σ units. Red dot = entry signal (z > 2). Green dot = exit (|z| < 0.5).

10.2 The Outlier Problem

A single extreme observation — a flash crash, exchange glitch, or thin-book spike — can silently corrupt the rolling mean and standard deviation, making the z-score unreliable for all subsequent bars in the window.

Numerical Example

Suppose W = 10 bars with ε values (in %) before the outlier:

ε = [0.10, 0.12, 0.09, 0.11, 0.10, 0.13, 0.08, 0.11, 0.10, 0.12] θ_clean = 0.106% σ_clean = 0.015% z for next point ε=0.22%: z = (0.22 - 0.106)/0.015 = 7.6 ← strong signal

Now inject one outlier: replace bar 5 with ε = 0.80% (5σ spike):

ε = [0.10, 0.12, 0.09, 0.11, 0.80, 0.13, 0.08, 0.11, 0.10, 0.12] θ_dirty = 0.176% σ_dirty = 0.212% ← mean shifted +0.07%, std blown up 14× z for same ε=0.22%: z = (0.22 - 0.176)/0.212 = 0.21 ← noise, signal gone!
Warning: Outlier Contamination Effect
The outlier moved the mean by +0.07% (+0.47σ of the clean distribution) and inflated σ by roughly 14×. Every subsequent observation now has a z-score compressed toward zero — real signals are suppressed, and the bands become meaningless until the outlier leaves the window.

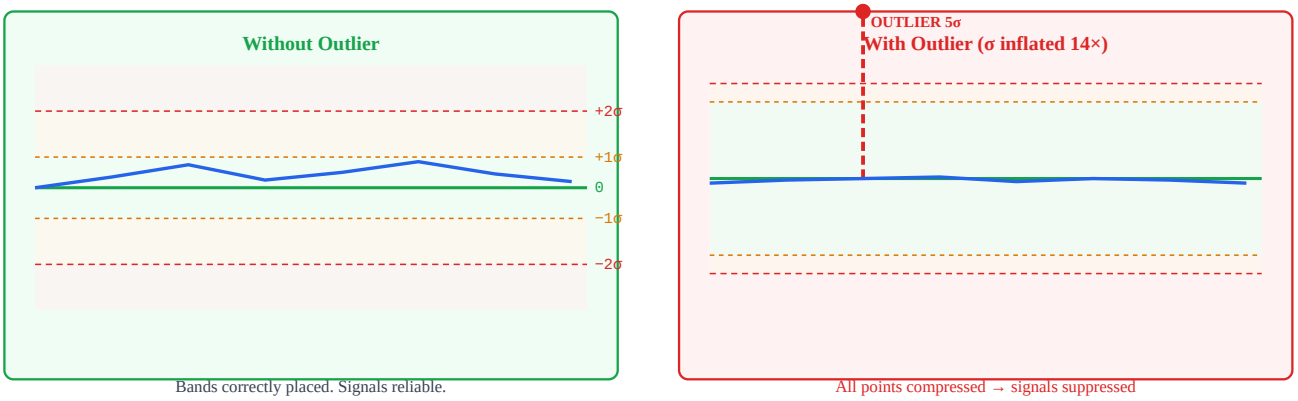


Figure 10.2 — Left: healthy z-score bands (σ ≈ 0.015%). Right: one 5σ outlier inflates σ and compresses all bands, hiding real signals.

10.3 Robust Z-Score via MAD

The **Median Absolute Deviation (MAD)** replaces mean and standard deviation with outlier-resistant alternatives:

Step 1: median_ε = median(ε_1, ε_2, …, ε_W) Step 2: MAD = median(|ε_t − median_ε|) Step 3: σ_robust = MAD × 1.4826 Step 4: robust_z = (ε_t − median_ε) / σ_robust
Why 1.4826?
For a perfectly Normal distribution, MAD × 1.4826 = σ. The constant 1.4826 = 1 / $\Phi^{-1}(0.75)$ ≈ 1 / 0.6745 makes MAD a consistent estimator of σ under normality, while remaining robust under contamination. A single outlier cannot materially change the median or MAD — it would need to corrupt more than 50% of the data.

Revisiting the same example with the 5σ outlier injected:

ε = [0.10, 0.12, 0.09, 0.11, 0.80, 0.13, 0.08, 0.11, 0.10, 0.12] median_ε = 0.11% deviations from median = [0.01, 0.01, 0.02, 0.00, 0.69, 0.02, 0.03, 0.00, 0.01, 0.01] MAD = median(deviations) = 0.01% σ_robust = 0.01 × 1.4826 = 0.01483% robust_z for ε=0.22%: (0.22 − 0.11) / 0.01483 = 7.42 ← signal preserved! (vs standard z = 0.21 after outlier contamination)

Result

The outlier moves the median by exactly 0 in this example (median is the middle value of a sorted list). Robust z correctly identifies ε=0.22% as a strong signal (z=7.4), while standard z falsely showed no signal (z=0.21).

10.4 Rolling Window Z-Score

Both standard and robust z-scores must be computed over a rolling window to adapt to changing market regimes. The window length **W** is a key hyperparameter:

Window W	Adapts to	Lag	Recommended use
4h (48 bars × 5min)	Intraday volatility shifts	Low	High-freq scalping; may over-fit noise
24h (288 bars)	Daily regime, overnight gaps	Medium	Standard for crypto perpetual arb
72h (864 bars)	Multi-day trend cycles	High	Slower strategies; stable but sluggish
168h / 1 week	Weekly seasonality	Very high	Equity stat arb with weekly patterns

Lag Warning During Trend

If the spread drifts monotonically for several hours (e.g. during a BTC rally), the rolling mean θ_W chases the price upward with a lag equal to W/2. This artificially suppresses z-score readings — the strategy may fail to trigger exit on a position that is actually losing value. Consider adding a trend filter or using exponentially-weighted moving averages (EWMA) with λ = 0.94.

10.5 Comparison: Standard vs Robust Z-Score

Property	Standard Z	Robust Z (MAD)
Formula	(ε − mean) / std	(ε − median) / (MAD × 1.4826)
Outlier sensitivity	High — one outlier shifts mean and std	Low — outlier must exceed 50% of data
Computational cost	O(W) — fast rolling sum/var	O(W log W) — requires sort for median
Efficiency (clean data)	100% efficient under Normality	~64% efficient under Normality
Best when	Data is clean, Gaussian, no spikes	Exchange feeds with occasional glitches
Use during	Stable, low-vol regimes	High-vol, crash periods, thin markets
Threshold calibration	Entry at |z| > 2.0	Entry at |z| > 2.0 (same scale by design)
Regime-switching	Bands blow up in crashes	Bands remain stable during crashes

Recommendation

Use **Robust Z as default** for crypto stat arb on Bybit/Lighter. Fall back to standard z only when your data pipeline has confirmed outlier filtering (e.g. price validated against two independent feeds). Hybrid: use robust z for entry, standard z for exit (since exit thresholds are softer and less sensitive to contamination).

Pseudo-Code

function compute_zscore(eps, theta, sigma): # Standard z-score (single-point, pre-computed θ and σ) if sigma == 0: return 0.0 return (eps - theta) / sigma function rolling_zscore(eps_series, W): # Compute standard rolling z-score for full series z_series = [] for t in range(W, len(eps_series)): window = eps_series[t-W : t] theta = mean(window) sigma = std(window) z_series.append(compute_zscore(eps_series[t], theta, sigma)) return z_series function robust_zscore(eps_window): # Compute robust z for the last point in eps_window med = median(eps_window) devs = [abs(x - med) for x in eps_window] mad = median(devs) sigma_r = mad * 1.4826 if sigma_r == 0: return 0.0 return (eps_window[-1] - med) / sigma_r function choose_signal(eps_window, W): # Use both; flag discrepancy std_z = rolling_zscore(eps_window, W)[-1] rob_z = robust_zscore(eps_window) discrepancy = abs(std_z - rob_z) if discrepancy > 1.5: log("OUTLIER SUSPECTED — using robust z") return rob_z, "robust" return std_z, "standard" Running Example — Bybit BTC-PERP ↔ Lighter BTC-PERP (W = 24h)

Scenario: BTC crashes 12% over 2 hours at 03:00 UTC. Bybit order book thins out. One 5-minute bar records ε = 0.45% (a feed glitch as a single large market order clears thin asks).

Metric	Standard Z	Robust Z	Outcome
Window mean / median	θ = 0.126%	median = 0.10%	Robust unaffected by spike
σ estimate	σ = 0.060% (inflated 4×)	σ_r = 0.015%	Standard σ blown up
Z-score for next bar ε = 0.32%	z = (0.32−0.126)/0.060 = **3.2**	z = (0.32−0.10)/0.015 = **14.7**	—
Z-score for real signal ε = 0.22%	z = **1.6** (below entry threshold 2.0)	z = **8.0** (strong signal)	Standard misses entry
After outlier leaves window (24h later)	z recalibrates correctly	z unchanged (already correct)	Robust reacts immediately

Decision: During the BTC crash window, use **robust z for entry decisions**. The standard z produced a false alarm at ε=0.32% (z=3.2 suggesting strong signal, but driven by inflated σ from the spike) and missed the real signal at ε=0.22%. The robust z correctly ranked signals by true deviation from median spread.

Exercises

Exercise 10.1 — Compute Robust Z

Given the following window of 7 ε values (all in %):

ε = [0.10, 0.12, 0.08, 0.11, 0.80, 0.09, 0.11]

Compute: (a) median, (b) MAD, (c) σ_robust, (d) robust_z for the *last point* ε = 0.11%.

▶ Show Answer

Exercise 10.2 — Discrepancy Between Standard Z and Robust Z

You observe: **robust_z = 2.3**, **standard_z = 4.1**. Your entry threshold is |z| > 2.0.

(a) What does the discrepancy of 1.8 σ-units suggest? (b) Should you enter the trade? (c) What additional check would you perform?

▶ Show Answer

Ch.10 Z-score & Robust Statistics | ต่อไป → **Ch.11: Entry/Exit State Machine**

Statistical Arbitrage · บทที่ 11

Entry/Exit State Machine: กลไกการเข้าและออกจาก Trade

State machine ควบคุมทุก transition — ไม่มี ad-hoc decision
ตัวอย่างหลัก: Bybit BTC Perp ↔ Lighter BTC Perp
แก่นของบทนี้ — อ่านก่อน
แทนที่จะตัดสินใจ "เข้า/ออก" แบบ ad-hoc ทุกครั้ง ให้ออกแบบ **State Machine** ที่กำหนด state ทั้งหมดและเงื่อนไขการ transition อย่างชัดเจน — ป้องกัน race condition, double-entry, และ logic กระจัดกระจาย
NO_TRADE → WATCH → PAPER_GO → MANUAL_GO → DE_RISK → EXIT → INVALIDATED

11.1 ทำไมต้องมี State Machine

ในระบบ trading จริง สิ่งที่เกิดขึ้นพร้อมกันมีมาก: ราคาอัปเดตทุก millisecond, สัญญาณ cointegration เปลี่ยน, regime model อัปเดต, manual approval เข้ามา — ถ้าไม่มีโครงสร้างที่ชัดเจน โค้ดจะกลายเป็น chaos

ปัญหาที่พบบ่อยเมื่อไม่มี State Machine

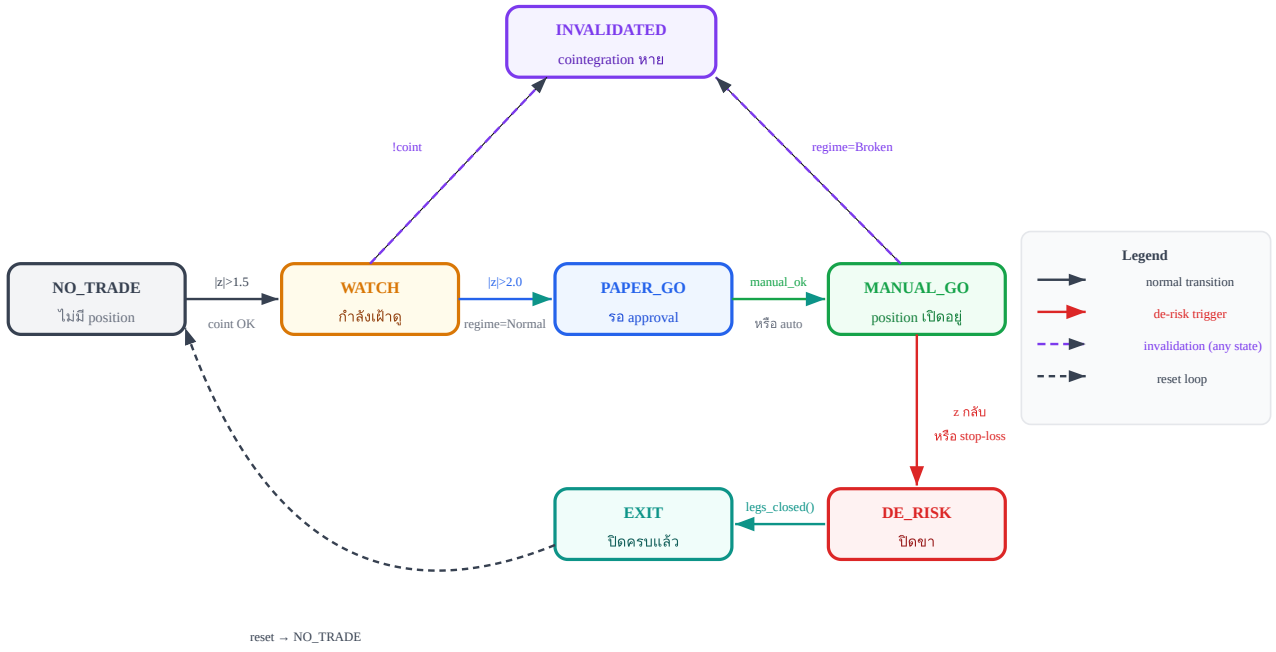
- **Race condition:** เปิด position สองครั้งเพราะ signal มาพร้อมกันสองทาง
- **Double-entry:** ระบบ "ลืม" ว่า position เปิดอยู่แล้ว และเพิ่มขนาดซ้ำ
- **Logic กระจัดกระจาย:** เงื่อนไข if/else กระจายทั่วโค้ด ไม่รู้ว่า "สถานะ" ปัจจุบันคืออะไร
- **Stop-loss ไม่ trigger:** เช็คเงื่อนไขผิด state ทำให้ stop-loss ไม่ทำงาน

ข้อดีของ State Machine

- ทุก transition ต้องผ่านเงื่อนไขที่กำหนดไว้ล่วงหน้า — ไม่มีการข้ามขั้นตอน
- ณ ทุกจังหวะ ระบบรู้ว่าอยู่ใน state ไหน และ action ที่ valid มีอะไรบ้าง
- ง่ายต่อการ log, debug, และ audit trail
- สามารถ test แต่ละ transition แยกกันได้

11.2 State Diagram — 7 States และ Transitions

State machine ของเราประกอบด้วย 7 states หลัก แต่ละ state มี transitions ที่ชัดเจน:



State diagram: ทุก transition มีเงื่อนไขชัดเจน — ไม่มีการข้ามขั้นตอน

11.3 Transition Conditions Table

ตารางด้านล่างสรุป transition ทั้งหมด พร้อมเงื่อนไขและ action ที่เกิดขึ้น:

State ปัจจุบัน	เงื่อนไข	Next State	Action ที่ทำ
NO_TRADE	|z| > 1.5 AND coint_ok AND regime = Normal	WATCH	บันทึก z_watch, ตั้ง alert
WATCH	|z| > 2.0 AND coint_ok AND regime = Normal	PAPER_GO	บันทึก entry_zscore, ส่ง paper signal
WATCH	!coint_ok OR regime = Broken	INVALIDATED	ล้าง watch state, log เหตุผล
WATCH	|z| < 1.0 (signal หาย)	NO_TRADE	reset กลับ
PAPER_GO	manual_ok = True (หรือ auto_mode)	MANUAL_GO	ส่ง order จริงทั้งสองขา
MANUAL_GO	z × sign(entry_z) < 0.5 OR stop_loss_hit()	DE_RISK	เริ่มปิด position ทีละขา
MANUAL_GO	!coint_ok OR regime = Broken	INVALIDATED	ปิด position ทันที (emergency)
DE_RISK	legs_closed() = True	EXIT	บันทึก P&L, ล้าง state
EXIT / INVALIDATED	หลัง cooldown	NO_TRADE	reset ทุก field

11.4 Pseudo-code: StateMachine Class

โค้ดด้านล่างแสดงโครงสร้างหลักของ State Machine — ทุก transition อยู่ใน method เดียว `update()` เพื่อความชัดเจน:

class StateMachine: def __init__(self): self.state = "NO_TRADE" self.entry_zscore = None def update(self, zscore, regime, coint_ok, manual_ok=False): if self.state == "NO_TRADE": if abs(zscore) > 1.5 and coint_ok and regime == "Normal": self.state = "WATCH" elif self.state == "WATCH": if abs(zscore) > 2.0 and coint_ok and regime == "Normal": self.state = "PAPER_GO" self.entry_zscore = zscore elif not coint_ok or regime == "Broken": self.state = "INVALIDATED" elif abs(zscore) < 1.0: self.state = "NO_TRADE" # signal หาย reset elif self.state == "PAPER_GO": if manual_ok: self.state = "MANUAL_GO" elif self.state == "MANUAL_GO": if zscore * sign(self.entry_zscore) < 0.5 or stop_loss_hit(): self.state = "DE_RISK" elif not coint_ok or regime == "Broken": self.state = "INVALIDATED" elif self.state == "DE_RISK": if legs_closed(): self.state = "EXIT" return self.state
หมายเหตุสำคัญเกี่ยวกับโค้ด

- `sign(self.entry_zscore)` บอกทิศทาง: ถ้า entry ที่ z = +2.3 → รอให้ z กลับมาต่ำกว่า +0.5
- `stop_loss_hit()` ควร implement แยกต่างหาก เช็คทั้ง unrealized P&L และ z-score absolute
- `legs_closed()` เช็คว่าทั้งสองขา (long และ short) ได้รับ fill ครบแล้ว
- ใน production ควรเพิ่ม persistence — บันทึก state ลง DB เพื่อกู้คืนได้หลัง crash

11.5 Running Example: Bybit↔Lighter BTC Perp

Running Example: Sequence ของ State Transitions จริง

สมมติเวลา 14:00 น. วันหนึ่ง ระบบเริ่มตรวจพบ signal ใน Bybit↔Lighter BTC perp pair:

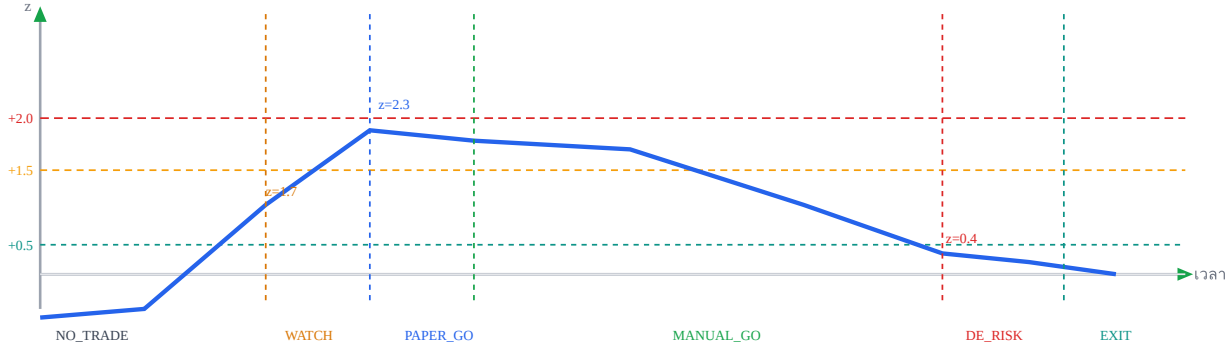
เวลา	z-score	Event	State เก่า	State ใหม่
14:00	0.8	ปกติ ไม่มีสัญญาณ	NO_TRADE	NO_TRADE
14:15	1.7	|z| > 1.5, coint OK, regime Normal	NO_TRADE	**WATCH**
14:22	2.3	|z| > 2.0, confirm signal	WATCH	**PAPER_GO**
14:25	2.1	trader กด approve (manual_ok=True)	PAPER_GO	**MANUAL_GO**
14:25	2.1	ส่ง order: Short Bybit, Long Lighter	MANUAL_GO	MANUAL_GO
15:10	0.4	z < 0.5 (mean reversion สำเร็จ)	MANUAL_GO	**DE_RISK**
15:12	0.3	ปิดทั้งสองขาครบ (legs_closed)	DE_RISK	**EXIT**
15:20	—	หลัง cooldown 8 นาที	EXIT	NO_TRADE

ผลลัพธ์ของ Trade นี้

Entry: z = +2.3 (Short Bybit ที่แพง, Long Lighter ที่ถูก)
หมายเหตุ: threshold เข้า trade คือ z = 2.0 — ค่า 2.3 คือ z จริง ณ ที่ fill (realized z) ที่ "เกิน" 2.0 พอดี ไม่ใช่ threshold ตัวที่สอง
Exit: z = +0.4 (spread กลับมาใกล้ค่ากลาง)

กำไร ≈ (2.3 − 0.4) × σ_spread × position_size = 1.9σ × ขนาด position
เวลาถือ: ~47 นาที ตรงกับ half-life ที่ estimate ไว้ ~45 นาที

การ Visualize Sequence บน z-score Timeline



z-score ขึ้นถึง +2.3 → ผ่านทุก state → กลับมา +0.4 และออกจาก position

11.6 Grid Trading บน OU Process

Key Idea
Grid Trading บน Raw Price เสี่ยงเรื่อง "กรอบแตก" เมื่อตลาดเป็นเทรนด์. แต่ถ้าวาง Grid บน ε ที่พิสูจน์แล้วว่า Mean-Revert (OU Process) — กรอบจะแตกก็ต่อเมื่อ Cointegration พัง ซึ่งเรามี Hamilton Filter คอยตรวจแล้ว

	Grid บน Raw Price	Grid บน OU Process (ε)
เสี่ยงกรอบแตก	สูง — เมื่อตลาด trend	ต่ำ — ε mean-reverts โดยนิสัย
Market-Neutral	❌ — เจ็บถ้า BTC พุ่ง/ดิ่ง	✅ — β hedge ปิดความเสี่ยงทิศทาง
Grid Step คำนวณยังไง	อิงจาก % หรือ ATR	อิงจาก σ_stat (มีสถิติหนุนหลัง)
Hard Stop	กำหนดยาก	ชัดเจน — Broken regime หรือ ADF p>0.1

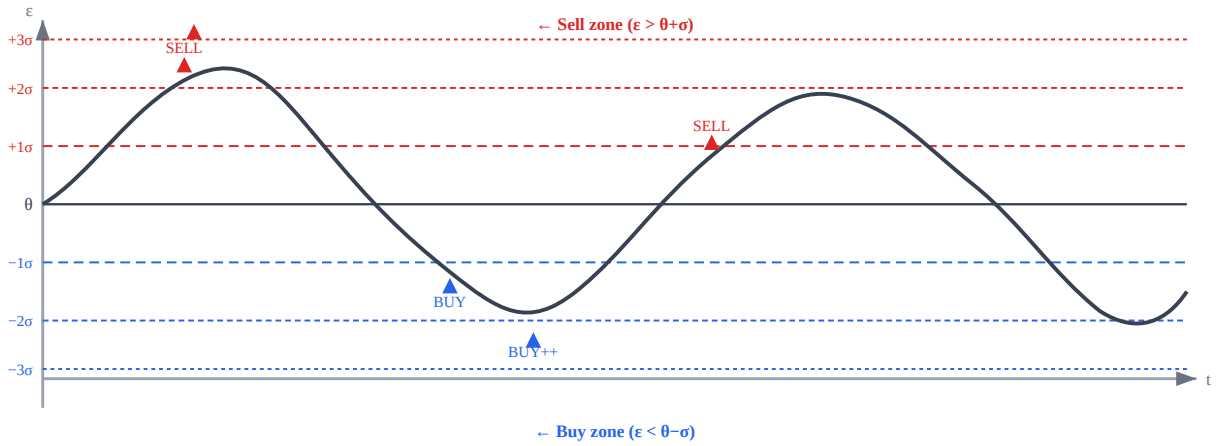
การคำนวณ Grid Levels

Grid levels คำนวณจาก:

- Grid levels: θ ± k × σ_stat สำหรับ k = 1, 2, 3, ...
- Grid step ขั้นต่ำ: step > total_friction_cost (ดูรายละเอียดใน ch19)

ตัวอย่าง: θ = 0.001, σ_stat = 0.003, step = 1σ

- Buy levels: ε < 0.001 − 0.003 = −0.002 (z < −1)
- Strong buy: ε < −0.004 (z < −2)
- Sell levels: ε > 0.004 (z > 1)
- Minimum step (จาก ch19): 19.6 bps taker → ต้องให้แต่ละ step ให้ gross ≥ 20 bps



Grid levels บน ε: Buy ที่ด้านล่าง (สีน้ำเงิน), Sell ที่ด้านบน (สีแดง) — ε mean-reverts กลับหา θ เสมอ

Pseudo-code: OUGrid Class

class OUGrid: def __init__(self, theta, sigma_stat, levels=3, step_multiplier=1.0): self.theta = theta self.step = sigma_stat * step_multiplier self.levels = levels def get_grid_action(self, epsilon, adf_pvalue=None, regime_probs=None): # Hard stop: close all grid if cointegration breaks if adf_pvalue is not None and regime_probs is not None: if not self.is_cointegration_healthy(adf_pvalue, regime_probs): return "CLOSE_ALL_GRID" z = (epsilon - self.theta) / self.step level = int(abs(z)) # which grid level crossed if level >= self.levels: return "HARD_STOP" # z >= max_levels → too far, may be broken if z < -0.5: # below θ → buy (long spread) return f"BUY_LEVEL_{level}" elif z > 0.5: # above θ → sell (short spread) return f"SELL_LEVEL_{level}" else: return "HOLD" def is_cointegration_healthy(self, adf_pvalue, regime_probs): # Close all grid if cointegration fails return adf_pvalue < 0.05 and regime_probs['broken'] < 0.5

Grid Trading vs State Machine

Grid Trading เป็น **parallel tracks** — สามารถมีหลาย position พร้อมกันได้ตามแต่ละ level ที่ถูก trigger ต่างจาก State Machine ใน Ch.11 ที่เป็น **sequential** (1 position ต่อครั้ง และต้องรอ EXIT ก่อนเปิดใหม่) Grid เหมาะกับพอร์ตที่มีทุนมากพอรองรับหลาย levels พร้อมกัน

ความเสี่ยงของ Grid Trading

ความเสี่ยงของ Grid คือการสะสม position ไปเรื่อยๆ ถ้า ε ไม่กลับ → ต้องตั้ง max_levels และ cointegration health check ทุก bar ถ้า ADF p-value > 0.1 หรือ regime = Broken → ปิด grid ทั้งหมดทันที

Running Example: Bybit↔Lighter — Grid บน OU Process

- θ = 0.001 (0.1%), σ_stat = 0.003 (0.3%), grid step = 1σ
- Total friction ≈ 20 bps (ดู ch19) → gross per step = 0.3% = 30 bps > 20 bps ✅
- Grid levels: BUY ที่ z < −1 (ε < −0.002), BUY+++ ที่ z < −2 (ε < −0.005), SELL ที่ z > 1
- ใน 30 วัน: grid triggered 47 ครั้ง, avg hold 5.2h, avg gross/trade = 28 bps, net = 8 bps (หลัง friction)

โจทย์ 11.4 — Grid Step ขั้นต่ำ

ถ้า σ_stat = 0.004 และ total friction = 20 bps — กี่ sigma ต่อ step จึงจะทำกำไรได้? (ใช้ 1% notional per step)

▶ ดูคำตอบ

11.7 แบบฝึกหัด

โจทย์ 11.1 — Transition ที่ถูกต้อง

ระบบอยู่ใน state MANUAL_GO กับ entry_zscore = +2.5 (Short Bybit, Long Lighter)
ปัจจุบัน z-score = +0.3 และ cointegration ยังคง OK
ถาม: ระบบควร transition ไป state ไหน? เพราะเหตุใด?

▶ ดูคำตอบ
โจทย์ 11.2 — Emergency Invalidation

ระบบอยู่ใน state MANUAL_GO กำลัง hold position (Short Bybit, Long Lighter)
z-score = +1.8 (ยังไม่ถึง exit threshold) แต่กะทันหัน cointegration test fail (p-value = 0.15)
ถาม: ระบบควรทำอะไร? อธิบาย action และเหตุผล

▶ ดูคำตอบ
โจทย์ 11.3 — ออกแบบ Anti-Thrashing Logic

ปัญหา: z-score วนอยู่ที่ 1.45–1.55 ทำให้ระบบ transition NO_TRADE → WATCH → NO_TRADE ซ้ำๆ ทุก 2 นาที (เรียกว่า "thrashing")
ถาม: ออกแบบ modification ใน StateMachine เพื่อแก้ปัญหานี้ โดยไม่เปลี่ยน threshold หลัก

▶ ดูคำตอบ

Ch.11 Entry/Exit State Machine | ต่อไป → **Ch.12: Position Sizing**

Statistical Arbitrage · บทที่ 12

Position Sizing: คำนวณขนาด Position อย่างถูกต้อง

Kelly criterion + half-life scaling + jump adjustment = position ที่ optimal
ตัวอย่างหลัก: Bybit BTC Perp ↔ Lighter BTC Perp
แก่นของบทนี้ — อ่านก่อน
Kelly criterion บอก **fraction ของทุนที่ควรเสี่ยง** เพื่อให้ long-run geometric growth สูงสุด แต่สำหรับ stat arb ต้องปรับด้วย half-life (ความเร็ว mean reversion) และ jump risk เพื่อให้ size เหมาะกับลักษณะ pair จริงๆ
$f^* = edge / \sigma^2$ — Kelly criterion (continuous return)

12.1 Kelly Criterion พื้นฐาน

Kelly criterion มาจาก maximization ของ E[log(wealth)] สำหรับ gamble ที่มี edge:

สูตร Kelly สำหรับ Continuous Return
$f^* = \mu / \sigma^2 = edge / variance$

μ = expected return ต่อ period (edge)
σ^2 = variance ของ return ต่อ period
f^* = fraction ของ capital ที่ควรลงทุน (0 ถึง 1)

ในรูปแบบ bet แบบ binary (win/lose) สูตรคือ $f^* = (p \cdot b - q)/b = (p \cdot b - (1-p))/b$

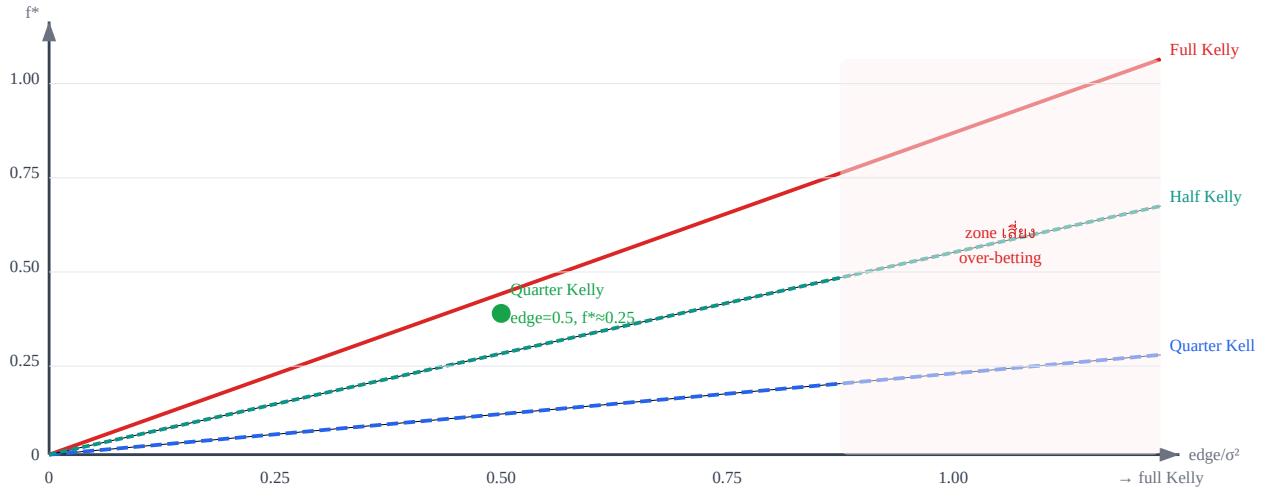
ตัวอย่างตัวเลข: Kelly fraction

สมมติ spread มี edge = 0.6% per trade และ σ = 1.2% per trade:

$f^* = edge / \sigma^2 = 0.006 / (0.012)^2 = 0.006 / 0.000144 = 41.67$ → f^* คือ fraction ของทุนที่ Kelly แนะนำให้ลงทุน — ไม่ใช่ค่า leverage → $f^* = 41.67$ (>> 1) เป็นค่าที่ไม่สมเหตุสมผล (สัญญาณว่า input ผิด — ดูหมายเหตุด้านล่าง) → ในทางปฏิบัติใช้ fractional Kelly: $f_{actual} = f^* \times 0.25$ (quarter Kelly) → $f_{actual} = 41.67 \times 0.25 \approx 10.4$ ซึ่งยังสูงเกินจริง → ต้องปรับด้วย half-life, jump discount และ drawdown limit ต่อไป
หมายเหตุ: $f^* > 1$ คือสัญญาณเตือน input ผิด

ถ้าคำนวณแล้วได้ $f^* > 1$ (เช่น 41.67) นั่นไม่ได้แปลว่าควรใช้ leverage 41 เท่า — แต่เป็น **red flag** ว่า μ และ σ ที่ใส่เข้าไปไม่ใช่ per-period strategy return จริง (edge 0.8% กับ vol 1.5% ต่อ trade ไม่ได้หมายความว่าควร bet 35 เท่าของทุน). f^* ของ stat arb จริงหลังหัก cost อย่างซื่อสัตย์มักต่ำกว่า 1 มาก — ในทางปฏิบัติให้ใช้ **drawdown-based sizing limit** (§12.4) เป็นตัวกำหนดขนาดจริง

Kelly Fraction vs Edge/Variance — Parabola Curve



Full Kelly ให้ growth สูงสุดในทางทฤษฎี แต่ drawdown รุนแรงมาก — ในทางปฏิบัติใช้ Quarter Kelly

12.2 Half-life Scaling — ปรับขนาดตาม Mean Reversion Speed

ถ้า half-life ยาว (spread mean-revert ช้า) เราต้อง hold position นานกว่า ความเสี่ยงสะสมมากขึ้น — Kelly เดิมไม่ได้รวม factor นี้

Half-life Scaling Factor
size_adjusted = f* × min(1, target_halflife / actual_halflife)

target_halflife = half-life ที่เราออกแบบ strategy สำหรับ (เช่น 5 ชั่วโมง)
actual_halflife = half-life จริงที่คำนวณได้จากข้อมูล
ถ้า actual > target → scale down size (position เล็กลง เพราะ mean-revert ช้า)

actual_halflife	target_halflife	scaling factor	ผลกระทบ
3 ชั่วโมง	5 ชั่วโมง	min(1, 5/3) = 1.0	ไม่ลด (fast mean reversion — ดี)
5 ชั่วโมง	5 ชั่วโมง	min(1, 5/5) = 1.0	ไม่ลด (ตรงเป้า)
10 ชั่วโมง	5 ชั่วโมง	min(1, 5/10) = 0.5	ลด size เหลือ 50%
24 ชั่วโมง	5 ชั่วโมง	min(1, 5/24) ≈ 0.21	ลด size เหลือ 21% (slow — ระวัง)

สังเกตสำคัญ: Half-life เปลี่ยนตามเวลา

Half-life ของ Bybit↔Lighter spread อาจสั้นกว่าช่วง sideways market (1-2h) และยาวขึ้นในช่วง trending market (8-12h) — ต้องคำนวณ half-life ใหม่ทุก N ชั่วโมง ไม่ใช่ fix ไว้ตลอด

12.3 Jump-adjusted Kelly

ในตลาด crypto มี jump risk สูง: ราคาอาจกระโดดข้ามคืนหรือช่วง liquidation cascade ค่าพยากรณ์ E[log(1+f·R)] ลดลงอย่างมากเมื่อมี jump — Kelly สูตรปกติ assume normal returns ซึ่งไม่ถูกต้อง

Jump-adjusted Kelly
f*_adjusted = f* × jump_discount

jump_discount = 1 − P(jump) × (expected_loss_given_jump / max_tolerable_loss)
ค่าปกติ: jump_discount ∈ [0.6, 0.9] ขึ้นอยู่กับ volatility regime

ช่วง low vol: 0.85–0.90 | ช่วง high vol (เช่น ก่อน major news): 0.60–0.70

ทำไม Jump ถึงทำลาย Kelly?

สมมติ Kelly บอก f* = 0.5 (ลง 50% ของทุน) แต่มี 1% chance ที่ spread จะกระโดด 5σ ในพริบตา:

ขาดทุน = 0.5 × capital × 5 × σ_spread ถ้า σ_spread = 2% และ capital = $100,000 ขาดทุน = 0.5 × $100,000 × 5 × 0.02 = $5,000 จากเหตุการณ์ 1 ครั้ง ถ้าลด f* ด้วย jump_discount = 0.8: f*_adj = 0.5 × 0.8 = 0.4 ขาดทุน = 0.4 × $100,000 × 5 × 0.02 = $4,000 (ดีขึ้น 20%)

12.4 Max Drawdown Constraint

นอกจาก Kelly และ half-life scaling แล้ว ต้องตั้ง hard limit จาก max drawdown ที่ยอมรับได้:

Max Drawdown Position Limit
size_dd = max_loss / (entry_z × σ_spread)

max_loss = capital × max_dd fraction (เช่น 2% ของ capital)

entry_z = z-score ณ จุดเข้า trade

σ_spread = volatility ของ spread

ขนาด position สุดท้าย = min(kelly_size, size_dd)

ตัวอย่าง: Max Drawdown เป็น Binding Constraint

Kelly บอก $80,000 แต่ drawdown limit บอก $45,000 → ใช้ $45,000
สถานการณ์นี้เกิดขึ้นเมื่อ entry_z ต่ำ (เช่น z = 2.0 ต่ำกว่า z = 3.5) หรือ σ_spread สูง
Kelly ไม่ได้รู้จัก "ทุนกี่บาท" — Drawdown limit ป้องกัน ruin ในกรณีเลวร้าย

12.5 Pseudo-code: compute_position_size()

def compute_position_size(capital, edge, sigma, halflife, entry_z, target_halflife=5, jump_discount=0.8, max_dd=0.02): # Step 1: Kelly fraction kelly_f = edge / (sigma ** 2) # Step 2: Half-life scaling (ลดถ้า mean reversion ช้า) hl_scale = min(1.0, target_halflife / halflife) # Step 3: Jump discount (ลดความเสี่ยง tail event) jump_scale = jump_discount # Step 4: Max drawdown constraint (hard limit) dd_limit = (capital * max_dd) / (entry_z * sigma) # Step 5: รวมทุก factor raw_size = capital * kelly_f * hl_scale * jump_scale # Step 6: ใช้ขนาดที่น้อยกว่า (conservative) return min(raw_size, dd_limit)
หมายเหตุเกี่ยวกับ sigma ใน dd_limit

ตัวหาร `entry_z * sigma` ใน dd_limit มาจากสมมติว่า worst-case adverse move คือ spread เคลื่อนจาก z ณ จุดเข้า (`entry_z`) กลับไปที่ค่ากลาง — ระยะที่ spread วิ่งสวน position ได้จึง scale ตาม `entry_z` โดยตรง ยิ่ง entry ที่ z สูง position ยิ่งต้องเล็กลง ถ้าต้องการ conservative มากขึ้นให้บวก buffer เข้ากับ entry_z ตาม risk appetite

12.6 Running Example: Bybit↔Lighter BTC Perp

Running Example: คำนวณขนาด Position ทีละขั้น

สถานการณ์: ระบบเพิ่ง transition ไป MANUAL_GO — ต้องคำนวณขนาด order ก่อนส่ง

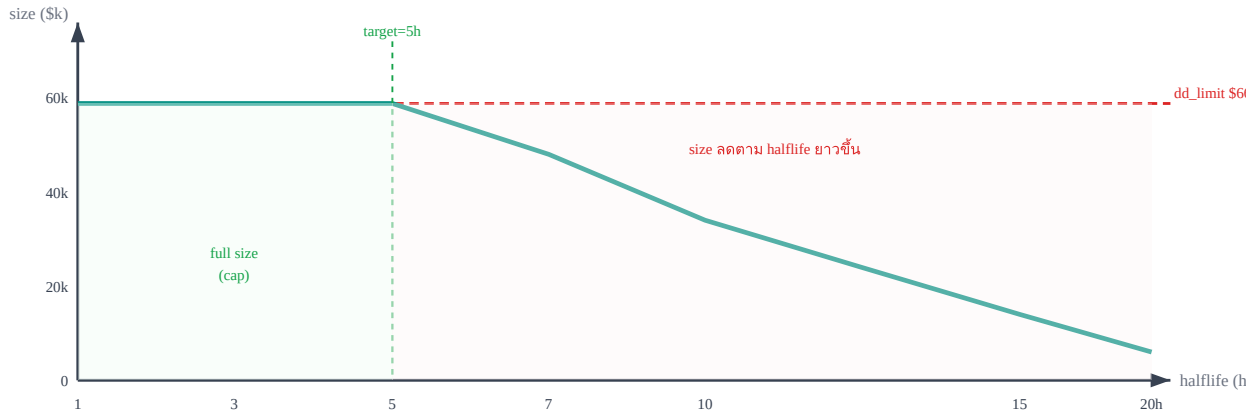
Parameter	ค่า	ที่มา
capital	$100,000	กำหนดโดย risk manager
edge (μ)	0.8% = 0.008	historical P&L per trade
sigma (σ)	1.5% = 0.015	std ของ spread returns (30d)
halflife	4 ชั่วโมง	OU model calibration
target_halflife	5 ชั่วโมง	ออกแบบ strategy สำหรับ 5h
jump_discount	0.80	low-vol regime
max_dd	2% = 0.02	risk limit

การคำนวณทีละขั้น

Step 1: Kelly fraction kelly_f = edge / sigma² = 0.008 / (0.015)² = 0.008 / 0.000225 = 35.56 ← f* >> 1: input ไม่ใช่ return จริง ต้องปรับ Step 2: Half-life scaling hl_scale = min(1.0, 5 / 4) = min(1.0, 1.25) = 1.0 ← actual (4h) เร็วกว่า target (5h) → ไม่ลด Step 3: Jump discount jump_scale = 0.80 Step 4: Max drawdown limit dd_limit = (100,000 × 0.02) / (2 × 0.015) = 2,000 / 0.03 = $66,667 Step 5: Raw size raw_size = 100,000 × 35.56 × 1.0 × 0.80 = $2,844,800 ← สูงกว่า dd_limit มาก! Step 6: Final size = min($2,844,800, $66,667) = $66,667
ผลลัพธ์และการตีความ

- Position size สุดท้าย: **$66,667** (ถูก cap โดย drawdown limit)
- Kelly บอก $2.8M แต่นั่นคือ theoretical optimum ภายใต้ assumptions ที่ unrealistic
- Drawdown limit เป็น binding constraint — ในทางปฏิบัติ max_dd มักเป็น primary constraint
- ทั้งสองขา: Long Lighter ≈ $66,667 | Short Bybit ≈ $66,667 × β

Sensitivity Analysis: ผลของ Half-life ที่ต่างกัน



เมื่อ halflife น้อยกว่า target (5h) → size cap ที่ dd_limit | เมื่อยาวกว่า → size ลดลงตามสัดส่วน

12.7 แบบฝึกหัด

โจทย์ 12.1 — คำนวณ Kelly Size

Pair ใหม่มี edge = 1.2% per trade, σ = 2.0% per trade
capital = $50,000, halflife = 8h, target_halflife = 5h
jump_discount = 0.75, max_dd = 3%
ถาม: คำนวณ position size ทีละขั้น และระบุว่า constraint ไหนเป็น binding?

▶ ดูคำตอบ

โจทย์ 12.2 — เปรียบเทียบ Two Pairs

เปรียบเทียบสอง pair ด้วย capital = $200,000 เดียวกัน:

- Pair A: edge=0.5%, σ=0.8%, halflife=3h, jump_discount=0.9, max_dd=2%
- Pair B: edge=1.5%, σ=3.0%, halflife=12h, jump_discount=0.65, max_dd=2%

ถาม: คำนวณ final size ของแต่ละ pair และอธิบายว่า pair ไหน "ดีกว่า" ในแง่ risk-adjusted

▶ ดูคำตอบ

โจทย์ 12.3 — Dynamic Sizing เมื่อ Regime เปลี่ยน

ระหว่าง hold position อยู่ ระบบตรวจพบ volatility regime เปลี่ยนจาก low-vol เป็น high-vol:
jump_discount ลดจาก 0.85 เป็น 0.60 และ σ_spread เพิ่มจาก 1.5% เป็น 2.5%
Current position size = $80,000, capital = $100,000, max_dd = 2%
ถาม: ควรทำอะไรกับ position ที่ถืออยู่?

▶ ดูคำตอบ

Ch.12 Position Sizing | ต่อไป → **Ch.13: Perpetual Basis Strategy**

Statistical Arbitrage · บทที่ 13

Perpetual Basis Strategy: Trade ความผิดปกติของ Funding

Basis = perp − spot index — เบี่ยงเกิน fair value เมื่อไหร่ให้เทรด
ตัวอย่างหลัก: Bybit BTC Perp ↔ Spot Index
แก่นของบทนี้ — อ่านก่อน
ราคา perpetual futures ควรอยู่ใกล้ spot index เพราะมีกลไก funding rate ดึงให้กลับ แต่ในช่วงที่ sentiment รุนแรง basis เบี่ยงเกิน fair value ชั่วคราว — นี่คือโอกาส trade ที่มีขอบเขตชัดเจน
Basis_fair ≈ spot × rate_per_8h × (hours_left/8) | entry เมื่อ |Basis_actual − Basis_fair| > k × σ_basis

13.1 กลไก Funding Rate

Perpetual futures ไม่มี expiry — กลไกที่ทำให้ราคาใกล้ spot คือ **funding rate** ที่จ่ายระหว่าง long และ short ทุกรอบ:

กลไก Funding Rate ทำงานอย่างไร

- ทุก 8 ชั่วโมง (Bybit) หรือทุก 1 ชั่วโมง (Lighter) คำนวณ funding rate จาก basis และ interest rate
- ถ้า perp > spot (basis > 0): **longs จ่าย shorts** — ดึงให้ longs ปิด position → ราคา perp ลง
- ถ้า perp < spot (basis < 0): **shorts จ่าย longs** — ดึงให้ shorts ปิด position → ราคา perp ขึ้น
- Funding amount = position_size × funding_rate × (interval / 8h)

ตัวอย่าง: Bybit BTC Funding

Funding rate = +0.05% (longs จ่าย shorts) ทุก 8 ชั่วโมง = +0.15%/วัน = +4.5%/เดือน
Long BTC perp $100,000 → จ่าย $50 ทุก 8 ชั่วโมง ≈ $150/วัน
นี่คือ "cost of carry" สำหรับ long perp ในช่วง bullish market

สูตร funding rate มาตรฐาน (Bybit-style):

funding_rate = clamp(premium_index + clamp(interest_rate - premium_index, -0.05%, +0.05%), -0.75%, +0.75%) premium_index = (P_perp - P_spot_index) / P_spot_index ← basis เป็น % interest_rate = 0.01% (Bybit กำหนด fixed = 0.01% ต่อรอบ 8h)

13.2 Basis = Perp − Spot: นิยามและ Fair Value

ในทางปฏิบัติ basis วัดเป็น absolute price หรือ percentage:

นิยาม Basis
Basis_t = P_perp_t − P_spot_index_t

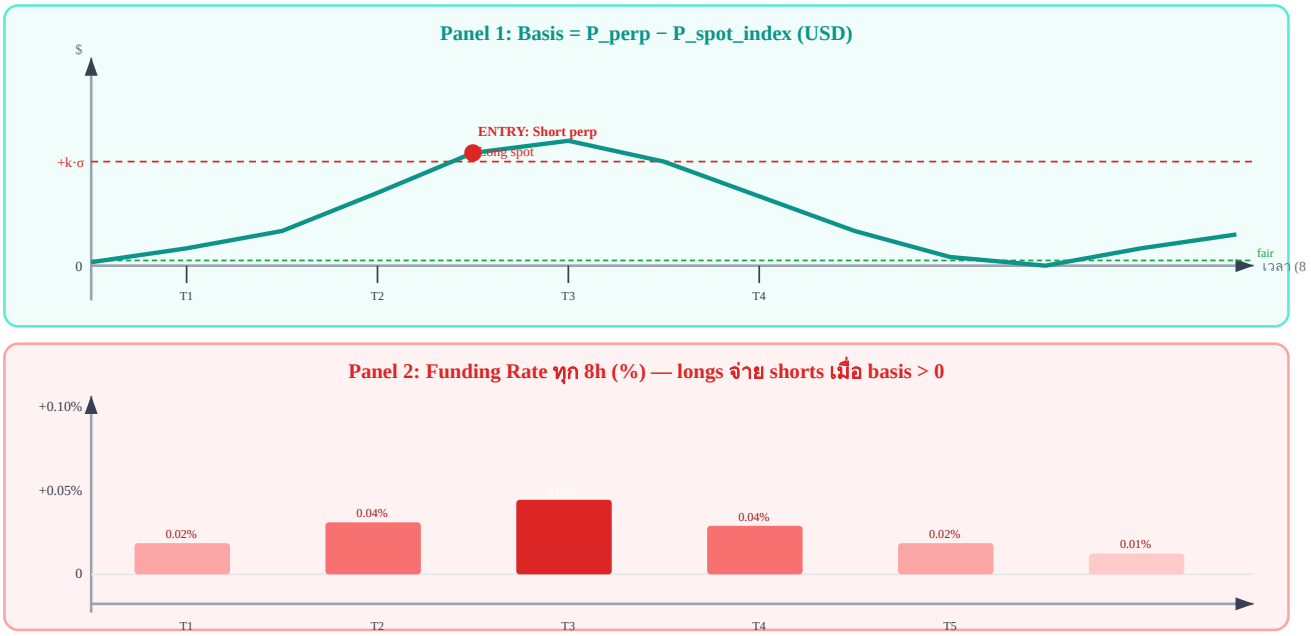
P_perp = mark price ของ perpetual futures
P_spot_index = weighted average ของราคา spot จาก multiple exchanges
Basis_fair = ค่าที่ basis "ควร" เป็น ตาม funding rate model

สำหรับ perpetual ที่ไม่มี expiry fair value basis ถูกขับเคลื่อนด้วย **funding rate** ไม่ใช่ cost of carry ของ money market — basis ที่ fair คือค่าที่ทำให้ funding payment ในรอบถัดไปสมดุล prorate ตามเวลาที่เหลือจนถึง funding ครั้งหน้า:

Basis_fair = P_spot × rate_per_8h × (hours_left / 8) rate_per_8h = funding rate ต่อรอบ 8 ชั่วโมง (Bybit default ≈ 0.01%) hours_left = เวลาที่เหลือจนถึง funding ครั้งถัดไป (หน่วย: ชั่วโมง) ตัวอย่าง: P_spot = $65,000, rate_per_8h = 0.01% = 0.0001, hours_left = 4 Basis_fair = 65,000 × 0.0001 × (4 / 8) = 65,000 × 0.0001 × 0.5 ≈ $3.25

Fair value basis มักใกล้ศูนย์มากสำหรับ BTC perp เพราะ rate_per_8h ต่ำและเหลือเวลาไม่กี่ชั่วโมง — basis เบี่ยงเกินจากการ sentiment หรือ imbalance ของ demand

13.3 SVG Diagram: Basis Timeline และ Funding Rate



Panel บน: basis สูงขึ้นจนเกิน threshold → Entry signal | Panel ล่าง: funding rate สูงสุดที่ T3 ตรงกับ basis สูงสุด

13.4 Entry/Exit Rules

กฎการเข้าและออก basis trade แบ่งตาม direction ของ basis deviation:

เงื่อนไข	Direction	Action	เหตุผล
basis > fair + k×σ_basis	Basis บวกสูงเกิน	Short perp + Long spot	Perp แพงเกินไป รอ basis กลับลง
basis < fair − k×σ_basis	Basis ลบต่ำเกิน	Long perp + Short spot	Perp ถูกเกินไป รอ basis กลับขึ้น
basis → fair value	กลับปกติ	ปิดทั้งสองขา	Mean reversion สำเร็จ take profit
funding payment ≥ target yield	ได้ yield แล้ว	ปิดทั้งสองขา	เก็บ funding เพียงพอแล้ว
|basis| > 3×σ_basis (blow-up)	เกิน stop-loss	ปิด emergency	Basis ไม่กลับ — regime เปลี่ยน

สองแหล่ง P&L ของ Basis Trade

1. **Basis convergence:** กำไรเมื่อ basis กลับมา fair value (capital gain จาก spread)
2. **Funding income:** ถ้า short perp ในช่วง basis > 0 → รับ funding payment ทุก 8h ขณะ hold position

Trade ที่ดีจะมีทั้งสองแหล่ง — basis กลับ + funding เข้ากระเป๋าระหว่างรอ

ความเสี่ยงของ Basis Trade

- **Basis blow-up:** ในช่วง mania (เช่น bull run รุนแรง) basis อาจขยายไปเรื่อยๆ ก่อนกลับ
- **Spot execution risk:** การ short spot อาจมีปัญหา borrow cost หรือ availability
- **Funding flip:** funding rate อาจกลับทิศ ทำให้ long perp ต้องจ่ายแทน
- **Index divergence:** spot_index อาจ diverge จาก exchange ที่ใช้ hedge จริง

13.5 Cost Breakdown Table

ก่อน trade ต้องรู้ต้นทุนทุกประเภท เพื่อคำนวณ net edge:

ประเภทต้นทุน	ค่าปกติ (Bybit BTC)	ผลต่อ P&L	หมายเหตุ
Taker fee (entry)	0.055% per leg	−0.11% รวมสองขา	ใช้ limit order ลดเหลือ −0.01% (maker)
Taker fee (exit)	0.055% per leg	−0.11% รวมสองขา	รวม round-trip: −0.22%
Bid-ask spread	~$1–5 (BTC perp)	≈ −0.002–0.008%	ขึ้นกับ liquidity ณ เวลานั้น
Slippage	0.01–0.05%	−0.01 ถึง −0.05%	ขึ้นกับขนาด order vs market depth

Funding cost (long)	±0.01–0.10% ทุก 8h	+ หรือ − ขึ้นกับทิศ	กำไรถ้า short perp ในช่วง basis+
Spot borrow (short spot)	0.01–0.05%/วัน	−ต่อวันที่ถือ	ค่าเช่า BTC สำหรับ short บน spot margin — Bybit เรียก ~0.01%/วัน; ถ้า hedge ผ่าน inverse perp ไม่มีต้นทุนนี้
รวม round-trip	**≈ −0.25 ถึง −0.40% ต่อ trade (ไม่รวม funding income)**		

Breakeven Analysis: Basis ต้องกว้างแค่ไหนถึงคุ้ม?

ถ้า total cost ≈ 0.35% per round-trip → basis ต้อง deviate จาก fair value อย่างน้อย 0.35% จึงจะ breakeven
ถ้า σ_basis = 0.15% → entry threshold k = 0.35/0.15 ≈ 2.3σ
ในทางปฏิบัติใช้ k ≥ 2.5σ เพื่อ buffer สำหรับ slippage และ execution uncertainty

13.6 Pseudo-code: perp_basis_signal()

def perp_basis_signal(perp_price, spot_index, rate_per_8h, time_to_funding_h, # ✅ FIX: ชัดเจนว่าหน่วยเป็น ชั่วโมง hist_sigma, k=2.0): # คำนวณ basis จริง basis = perp_price - spot_index # คำนวณ fair value basis จาก funding rate (prorate ตามเวลาที่เหลือ) # ✅ FIX: rate_per_8h × (เวลาที่เหลือ/8) — unit สอดคล้องกันทั้งหมด # ❌ เดิม: rate * (time_to_funding / (8 * 3600)) — ปนหน่วย seconds กับ per-8h fair_basis = spot_index * rate_per_8h * (time_to_funding_h / 8.0) # คำนวณ z-score ของ basis deviation zscore = (basis - fair_basis) / hist_sigma # Signal logic if zscore > k: return "SHORT_PERP_LONG_SPOT", zscore # basis สูงเกิน elif zscore < -k: return "LONG_PERP_SHORT_SPOT", zscore # basis ต่ำเกิน return "NO_SIGNAL", zscore
การคำนวณ hist_sigma ของ Basis

`hist_sigma` ควรคำนวณจาก rolling std ของ (basis − fair_basis) ในช่วง 7–30 วัน
ไม่ควรใช้ σ คงที่ เพราะ basis volatility เปลี่ยนตาม market regime
ในช่วง sideways: σ ต่ำ → threshold ต่ำ → เข้าง่ายขึ้น
ในช่วง trending: σ สูง → threshold สูง → เข้ายากขึ้น (ป้องกัน false signal)

การเพิ่ม Funding Yield Forecast ใน Signal

def basis_trade_expected_pnl(basis_entry, fair_basis, hist_sigma, funding_rate, hold_hours, position_size, spot_price): # P&L จาก basis convergence (mean reversion) basis_move = basis_entry - fair_basis # คาดว่า basis จะกลับมา fair basis_pnl = basis_move * position_size / spot_price # P&L จาก funding (short perp receives funding) n_payments = hold_hours / 8 # จำนวนรอบ funding (ทุก 8h) funding_pnl = funding_rate * position_size * n_payments # ต้นทุน round-trip (fee + slippage) total_cost = position_size * 0.0035 # ≈ 0.35% return basis_pnl + funding_pnl - total_cost

13.7 Running Example: Bybit BTC Perp Basis Trade

Running Example: Bybit BTC Perp Basis Trade — ตัวเลขทีละขั้น

สถานการณ์: วันที่ BTC ราคา pump แรง longs เปิดมาก basis บวกสูงผิดปกติ

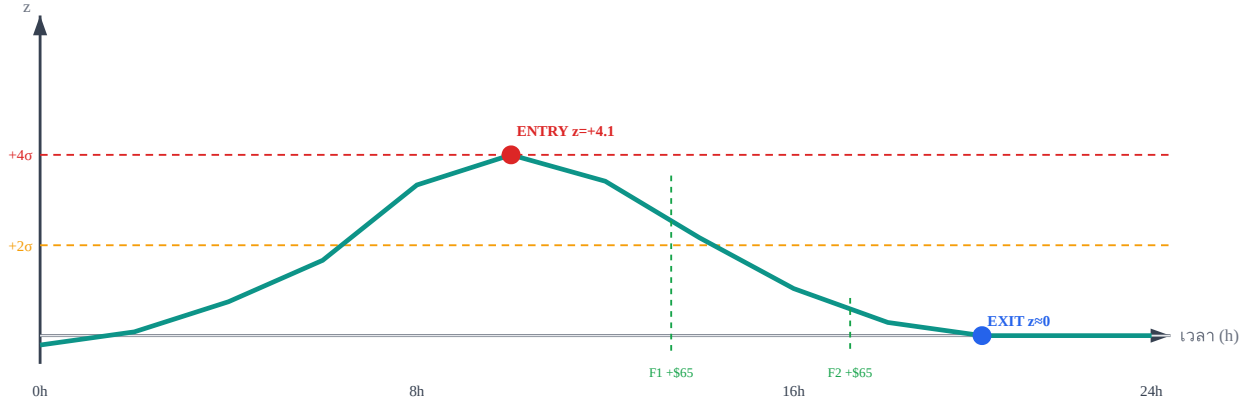
ข้อมูล Input	ค่า	ที่มา
P_perp (Bybit mark price)	$67,450	Bybit API
P_spot_index	$67,100	Bybit spot index (avg หลาย exchanges)
Basis_actual	$350 (+0.521%)	67,450 − 67,100
rate_per_8h (Bybit funding)	0.01% = 0.0001	Bybit default funding rate per 8h interval
time_to_funding_h	3.5 ชั่วโมง	เวลาถึง funding ถัดไป (หน่วย: ชั่วโมง)
Basis_fair	67,100 × 0.0001 × (3.5/8.0) ≈ $2.94	spot × rate_per_8h × (hours_left/8) ✅
hist_sigma (30d rolling)	$85 (0.127%)	std ของ (basis − fair_basis)
z-score	(350 − 2.94) / 85 ≈ **+4.1σ**	ENTRY SIGNAL!
funding_rate (current)	+0.065%	Bybit funding history

Execution Plan และ P&L

Entry (z = +4.1σ, basis = +$350): Short Bybit BTC perp : $100,000 notional at $67,450 Long spot BTC (หรือ spot index hedge): $100,000 at $67,100 Hold 16 ชั่วโมง (2 รอบ funding): Funding received (short perp, longs pay us): = 0.065% × $100,000 × 2 รอบ = $130 Exit (basis กลับ fair: basis ≈ $20): ปิด Short perp และ Long spot — กำไรมาจาก basis ที่แคบลง Position size = $100,000 / $67,450 ≈ 1.483 BTC Basis convergence P&L = (basis_entry − basis_exit) × size = ($350 − $20) × 1.483 ≈ $489 ต้นทุน: Taker fee (entry + exit, 2 legs each) = 0.22% × $100,000 = $220 Slippage estimate = 0.03% × $100,000 = $30 Net P&L: = $489 (basis convergence) + $130 (funding income) − $220 (taker fees) − $30 (slippage) = +$369 → +0.37% บน $100,000 ใน 16 ชั่วโมง ≈ Annualized ~20% (ถ้า trade frequency ≈ 1 trade/วัน)
สรุป: โครงสร้าง P&L ของ Basis Trade นี้

- **Basis convergence:** $489 — แหล่งกำไรหลัก (basis เบี่ยง 4σ กลับมา fair)
- **Funding income:** $130 — bonus ระหว่างรอ mean reversion
- **ต้นทุนรวม:** $250 — fee + slippage
- **Net: +$369** ใน 16 ชั่วโมง — measurable edge ที่ repeatable

Z-score Timeline ของ Trade นี้



Basis z-score ขึ้นถึง +4.1σ → เข้า trade → กลับมา 0 ใน 16 ชั่วโมง รับ 2 รอบ funding ระหว่างทาง

13.8 แบบฝึกหัด

โจทย์ 13.1 — คำนวณ Fair Basis และ Z-score

ข้อมูล: P_perp = $42,300, P_spot_index = $42,100
rate_per_8h = 0.01% (Bybit default), time_to_funding_h = 5.5 ชั่วโมง
hist_sigma ของ (basis − fair_basis) = $60, k = 2.0
ถาม: คำนวณ fair_basis, z-score และระบุว่ามี signal หรือไม่ถ้า k = 2.0

▶ ดูคำตอบ

โจทย์ 13.2 — คำนวณ Net P&L ของ Basis Trade

Trade: Short perp $80,000 + Long spot $80,000
Entry basis = +$280, Exit basis = +$15
Hold 24 ชั่วโมง (3 รอบ funding), funding_rate = +0.05% ต่อรอบ
Taker fee = 0.055% per leg, 4 legs total, slippage = 0.025%
สมมติ spot_price entry ≈ $42,100
ถาม: คำนวณ net P&L ทีละส่วน

▶ ดูคำตอบ

โจทย์ 13.3 — Basis Blow-up: เมื่อ Trade ขาดทุน

คุณ Short perp ที่ basis = +$300 (z = +3.0σ, hist_sigma = $100)
แต่แทนที่ basis จะกลับ basis กลับขยายไปเรื่อยๆ
Stop-loss กำหนดไว้ที่ z = +5.0σ, position size = $50,000, spot_price ≈ $67,000
ถาม: (1) Stop-loss trigger ที่ basis เท่าไร? (2) ขาดทุนเท่าไรเมื่อ stop-loss ถูก trigger?

▶ ดูคำตอบ

Ch.13 Perpetual Basis Strategy | ต่อไป → **Ch.14: Cross-venue Strategy**

Statistical Arbitrage · บทที่ 14

Cross-venue Strategy: ซื้อขายระหว่างสองตลาดพร้อมกัน

ควบคุม leg risk, synchronization, และ execution timing ข้ามตลาด
ตัวอย่างหลัก: Bybit BTC Perp ↔ Lighter BTC Perp
แก่นของบทนี้ — อ่านก่อน
กำไรมาจาก **spread** ไม่ใช่ direction — ต้องเปิดสองขาพร้อมกันเพื่อ lock spread
ถ้าเปิดได้ขาเดียว คุณไม่ได้ trade spread แต่กำลัง directional trade โดยไม่ตั้งใจ
กำไร = (ε_entry − ε_exit) − fees_A − fees_B

14.1 Latency Arbitrage vs Statistical Arbitrage

ทั้งสองกลยุทธ์ดูคล้ายกัน แต่มีความต่างพื้นฐาน ที่กำหนดทั้ง infrastructure และ edge ที่ต้องการ

มิติ	Latency Arbitrage (LA)	Statistical Arbitrage (SA)
Edge มาจาก	เร็วกว่าคนอื่น — race to fill	OU process ที่ mean-revert
Infrastructure	Colocation, FPGA, microsecond	Standard server, millisecond ก็พอ
Signal	Price discrepancy ชั่วคราว <100ms	z-score ของ spread หลาย minutes/hours
Competition	สูงมาก — HFT firms	ปานกลาง — hedge funds, prop desks
กำไรต่อ trade	เล็ก แต่บ่อยมาก	ใหญ่กว่า แต่รอนาน
Leg risk	สูงมาก — ต้องเร็วทั้งสองขา	ปานกลาง — มีเวลาจัดการ

ใช้ SA ในเล่มนี้ เพราะอะไร?

Bybit↔Lighter BTC perp มี spread ที่ mean-revert ด้วย half-life หลายชั่วโมง ไม่ต้องการความเร็วระดับ microsecond แต่ต้องการ execution ที่ **synchronous พอ** เพื่อไม่ให้ leg risk กินกำไร

14.2 Leg Risk — ความเสี่ยงที่อันตรายที่สุดใน Cross-venue

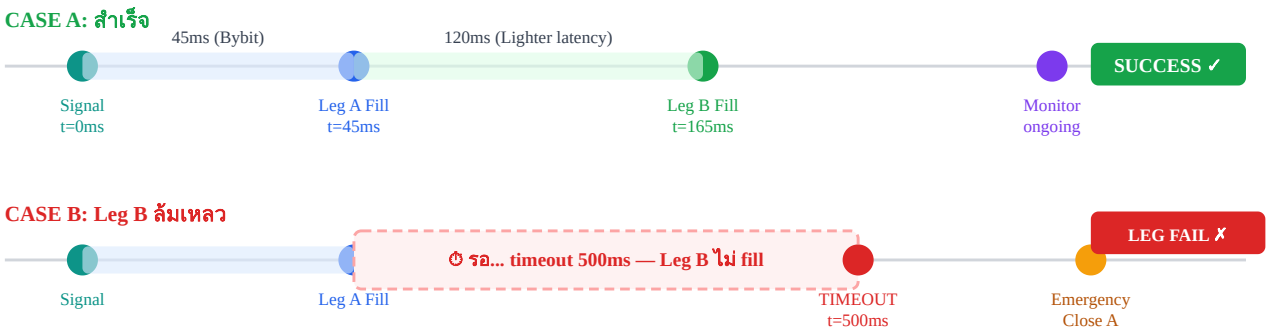
Leg risk เกิดเมื่อเปิด Leg A สำเร็จ แต่ Leg B ล้มเหลว ทำให้เกิด *single-leg exposure* — ขณะนี้คุณไม่ได้ trade spread แต่กำลัง directional long/short โดยไม่ตั้งใจ และไม่มี hedge

สถานการณ์อันตราย — Leg B Fail

สัญญาณ: ε สูงเกิน → Long Bybit + Short Lighter
สำเร็จ: Long Bybit $100,000 (Leg A) ✓

ล้มเหลว: Short Lighter ไม่ fill ใน 500ms (Leg B) ✗
ผล: คุณ Long BTC $100,000 โดยไม่มี hedge — ถ้า BTC ร่วง 2% = ขาดทุน $2,000 ทันที

Timeline ของ Leg Execution



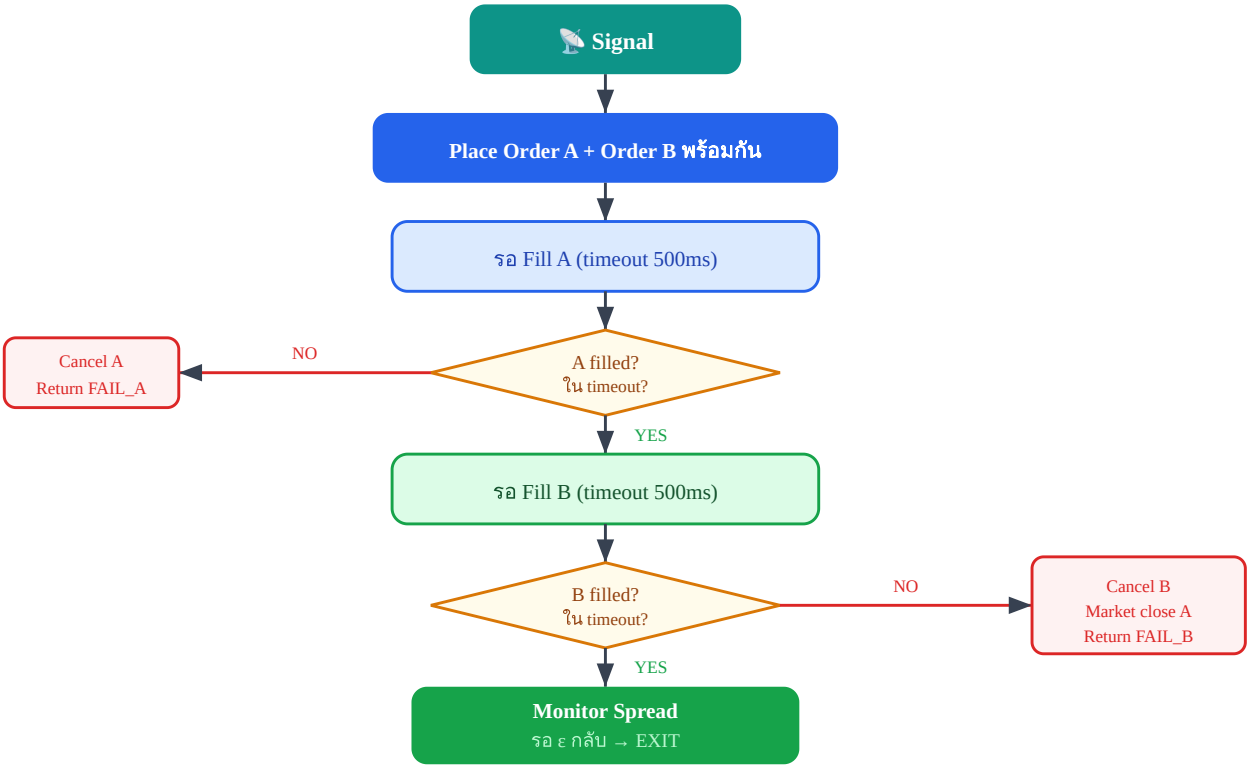
ช่วง "Leg A filled แต่ Leg B ยังไม่ fill" คือ single-leg exposure — อันตรายที่สุด

ช่วงสีชมพูระหว่าง Leg A fill ถึง timeout คือ window ที่ directional risk เต็มๆ
วิธีลด Leg Risk

- ใช้ **simultaneous order placement** — ส่งคำสั่งทั้งสองตลาดเกือบพร้อมกัน
- ตั้ง **timeout สั้น** (300–500ms) — ถ้า B ไม่ fill ให้ปิด A ทันที
- ใช้ **limit order ที่ aggressive** — ใกล้ mid price เพื่อให้ fill เร็ว
- มี **emergency close routine** พร้อมใช้ตลอดเวลา

14.3 Execution Flow Diagram

กระบวนการ execution ที่สมบูรณ์ต้องจัดการ happy path และ failure path ทุก branch



Flowchart execution: ทุก branch ต้องมี handler — ไม่มี "undefined state"

14.4 Synchronization Strategy — สามแนวทาง

วิธีที่จะส่ง order ทั้งสองขามีผลต่อ leg risk, latency overhead และ complexity ของ code

แนวทาง	วิธีการ	Leg Risk	Latency	Complexity
Simultaneous	ส่งทั้ง A และ B ในเวลาเดียวกัน (async)	ต่ำ — overlap เล็กน้อย	ต่ำ	กลาง
Sequential	ส่ง A ก่อน รอ fill แล้วจึงส่ง B	สูง — รอ fill A = exposed	สูง (A_latency + B_latency)	ต่ำ
Conditional	ส่ง A ก่อน ถ้า fill ส่ง B ทันที ถ้าไม่ cancel	กลาง — มี logic ควบคุม	กลาง	สูง

Bybit↔Lighter แนะนำ: Simultaneous + Timeout

ส่ง order ทั้งสองตลาดพร้อมกันใน async threads จากนั้น wait ด้วย timeout
ถ้าทั้งคู่ fill ภายใน 500ms → SUCCESS
ถ้า A fill แต่ B ไม่ fill → cancel B แล้ว market close A ทันที

14.5 Order Types ที่ใช้ใน Stat Arb

การเลือก order type มีผลต่อทั้ง fill probability และ slippage cost

Limit Order — แนะนำสำหรับ Stat Arb

- **ข้อดี:** ควบคุมราคาได้ — ไม่จ่ายเกิน entry price ที่คำนวณ
- **ข้อดี:** Maker fee ถูกกว่า taker fee (บางตลาดได้ rebate)
- **ข้อเสีย:** อาจไม่ fill ถ้าราคาวิ่งออกไป → partial fill risk
- **แก้ไข:** ใช้ limit ที่ aggressive (1 tick ใน spread) เพื่อ fill เร็ว

Market Order — ใช้เฉพาะกรณีฉุกเฉิน

- **ข้อดี:** Guarantee fill — ปิด position ได้แน่นอน
- **ข้อเสีย:** Slippage สูง โดยเฉพาะใน thin market เช่น Lighter
- **เมื่อใช้:** Emergency close เมื่อ Leg B ล้มเหลว เท่านั้น

14.6 Pseudo-code: cross_venue_execute()

def cross_venue_execute(signal, size, max_leg_wait_ms=500): if signal == "LONG_A_SHORT_B": order_a = place_limit_order(venue="Bybit", side="BUY", size=size) order_b = place_limit_order(venue="Lighter", side="SELL", size=size) filled_a = wait_fill(order_a, timeout_ms=max_leg_wait_ms) if not filled_a: cancel(order_a) return "LEG_FAIL_A" filled_b = wait_fill(order_b, timeout_ms=max_leg_wait_ms) if not filled_b: cancel(order_b) # emergency: close leg A immediately with market order market_close(venue="Bybit", side="SELL", size=filled_a.qty) return "LEG_FAIL_B_CLOSED_A" return "SUCCESS", filled_a, filled_b

อธิบาย Logic แต่ละส่วน

- `place_limit_order` ทั้งสองบรรทัดควร call พร้อมกัน (async/thread) ไม่ใช่ sequential
- `wait_fill(order_a, timeout_ms=500)` — block ด้วย timeout 500ms
- ถ้า A ไม่ fill: cancel แล้วออก — ไม่มี exposure เลย
- ถ้า A fill แต่ B ไม่ fill: **ต้อง market close A ทันที** อย่ารอให้ราคาขยับ
- Return tuple (SUCCESS, filled_a, filled_b) สำหรับ logging และ monitor

สิ่งที่ต้อง Log ทุก Execution

timestamp, signal_value, venue_a_fill_price, venue_b_fill_price, venue_a_latency_ms, venue_b_latency_ms, result, slippage_a, slippage_b

Log นี้ใช้ analyze เพื่อ tune timeout และตรวจจับ venue degradation

14.7 Running Example — Bybit↔Lighter Execution Timing

Running Example: Bybit BTC Perp ↔ Lighter BTC Perp

สถานการณ์: z-score ของ spread = 2.3 → สัญญาณ Long Bybit, Short Lighter

- Bybit order placement → confirmation latency: **45ms**
- Lighter order placement → confirmation latency: **120ms**

- รวม sequence ที่ดีที่สุด (simultaneous): **max(45, 120) = 120ms**
- รวม sequence ถ้า sequential: **45 + 120 = 165ms**

Timeline (simultaneous): t=0ms : ส่ง order Bybit + order Lighter พร้อมกัน t=45ms : Bybit fill ✓ @ $65,023.5 t=120ms : Lighter fill ✓ @ $64,998.2 t=120ms : BOTH FILLED — spread locked = $25.3 (ε = log(65023.5) - 1.003 * log(64998.2) = 0.0389%) ถ้า target exit ε = 0.00% → expected profit = 0.0389% × $100,000 = $38.9 ก่อน fee

Slippage จริง vs คาด: ถ้า signal คำนวณที่ mid price แต่ fill ที่ ask/bid → slippage ~0.5 tick ต่อข้าง ต้องคิดในสมการ expected profit

14.8 แบบฝึกหัด

โจทย์ 14.1 — คำนวณ Leg Risk Window

Bybit latency = 60ms, Lighter latency = 150ms ใช้ sequential execution (ส่ง Bybit ก่อน แล้วส่ง Lighter หลัง Bybit fill)

ถาม: (ก) Leg risk window กว้างแค่ไหน? (ข) ถ้าใช้ simultaneous execution แทน risk window เหลือเท่าไร?

▶ ดูคำตอบ

โจทย์ 14.2 — Leg B ล้มเหลว สูญเสียเท่าไร?

Long Bybit $200,000 fill ที่ $65,000 แต่ Short Lighter ไม่ fill
Emergency close: market sell Bybit ที่ $64,700 (BTC ร่วง 0.46% ระหว่างรอ)
Bybit taker fee = 0.05%

ถาม: ขาดทุนรวม (รวม fee) จากเหตุการณ์นี้เท่าไร?

▶ ดูคำตอบ

โจทย์ 14.3 — เลือก Timeout ที่เหมาะสม

จากข้อมูล historical: Lighter fill time distribution = median 80ms, P95 = 320ms, P99 = 680ms
ถ้าตั้ง timeout = 300ms จะ reject กี่% ของ trades? ถ้าตั้ง timeout = 600ms เพิ่มขึ้นอีก?

ถาม: ควรตั้ง timeout เท่าไรและทำไม?

▶ ดูคำตอบ

Ch.14 Cross-venue Strategy | ต่อไป → **Ch.15: Dynamic Hedging ด้วย Kalman Filter**

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 นี่คือ method หลักของ **Funding Rate Harvest strategy** (ดู บทที่ 4 §4.5) ที่ต้องการ $\varepsilon_t \approx 0$ ตลอดเวลา
State: $\beta_t = \beta_{t-1} + w_t$ | Observation: $r_{A,t} = \beta_t \cdot r_{B,t} + v_t$

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: $\beta_t = \beta_{t-1} + w_t$, $w_t \sim N(0, Q)$
Observation equation: $r_{A,t} = \beta_t \cdot r_{B,t} + v_t$, $v_t \sim N(0, R)$

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

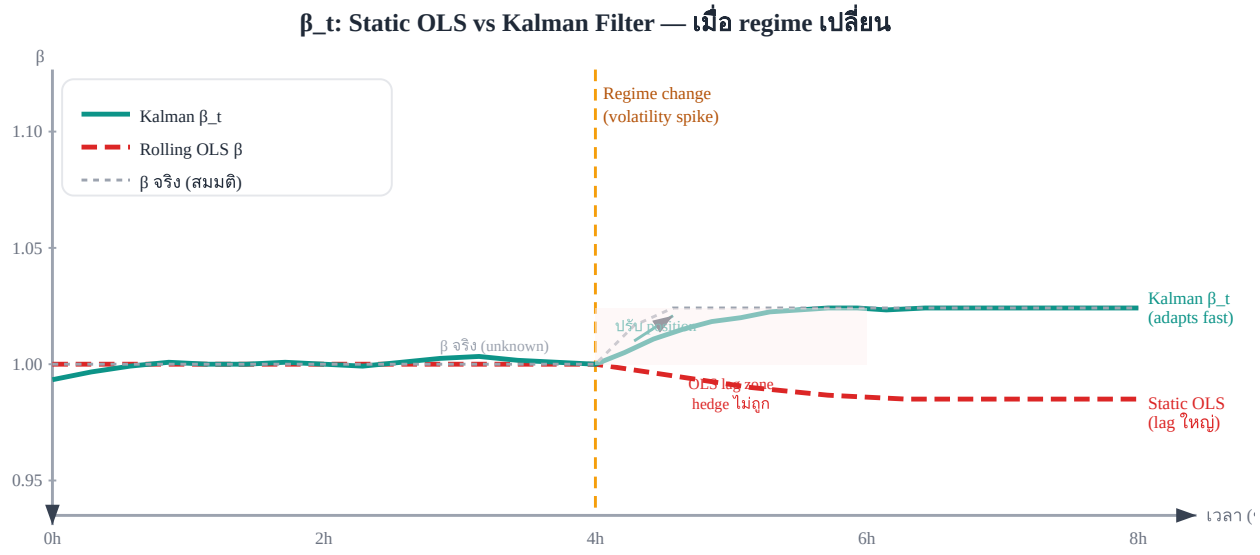
ขั้นตอน Update (เมื่อเห็น r_A,t และ r_B,t)
innovation = r_{A,t} - β̂_{t|t-1} · r_{B,t} (ความต่างจากที่คาด) S = r_{B,t}² · P_{t|t-1} + R (variance ของ innovation) K_t = P_{t|t-1} · r_{B,t} / S (Kalman Gain) β̂_{t|t} = β̂_{t|t-1} + K_t · innovation (update β) P_{t|t} = (1 - K_t · r_{B,t}) · P_{t|t-1} (update 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 # ค่า

เริ่มต้นจาก 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
delta_position_B = (β_new − β_old) × 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

Ch.15 Dynamic Hedging ด้วย Kalman Filter | ต่อไป → **Ch.16: Multi-leg Portfolios**

Statistical Arbitrage · บทที่ 16

Multi-leg Portfolios: บริหาร Portfolio ของ Spreads

หลาย spread พร้อมกัน — จัดการ correlation และ allocate capital อย่างมีประสิทธิภาพ
ตัวอย่าง: BTC Bybit↔Lighter + ETH Bybit↔Lighter + BTC Basis
แก่นของบทนี้ — อ่านก่อน
การรัน spread เดียวมีความเสี่ยงจาก idiosyncratic event ของ pair นั้น — portfolio ของหลาย spread ลด risk แต่ต้องระวัง **hidden correlation** ที่เพิ่มขึ้นในช่วงวิกฤต
$w^* = (1/2\lambda)\ \Sigma^{-1}\ \mu$ ← mean-variance optimal weights

16.1 ทำไมต้องหลาย Spread

การ diversify ข้าม spread ให้ประโยชน์หลัก:

ประโยชน์

- ลด variance ของ P&L รวม — spread แต่ละคู่ไม่ correlated 100%
- Deploy capital ต่อเนื่อง — ถ้า spread หนึ่ง "ไม่มีสัญญาณ" spread อื่นอาจมี
- ลด dependency บน pair เดียว — ถ้า exchange หนึ่ง maintenance อีก strategy ยังรัน

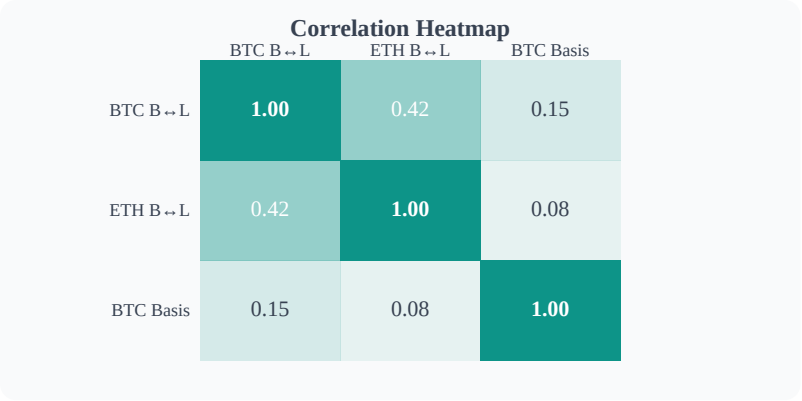
ความเสี่ยง

- Hidden correlation — BTC dump → ทุก crypto spread volatile พร้อมกัน
- Capital lock-up — ต้องการ margin สำหรับแต่ละ spread แยกกัน
- Operational complexity — monitor หลาย pair พร้อมกัน

16.2 Covariance Matrix ของ Spreads

นิยาม Σ = covariance matrix ขนาด n×n (n = จำนวน spread):

$\Sigma_{ij} = Cov(\varepsilon_i, \varepsilon_j) = E[(\varepsilon_i - \mu_i)(\varepsilon_j - \mu_j)]$ ตัวอย่าง 3 spreads: BTC_B↔L ETH_B↔L BTC_Basis BTC_B↔L [1.00 0.42 0.15] ← variance normalized ETH_B↔L [0.42 1.00 0.08] BTC_Basis[0.15 0.08 1.00]



Correlation heatmap — สี่เหลี่ยมเข้มหมายถึง correlation สูง (ไม่ดีสำหรับ diversification)

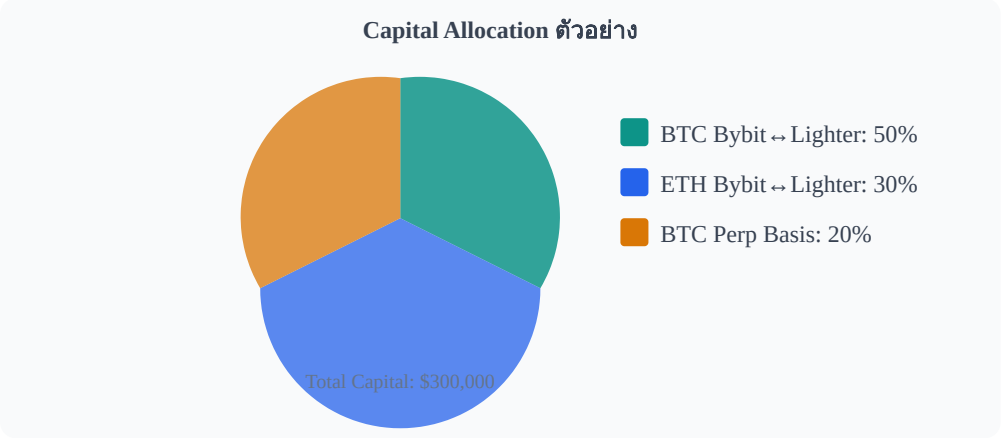
16.3 Mean-Variance Allocation

เป้าหมาย: หา weight vector **w** ที่ maximize risk-adjusted return:

max w'μ − λ × w'Σw w subject to: Σ w_i = 1, w_i ≥ 0, w_i ≤ w_max Solution (unconstrained): w* = (1/2λ) × Σ^{-1} × μ โดย: μ = vector ของ expected return แต่ละ spread Σ = covariance matrix λ = risk aversion parameter (ยิ่งสูง = conservative มากขึ้น) w* = optimal weights
ข้อควรระวัง

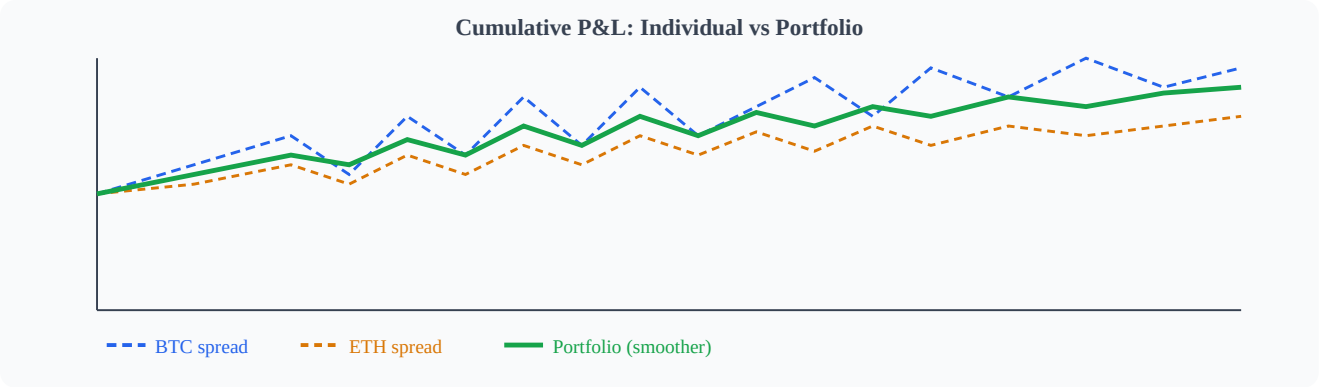
Mean-variance optimization **sensitive มากต่อ estimate ของ μ** — error เล็กน้อยใน μ ทำให้ w* เปลี่ยนแปลงมาก วิธีแก้: ใช้ shrinkage (เช่น Ledoit-Wolf) หรือ equal-weight เป็น baseline แล้วปรับ deviation ตาม conviction เท่านั้น

16.4 Portfolio Weight — Pie Chart



ตัวอย่าง capital allocation — BTC spread ได้ weight มากสุดเพราะ Sharpe สูงสุดและ risk ต่ำสุด

16.5 Drawdown Comparison



Portfolio ผสม (เขียว) มี drawdown น้อยกว่าและ P&L smoother กว่า individual spread

16.6 Pseudo-code portfolio_allocate()

def portfolio_allocate(expected_returns, cov_matrix, risk_aversion=1.0, max_weight=0.6, min_weight=0.05): """ expected_returns: vector of μ_i (annualized expected return) cov_matrix: Σ matrix (n×n) risk_aversion: λ parameter (higher = more conservative) max_weight: cap ต่อ strategy เดียว min_weight: minimum allocation ถ้า include strategy นั้น """ n = len(expected_returns) # Inverse covariance (Ledoit-Wolf shrinkage recommended) inv_cov = ledoit_wolf_inverse(cov_matrix) # Raw MV optimal weights raw_weights = inv_cov @ expected_returns / (2 * risk_aversion) # Clip to [0, max_weight] weights = clip(raw_weights, 0, max_weight) # Zero out strategies below min_weight threshold weights = [w if w >= min_weight else 0.0 for w in weights] # Normalize to sum = 1 total = sum(weights) if total > 0: weights = [w / total for w in weights] return weights # list of n weights summing to 1.0

16.7 Correlation Risk ในช่วงวิกฤต

Hidden Correlation Problem

ในช่วงตลาดปกติ BTC spread กับ ETH spread correlate 0.42 — แต่ในช่วง BTC dump correlation อาจพุ่งเป็น 0.85+ เพราะทั้งสองตลาดถูก panic ขายพร้อมกัน

นี่คือ "correlation breakdown" — assumption ของ portfolio optimization ล้มเหลวตอนที่ต้องการ protection มากที่สุด

def detect_correlation_spike(spread_returns, window=30, threshold=0.70): """ตรวจว่า rolling correlation ระหว่าง spreads สูงขึ้นผิดปกติ""" rolling_corr = compute_rolling_correlation(spread_returns, window) max_off_diagonal = max_corr_excluding_diagonal(rolling_corr) if max_off_diagonal > threshold: return True, f"Correlation spike: {max_off_diagonal:.2f}" return False, None

16.8 Running Example — 3 Spread Portfolio

🔁 Running Example: Portfolio ของ 3 Spreads

Strategy	Weight	Capital	Expected Sharpe	σ ต่อปี
BTC Bybit↔Lighter	50%	$150,000	2.1	12%
ETH Bybit↔Lighter	30%	$90,000	1.7	15%
BTC Perp Basis	20%	$60,000	1.4	8%
Portfolio	**100%**	**$300,000**	**2.3**	**10%**

สังเกต: Portfolio Sharpe (2.3) สูงกว่า Sharpe ของแต่ละ strategy เดี่ยว — เป็นผลจาก diversification benefit เมื่อ correlation ไม่เท่ากับ 1

16.9 แบบฝึกหัด

แบบฝึกหัดบทที่ 16

▶ ข้อ 1: ถ้า BTC dump ทำให้ correlation ระหว่าง BTC และ ETH spread เพิ่มเป็น 0.90 — ควรทำอย่างไร?

▶ ข้อ 2: ทำไม equal-weight portfolio มักดีกว่า MV optimal ในทางปฏิบัติ?

▶ ข้อ 3: เพิ่ม spread ที่ 4 เข้า portfolio — ควรตรวจสิ่งใดก่อน?

Statistical Arbitrage · บทที่ 17

Commodity Basis: Calendar Spread ในตลาด Futures

Front − Back futures basis มาจาก storage cost + convenience yield
ทำความเข้าใจโครงสร้างราคา Futures ก่อนที่จะ trade calendar spread
แก่นของบทนี้ — อ่านก่อน
Calendar spread คือ basis ระหว่าง futures สองตัวต่างวัน expiry ของ asset เดียวกัน ค่านี้ไม่ใช่ random — มันถูกกำหนดโดยต้นทุนการเก็บรักษาและ convenience yield ของ commodity นั้น เมื่อ basis เบี่ยงออกจาก fair value ก็เกิดโอกาส arbitrage
Basis ≈ (storage_cost − convenience_yield) × $(T_2 - T_1)$ / 365

17.1 Calendar Spread คืออะไร

Calendar spread คือการ long futures ที่ expiry เร็ว (front month) และ short futures ที่ expiry ช้า (back month) ของ underlying เดียวกัน หรือกลับกัน โดยที่เราไม่ได้ bet ทิศทางของราคา underlying แต่ bet ว่า **ความต่างระหว่างสอง expiry** จะกลับสู่ fair value

ตัวอย่างพื้นฐาน: Crude Oil Calendar Spread

CL June futures = $80.00 | CL September futures = $82.50
Calendar spread = Front − Back = $80.00 − $82.50 = **−$2.50**
Market คาดว่า oil ใน Sep จะแพงกว่า Jun $2.50 → ตลาดอยู่ใน **Contango**

เปรียบเทียบ: Calendar Spread vs. Outright

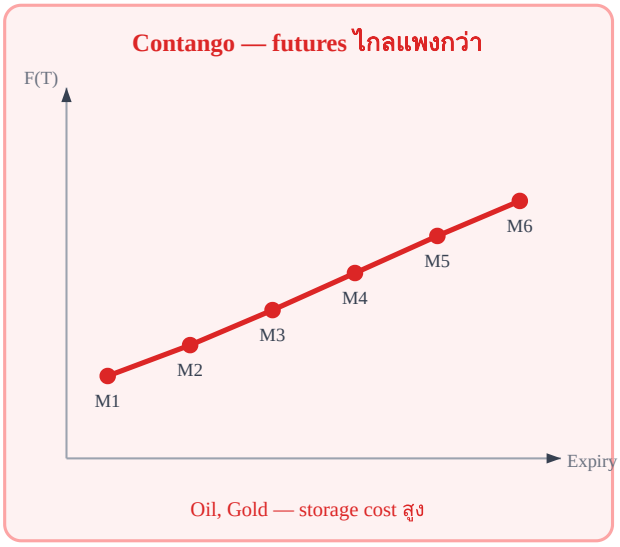
Outright long CL: P&L ขึ้นตรงกับราคา oil — เสี่ยง directional risk เต็มๆ
Calendar Spread: P&L ขึ้นกับ *ความต่าง* ระหว่าง expiry — delta ต่อ oil เกือบ zero, เสี่ยงน้อยกว่ามาก

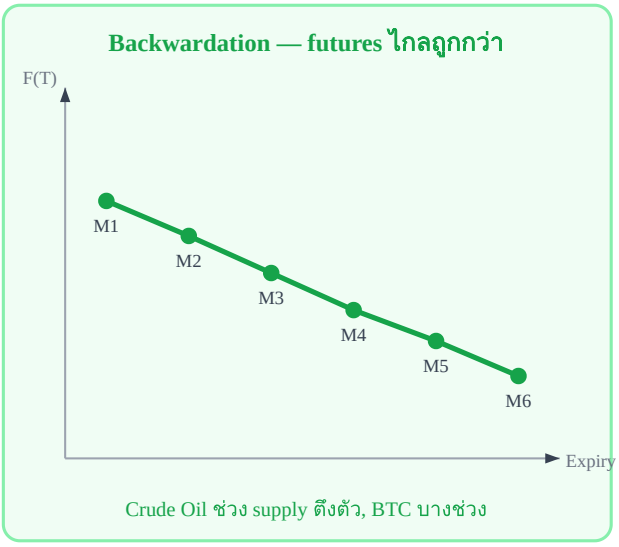
ประเภท Position	Long Front / Short Back	Short Front / Long Back
ชื่อเรียก	Long Calendar Spread	Short Calendar Spread
กำไรเมื่อ	Contango ลดลง (spread แคบลง)	Contango เพิ่มขึ้น (spread กว้างขึ้น)
สถานการณ์ที่ดี	อุปสงค์ระยะสั้นพุ่งสูง, storage ตึงตัว	supply ระยะสั้นล้นตลาด
Risk หลัก	Contango ขยายมากกว่าที่คาด	Market พลิกเป็น Backwardation

17.2 Contango vs. Backwardation

โครงสร้าง futures term structure บอกว่า market คาดการณ์ราคาในอนาคตอย่างไร:

- **Contango:** F(T_back) > F(T_front) — futures ไกลแพงกว่า futures ใกล้ — ปกติสำหรับ commodity ที่มีต้นทุนเก็บรักษาสูง (oil, grain, metals)
- **Backwardation:** F(T_back) < F(T_front) — futures ไกลถูกกว่า — เกิดเมื่อ convenience yield สูง หรืออุปสงค์ระยะสั้นตึงตัวมาก
- **Flat:** ราคาเท่ากันทุก expiry — หายาก มักเกิดช่วงผ่านระหว่าง Contango ↔ Backwardation





Contango (ซ้าย): term structure ลาดขึ้น | Backwardation (ขวา): term structure ลาดลง

17.3 Fair Basis Formula — Cost of Carry Model

ราคา futures ที่ "fair" ต้องสะท้อน cost of carry ครบถ้วน มิฉะนั้นจะมี cash-and-carry arbitrage:

สูตร Cost of Carry
$F(T) = S \times e^{(r + u - q) \times T}$

โดยที่: **S** = ราคา spot, **r** = risk-free rate, **u** = storage cost (% ต่อปี), **q** = convenience yield (% ต่อปี), **T** = เวลาถึง expiry (ปี)
Calendar Spread Fair = $F(T_{back}) - F(T_{front}) = F_{front} \times (e^{(r+u-q)\cdot\Delta T} - 1)$ โดย $\Delta T = T_{back} - T_{front}$

เรานิยาม calendar spread = **Back-month − Front-month** เสมอ ภายใต้ contango ($r+u-q > 0$) ค่านี้จะ**เป็นบวก** สอดคล้องกับ key-idea ที่ว่า basis ≈ (storage − convenience) × $\Delta T/365$ สูตรนี้คือสูตรเดียวกับที่ใช้ในตัวอย่าง §17.4 และแบบฝึกหัด §17.7

เมื่อ basis เบี่ยงจาก fair value เกิดโอกาส arbitrage:

ถ้า $F(T_2) - F(T_1) > fair_basis$ → contango แพงเกินไป → Long $F(T_1)$, Short $F(T_2)$ → wait for convergence ถ้า $F(T_2) - F(T_1) < fair_basis$ → backwardation/contango น้อยเกินไป → Short $F(T_1)$, Long $F(T_2)$ → wait for convergence

Parameter	Crude Oil (ตัวอย่าง)	Gold (ตัวอย่าง)	BTC Futures
r (risk-free)	5.25% / ปี	5.25% / ปี	5.25% / ปี
u (storage)	0.5–1.5% / ปี	0.1–0.3% / ปี	~0% (digital)
q (convenience)	2–10% / ปี (volatile)	0.5–1% / ปี	0% (ไม่มี utility)
ลักษณะ	Contango หรือ Back ได้ทั้งคู่ มักเป็น Contango เสมอ ขึ้นกับ funding rate		

17.4 ตัวอย่าง: Crude Oil Calendar Spread

Running Example: CL (Crude Oil) Calendar Spread

ข้อมูล ณ วันที่วิเคราะห์:

- CL Front (July) = **$80.00**/barrel
- CL Back (September) = **$82.00**/barrel
- Observed spread (Back − Front) = +$2.00 (Contango $2)

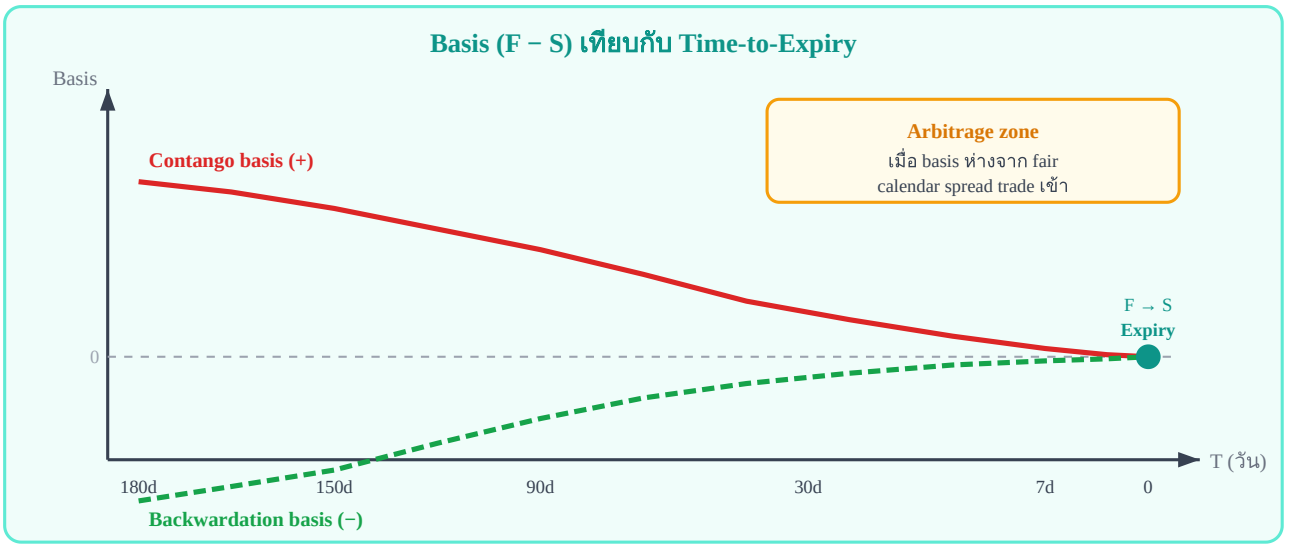
คำนวณ fair basis สำหรับ 60 วัน ($T = 60/365 \approx 0.164$ ปี):

$r = 5.25\%$, $u = 1.0\%$, $q = 3.0\%$ → $(r + u - q) = 3.25\%$ Fair F(back) / Fair F(front) = $e^{(r+u-q) \times (T_{back} - T_{front})} = e^{0.0325 \times (60/365)} = e^{0.00534} \approx 1.00535$ Fair spread = F(front) × 0.00535 = $80 × 0.00535 = $0.428 Observed spread = $2.00 vs Fair = $0.43 → Contango แพงเกินไปมาก ($1.57 excess) → Trade: Long Front (July) + Short Back (Sep)
ความเสี่ยงที่ต้องระวัง

- **Convenience yield เปลี่ยนแปลงกะทันหัน:** ข่าว OPEC, supply disruption → Backwardation พุ่งแรง
- **Delivery risk:** futures ใกล้ expiry ต้อง physical delivery หรือ roll ออก
- **Roll cost:** ถ้า contango กว้าง การ roll position มีต้นทุนสูง

17.5 Basis vs. Time-to-Expiry

เมื่อ expiry ใกล้เข้ามา basis ของ futures จะต้อง converge สู่ 0 (เนื่องจาก $F(T \to 0) = S$)



Basis ทุกประเภทต้อง converge สู่ 0 เมื่อถึง expiry — นี่คือ "pull to par" ของ futures

17.6 เทียบกับ Crypto Perpetual — Concept เดียวกัน แต่ต่างกันตรงไหน?

Crypto perpetual futures ใช้ concept เดียวกับ calendar spread แต่มีความแตกต่างสำคัญ:

ลักษณะ	Commodity Calendar Spread	Crypto Perpetual Basis
Expiry	มี expiry — convergence มีวันกำหนด	ไม่มี expiry — ไม่มี forced convergence
Mechanism ดึงกลับ	Cost of carry + expiry convergence	Funding rate (ทุก 8h หรือ 1h)
Basis = ?	$F(T_2) - F(T_1)$ vs cost of carry	Perp − Spot vs funding rate integral
Risk	Delivery, roll cost, supply shock	Funding ขาด liquidity, exchange risk
Calendar spread ใน crypto	ทำได้กับ quarterly futures	Perp vs. quarterly = "basis trade"

BTC Basis Trade: Perp vs. Quarterly Future

BTC Perp = $65,000 | BTC Sep Quarterly = $66,200
Observed basis = +$1,200 (quarterly แพงกว่า 1.85%)
Annualized funding rate implied = 1.85% × (365/90) ≈ **7.5% / ปี**
ถ้า funding rate จริงต่ำกว่า 7.5% → Quarterly แพงเกินไป → **Long Perp + Short Quarterly**

17.7 แบบฝึกหัด

โจทย์ 17.1 — คำนวณ Fair Calendar Spread

Natural Gas (NG) futures: Front (August) = $2.80/MMBtu, Back (November) = $3.20/MMBtu
r = 5%, storage cost u = 8% ต่อปี, convenience yield q = 2% ต่อปี, T_back − T_front = 90 วัน

ถาม: Fair spread คือเท่าไร? Basis ปัจจุบัน (observed) เป็น Contango มากเกินไปหรือน้อยเกินไป?

▶ ดูคำตอบ

โจทย์ 17.2 — Crypto Basis Comparison

BTC Perpetual = $67,500 | BTC December Quarterly Futures = $68,900
เวลาถึง expiry Dec = 120 วัน | Risk-free rate = 5% ต่อปี

ถาม: Implied annualized basis คือเท่าไร? ถ้า funding rate เฉลี่ยที่คาด = 10% ต่อปี ควร trade อย่างไร?

▶ ดูคำตอบ

Ch.17 Commodity Basis | ต่อไป → **Ch.18: Options Overlay — Put-Call Parity Arbitrage**

Statistical Arbitrage · บทที่ 18

Options Overlay: Put-Call Parity Arbitrage

ε = Call − Put − (F−K)e^{−rT} — deviation คือโอกาส arbitrage
ค้นหาและ exploit ความไม่สมดุลใน options market อย่างเป็นระบบ
แก่นของบทนี้ — อ่านก่อน
Put-Call Parity เป็นกฎพื้นฐานที่ต้องเป็นจริงในตลาดที่ไม่มี arbitrage เมื่อ ε ≠ 0 แสดงว่า call หรือ put ราคาผิดสัมพัทธ์กัน และสามารถทำ riskless profit ได้โดยการ construct portfolio ที่สอดคล้อง
ε = C − P − (F − K) × e^{−rT}

18.1 Put-Call Parity — สูตรและที่มา

Put-Call Parity เป็น no-arbitrage relationship ที่เชื่อม call, put, forward, และ strike เข้าด้วยกัน:

Put-Call Parity สำหรับ European Options
สำหรับ European options บน futures contract F ที่ strike K expiry T:
C − P = (F − K) × e^{−rT}

สำหรับ European options บน spot asset S:
C − P = S × e^{−qT} − K × e^{−rT}

โดยที่: C = call price, P = put price, r = risk-free rate, q = dividend/funding yield

ที่มาของสูตร: พิจารณา 2 portfolio ที่ให้ payoff เดียวกัน:

Portfolio	ส่วนประกอบ	Payoff เมื่อ F_T > K	Payoff เมื่อ F_T ≤ K
A: Long Call	Buy C, Invest (F−K)e^{−rT}	(F_T − K) + (F−K)	0 + (F−K)
B: Long Fwd + Put	Long F at K, Buy P	(F_T − K) + 0	0 + (K − F_T)

เนื่องจาก payoff เท่ากัน ราคาต้องเท่ากัน → C − P = (F − K)e^{−rT} ต้องเป็นจริงเสมอ

18.2 Parity Deviation ε — เมื่อไรจึง Arbitrage ได้

นิยาม Parity Deviation
ε = C − P − (F − K) × e^{−rT}

ถ้า **ε > 0**: Call แพงเกินไปเทียบกับ put → Sell call, Buy put, Buy forward
ถ้า **ε < 0**: Put แพงเกินไปเทียบกับ call → Buy call, Sell put, Sell forward
ถ้า **ε = 0**: parity สมบูรณ์ — ไม่มีโอกาส arbitrage
เงื่อนไขที่ ε ≠ 0 ในทางปฏิบัติ

- **Bid-Ask spread:** ε ต้องเกิน bid-ask cost รวมทุก leg ก่อนจะ profitable
- **Margin/collateral cost:** ต้องใช้ margin วาง → มีต้นทุน opportunity
- **Execution risk:** ต้อง fill ทุก leg พร้อมกัน ไม่เช่นนั้นเป็น legged risk
- **American vs European:** American options มี early exercise premium → parity แตกต่าง

18.3 BTC Options ตัวอย่าง — Deribit Put-Call Parity

Running Example: BTC Options บน Deribit

ข้อมูลจาก Deribit สำหรับ BTC-28JUN-65000 (Strike K = $65,000, expiry 30 วัน):

- BTC Futures (28JUN) = **$66,200**
- Call (K=65000, T=30d) = **$2,850**
- Put (K=65000, T=30d) = **$1,520**
- Risk-free rate r = 5% ต่อปี → $e^{-rT} = e^{-0.05\times30/365} = 0.9959$

Theoretical C - P = (F - K) × e^{-rT} = ($66,200 - $65,000) × 0.9959 = $1,200 × 0.9959 = $1,195.08
Observed C - P = $2,850 - $1,520 = $1,330 ε = $1,330 - $1,195 = +$135 → Call แพงเกินไปเทียบกับ Put Trade: Sell Call + Buy Put + Long Forward (at K=$65,000) Expected profit ≈ $135 (ก่อนหักค่าธรรมเนียม)
ตรวจสอบต้นทุนก่อน Trade

Deribit taker fee ≈ 0.03% ต่อ option position (per leg)
3 legs: Sell C + Buy P + Long F → total fee ≈ 0.09% × notional
Notional ≈ $65,000 → fee ≈ $58.50
Net profit ≈ $135 − $58.50 = **$76.50** ต่อ BTC → ยังคุ้ม แต่ต้องระวัง execution
หมายเหตุ: conversion เป็น delta-flat โดยโครงสร้าง (ดู §18.6) จึงไม่ต้องมี hedge leg แยกต่างหาก

18.4 IV Surface Arbitrage — Calendar Spread ใน IV

นอกจาก put-call parity แล้ว ยังมีโอกาส arbitrage จากความไม่สอดคล้องของ Implied Volatility (IV) ข้าม expiry:

ประเภทของ IV Surface Arbitrage

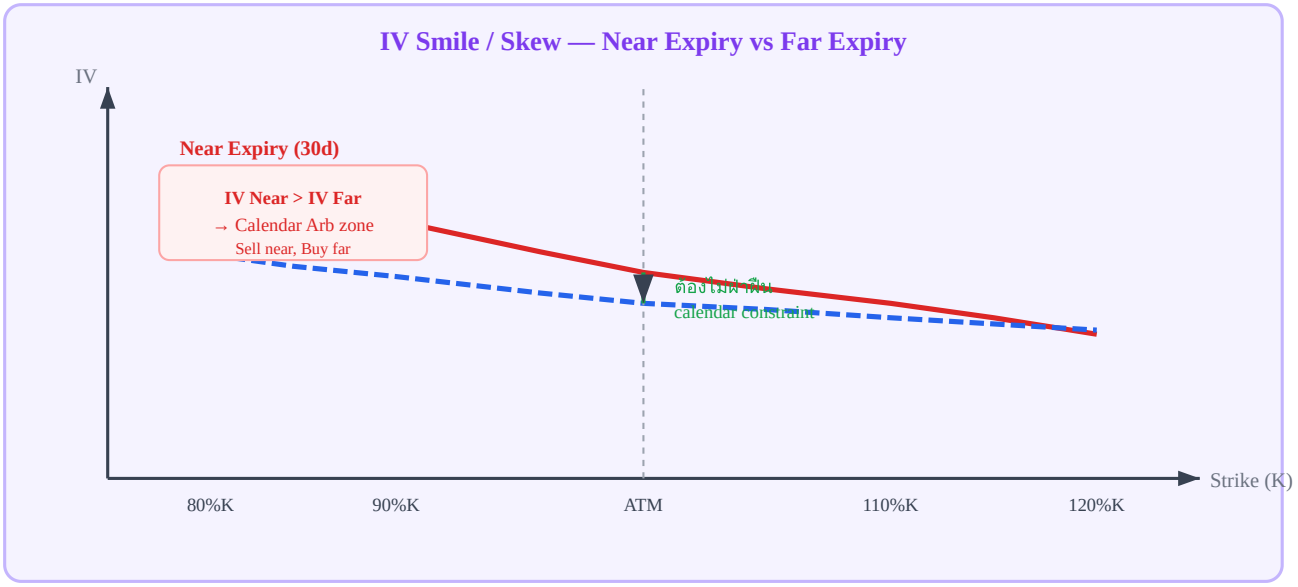
- **Calendar Spread IV Arb:** IV ของ near-expiry สูงกว่า far-expiry ผิดปกติ → sell near, buy far vega
- **Strike Spread (Skew Arb):** IV skew ระหว่าง OTM put และ OTM call ผิดปกติ
- **Butterfly Arb:** IV ของ ATM ผิดจาก weighted average ของ OTM wings

No-Arbitrage Constraints ของ IV Surface

Constraint	สูตร / เงื่อนไข	ถ้าฝ่าฝืน
Calendar spread	Total variance $V(T) = IV^2 \times T$ ต้องเพิ่มตาม T	Sell near, buy far → calendar arb
Butterfly	$\partial^2 C/\partial K^2 \geq 0$ (call spread เป็น convex)	Butterfly spread ให้ riskless profit
Call spread	$\partial C/\partial K \leq 0$ (call price ลดเมื่อ K ขึ้น)	Vertical spread arb
Horizontal bound	$C(K,T_2) \geq C(K,T_1)$ สำหรับ $T_2 > T_1$	Calendar spread arb

18.5 IV Surface Diagram (2D Slice)

แผนภาพ 2D slice ของ IV surface ที่ expiry คงที่ (IV vs. Strike):



Near expiry IV (แดง) ต้องไม่สูงกว่า far expiry IV อย่างผิดปกติที่ strike เดียวกัน มิฉะนั้นเป็น calendar arb

18.6 Delta Hedge ของ Options Spread

เมื่อ trade IV surface arb (เช่น calendar spread ใน IV) ต้องระวังว่า position มี **delta exposure** ที่ต้อง hedge ออก ส่วน conversion (put-call parity) นั้น delta-flat โดยโครงสร้างอยู่แล้ว — ดูตัวอย่างด้านล่าง:

ขั้นตอน Delta Hedge

1. คำนวณ net delta ของ options portfolio: $\Delta_net = \Delta_call - \Delta_put + \Delta_forward$
2. Sell/Buy futures ในปริมาณ Δ_net เพื่อให้ net delta ≈ 0
3. Re-hedge เมื่อ delta drift (เนื่องจาก gamma effect) หรือทุก interval ที่กำหนด
4. ระวัง gamma risk: position ที่ delta-neutral อาจมี gamma ที่ทำให้ P&L เป็น non-linear

ตัวอย่าง: ε trade บน BTC K=65000, T=30d Long Put: Δ_put = −0.42 → contribution = +1×(−0.42) = −0.42 Short Call: Δ_call = +0.58 → contribution = −1×(+0.58) = −0.58 Long Fwd: Δ_fwd = +1.00 → contribution = +1×(+1.00) = +1.00 Net Δ = −0.42 − 0.58 + 1.00 ≈ 0.00 → conversion เป็น delta-neutral โดยอัตโนมัติ ไม่ต้อง hedge เพิ่ม
Re-hedge Frequency

สำหรับ crypto options ที่ volatile มาก: re-hedge ทุก 30–60 นาที หรือทุกครั้งที่ delta drift เกิน ±0.05
Transaction cost ของ re-hedge ต้องรวมใน edge calculation (บทที่ 19)

18.7 แบบฝึกหัด

โจทย์ 18.1 — คำนวณ ε และ Direction

ETH Options บน Deribit: Strike K = $3,500, Expiry = 45 วัน
ETH Futures (45d) = $3,620 | Call = $285 | Put = $148 | r = 5% ต่อปี

ถาม: คำนวณ ε และระบุ direction ของ trade arbitrage

▶ ดูคำตอบ
โจทย์ 18.2 — IV Calendar Constraint

BTC options ที่ K = ATM: IV_30d = 72%, IV_90d = 58%
Total variance: $V_{30} = 0.72^2 \times (30/365) = 0.0426$, $V_{90} = 0.58^2 \times (90/365) = 0.0832$

ถาม: IV surface นี้ violate calendar constraint หรือไม่? ถ้า violate ควร trade อย่างไร?

▶ ดูคำตอบ

18.8 PCP Violation เป็น ε — เชื่อมสู่ Stat Arb Framework

ตลอด §18.1–18.7 เราใช้ PCP เพื่อ *สร้าง* synthetic position — แต่มีมุมมองที่ทรงพลังกว่า: **PCP violation คือ ε ในรูปแบบหนึ่ง** ที่สามารถนำเข้า stat arb framework ทั้งเล่มได้ทันที

Key Idea — PCP Violation = ε ชนิดใหม่
$\varepsilon_{PCP} = (C - P) - (S - K \cdot e^{-rT})$

PCP บอกว่า ε นี้ควร = 0 ในตลาดไร้ friction
ในตลาดจริง $\varepsilon_{PCP} \neq 0$ เล็กน้อยและ mean-revert รอบ 0

โครงสร้างนี้เหมือนกับ $\varepsilon = \log(P_A) - \beta \cdot \log(P_B)$ ใน บทที่ 3 ทุกประการ — เราสร้าง synthetic process ที่ควรเป็น 0 แล้วเทรดเมื่อมันเบี่ยงออก ความแตกต่างคือ ε_{PCP} มาจาก no-arbitrage constraint ของ PCP ไม่ใช่ cointegration ของราคา

ε_{PCP} mean-revert ได้ไหม?

ε_PCP ≠ 0 เกิดจาก:

- **Liquidity mismatch:** bid-ask spread ของ C และ P ไม่เท่ากัน
- **Borrow cost:** หายากสำหรับ short stock → put premium ขึ้น
- **Vol regime:** ช่วง crash IV_put เพิ่มเร็วกว่า IV_call → skew spike
- **Funding:** crypto options มี basis risk ระหว่าง underlying spot กับ perp

แต่ทั้งหมดนี้เป็น transient → ε_PCP mean-revert รอบ 0 ในระยะสั้น

	ε ราคา (ch3–ch14)	**ε_PCP (ch18.8)**
สร้างจาก	log(P_A) − β·log(P_B)	C − P − (S − K·e^{−rT})
ควรเป็น	ค่าคงที่ θ (mean)	0 (no-arbitrage)
เบี่ยงเพราะ	market inefficiency	liquidity/skew/funding
test ด้วย	ADF, half-life, z-score	เหมือนกันทุกอย่าง
trade	z-score entry/exit	เหมือนกัน

Running Example: BTC Options — ε_PCP จาก Deribit
ข้อมูล ณ วันที่ T (BTC ≈ $95,000): Call ATM (K=95,000, T=30d): C = $4,820 Put ATM (K=95,000, T=30d): P = $4,310 Spot BTC: S = $95,000 r = 5%/yr, T = 30/365 PCP fair value: S − K·e^{−rT} = 95,000 − 95,000·e^{−0.05×30/365} = 95,000 − 95,000·0.99590 = 95,000 − 94,610 = 390 ε_PCP = (C − P) − 390 = (4,820 − 4,310) − 390 = 510 − 390 = +120 → ε_PCP = +$120 → Call แพงกว่า Put ผิดปกติ → ถ้า ε_PCP mean-revert → เปิด: Short Call + Long Put + Long Stock (Conversion) รับกำไรเมื่อ ε_PCP กลับสู่ 0
ข้อควรระวัง — Options friction สูงกว่า Perp มาก

- Bid-ask ของ options กว้างกว่า perp 5–20x → breakeven ε_PCP สูงมาก
- American options (equity): early exercise ทำให้ PCP ไม่ strict — ใช้เฉพาะ European
- Crypto options (Deribit): European, ไม่มี dividend → PCP strict ✅
  หมายเหตุ: Deribit settle against futures reference index ไม่ใช่ spot → PCP ที่ถูกต้องคือ C−P = (F−K)·e^{−rT} (F คือ futures price) ความต่างจาก spot PCP ≈ basis × T ซึ่งเล็กมากสำหรับ T ≤ 30 วัน แต่ควรระวังถ้า basis กว้าง
- ต้องคำนวณ breakeven ε_PCP ≥ total_friction ก่อนเทรดเสมอ (ดู บทที่ 19)

บทที่ 22 — Options-Based Stat Arb เต็มรูปแบบ

บทที่ 22 ขยาย ε_PCP ไปสู่ IV Surface Stat Arb ครบชุด:

§22.1 IV series เป็น process ใหม่ · §22.2 Box Spread Arb
§22.3 IV Skew Stat Arb · §22.4 Cross-venue IV Arb (Deribit↔Bybit)
§22.5 Vol Risk Premium · §22.6 Friction · §22.7 Practical Setup

โจทย์ 18.8 — คำนวณ ε_PCP

Given: BTC C(K=100k, T=7d) = $2,400, P(K=100k, T=7d) = $2,350, S = $100,000, K = $100,000, r = 5%/yr

(ก) คำนวณ PCP fair value = S − K·e^{−rT}

(ข) คำนวณ ε_PCP

(ค) ถ้า ε_PCP = +$80 และ normal range ±$30 → ควรเปิด position ไหน?

▶ ดูคำตอบ

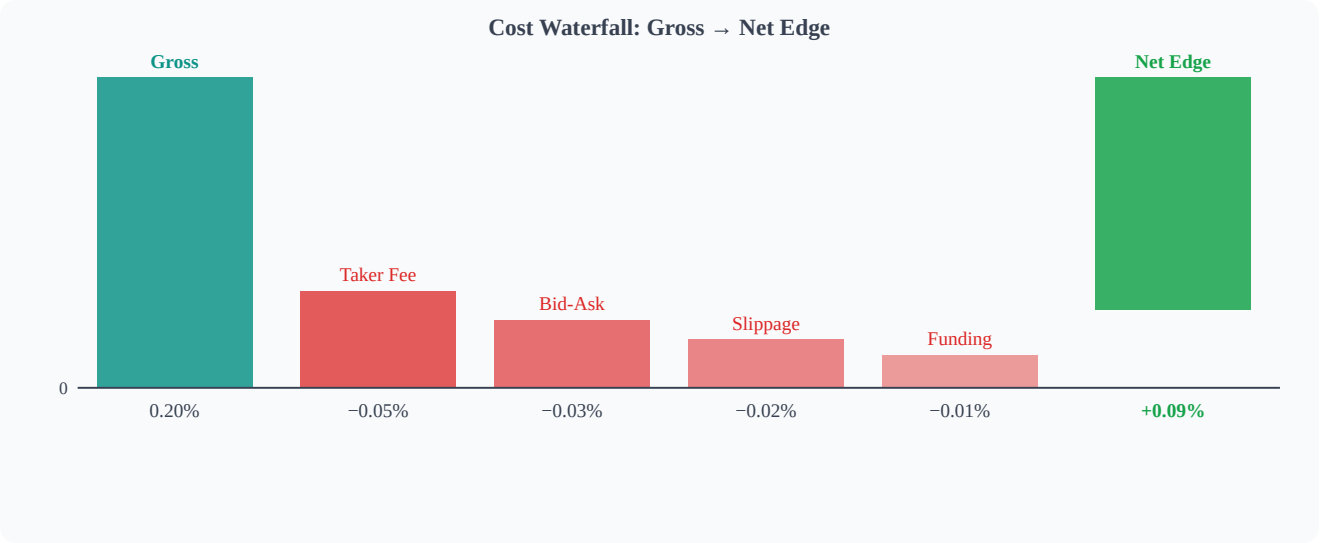
Ch.18 Options Overlay | ต่อไป → **Ch.19: Cost Analysis & Edge — คำนวณ Net Edge ก่อนเทรด** | §18.8 ดู ch22 สำหรับ Options Stat Arb เต็มรูปแบบ

Statistical Arbitrage · บทที่ 19

Cost Analysis & Edge: คำนวณ Net Edge ก่อนเทรด

Gross spread − fee − bid/ask − slippage − funding − legging buffer = Net edge
ถ้า net edge ≤ 0 → ไม่เทรด ไม่ว่า signal จะดูดีแค่ไหน
แก่นของบทนี้ — อ่านก่อน
Stat arb มีต้นทุนซ่อนอยู่หลายชั้น — signal ที่ดูมี edge 0.20% อาจมี net edge ลบหลัง cost หักออกทั้งหมด **คำนวณก่อนเสมอ — ห้าม assume**
Net Edge = Gross − Σ(fees + spread + slippage + funding + legging)

19.1 Edge Waterfall Diagram



Waterfall chart: แต่ละ cost component ลดลงจาก gross spread ของ 0.20% — เหลือ net edge เพียง 0.09%

19.2 Fee Structure

Exchange	Maker Fee	Taker Fee	หมายเหตุ
Bybit Perp	−0.025% (rebate)	+0.055%	VIP levels ลด taker ได้
Lighter Perp	−0.010%	+0.030%	clob-based, faster fills
Round trip cost (2 takers)			(0.055+0.030)×2 = 0.17%
Round trip cost (2 makers)			rebate: −0.07% (receive)

Maker vs Taker Strategy

ถ้า latency ไม่ critical → ใช้ limit order (maker) ทั้งคู่ — ลด cost จาก −0.17% เป็น −0.03% ต่อ round trip แต่ risk partial fill และ leg mismatch เพิ่มขึ้น

19.3 Bid-Ask Spread Cost

Cost per leg = (Ask − Bid) / 2 = half-spread สำหรับ 2 legs round trip: Total bid-ask cost = half-spread_A + half-spread_B ตัวอย่าง Bybit BTC: Bid = 65,000, Ask = 65,002 Half-spread = $1 = 1/65,000 = 0.0015% Lighter BTC: Bid = 64,998, Ask = 65,001 Half-spread = $1.50 = 0.0023% Total: 0.0015% + 0.0023% = 0.0038% ต่อ direction Round trip (entry + exit): 0.0076% ≈ 0.008%

19.4 Slippage Model

Market impact (slippage) เพิ่มขึ้นตาม size ที่เทียบกับ average daily volume:

Slippage ≈ σ_daily × sqrt(size / ADV) โดย: σ_daily = daily volatility ของ asset size = ขนาด order (notional) ADV = average daily volume (notional) ตัวอย่าง: σ_daily = 0.02 (2%) size = $100,000 ADV = $500,000,000 (Bybit BTC perp ~$500M/day) Slippage = 0.02 × sqrt(100,000/500,000,000) = 0.02 × 0.014 = 0.00028 = 0.028%

19.5 Funding Cost

Funding cost ต่อ 8h = |funding_rate| × notional Total funding cost = funding_rate × hold_periods × notional ตัวอย่าง: Funding rate = 0.01% ต่อ 8h (positive → longs จ่าย) Hold time = 2 funding periods = 16h Notional = $100,000 Cost (ถ้า short perp) = +0.01% × 2 × $100,000 = +$20 (รับ) Cost (ถ้า long perp) = −0.01% × 2 × $100,000 = −$20 (จ่าย)
Funding Direction ขึ้นกับ Position

ถ้า basis > 0 และ short perp → รับ funding (ลด cost)
ถ้า basis > 0 และ long perp → จ่าย funding (เพิ่ม cost)
ต้องคำนวณรวมในทุก scenario ก่อน decide trade direction

19.6 Legging Buffer

Legging risk = ความเสี่ยงจากช่วงเวลาระหว่าง fill leg A และ fill leg B ในช่วงนั้น position เป็น single-leg → exposed ต่อ market movement Legging buffer estimate: buffer = σ_spread × sqrt(avg_fill_latency_s / seconds_per_period) ตัวอย่าง: σ_spread = 0.05% per period avg_fill_latency = 0.5 วินาที seconds_per_period = 60 วินาที (1 นาที) buffer = 0.05% × sqrt(0.5/60) = 0.05% × 0.091 = 0.0045%

19.7 Break-even Analysis — Minimum Viable σ_stat

σ_stat ที่ใช้ในสมการด้านล่างมาจาก Objective-Driven Design ใน บทที่ 4.5 — กลยุทธ์ Hybrid ต้องการ σ_stat ที่ "Optimal Substantial" (ไม่ต่ำเกินจน edge หมด ไม่สูงเกินจน half-life ยาวเกิน) สมการ Breakeven ด้านล่างคือเส้นขอบล่างของช่วง tradeable นั้น

Minimum σ_stat เพื่อให้ trade คุ้มค่า: Break-even σ = total_cost / (z_entry − z_exit) ตัวอย่าง: total_cost = 0.11% z_entry = 2.0, z_exit = 0.5 σ_stat = z × σ_spread → spread ที่ entry = 2.0 × σ_spread Break-even: 0.11% / (2.0 − 0.5) = 0.073% → ต้องการ σ_spread ≥ 0.073% เพื่อให้ entry at z=2.0 cover cost → ถ้า σ_spread = 0.05% → entry spread = 0.10%, net edge = −0.01% (ไม่ trade!)

19.8 Pseudo-code compute_edge()

def compute_edge(spread_at_entry_pct, sigma_spread_pct, entry_z, exit_z, fee_a=0.00055, fee_b=0.00030, half_spread_a=0.00002, half_spread_b=0.00003, size=100_000, adv=500_000_000, sigma_daily=0.02, funding_per_8h=0.0001, hold_periods=1, avg_fill_latency_s=0.5, seconds_per_period=60): gross = spread_at_entry_pct / 100.0 # convert to decimal # Fee cost (round trip, taker both sides) fee_cost = (fee_a + fee_b) * 2 # Bid-ask cost (round trip) ba_cost = (half_spread_a + half_spread_b) * 2 # Slippage cost slip_cost = sigma_daily * sqrt(size / adv) * 2 # entry + exit # Funding cost (funding_per_8h in decimal fraction, e.g. 0.0001 = 0.01%/8h) fund_cost = funding_per_8h * hold_periods # Legging buffer leg_buffer = (sigma_spread_pct/100) * sqrt(avg_fill_latency_s/seconds_per_period) total_cost = fee_cost + ba_cost + slip_cost + fund_cost + leg_buffer net_edge = gross - total_cost breakeven_sigma = total_cost / (entry_z - exit_z) * 100 # as percent return { "gross_pct": gross * 100, "total_cost_pct": total_cost * 100, "net_edge_pct": net_edge * 100, "breakeven_sigma_pct": breakeven_sigma, "viable": net_edge > 0 }

19.9 Running Example — Bybit ↔ Lighter Cost Breakdown

🔁 Running Example: BTC Spread Trade Cost Analysis

Component	Calculation	Cost (bps)
Gross spread at entry (z=2.0, σ=7.5bps)	2.0 × 7.5bps	+15.0 bps
Taker fee Bybit (0.055% × 2)	round trip	−11.0 bps
Taker fee Lighter (0.030% × 2)	round trip	−6.0 bps
Bid-ask spread (both venues)	2 × half-spread	−0.8 bps
Slippage ($100k on $500M ADV)	σ×sqrt(size/ADV)	−0.3 bps
Funding (1 period, 0.01%/8h)	hold 8h	−1.0 bps
Legging buffer (0.5s latency)	σ×sqrt(0.5/60)	−0.5 bps
Net Edge		**−4.6 bps (−0.046%, LOSS)**

สรุป: Gross 15bps แต่ taker cost (17bps) + อื่น ๆ (2.6bps) = 19.6bps → net = −4.6bps → ห้ามเทรดในสภาพ retail taker — ต้องใช้ maker tier (rebate) หรือรอจน z ≥ 3.3

Break-even σ: total cost 19.6bps / (2.0 − 0.5) = 13.1bps → ต้องการ σ_spread ≥ 13.1bps (σ จริง = 7.5bps ❌ ไม่ผ่าน) → ลด cost ด้วย maker pricing หรือเลือก venue ที่ fee ต่ำกว่า

19.10 แบบฝึกหัด

แบบฝึกหัดบทที่ 19

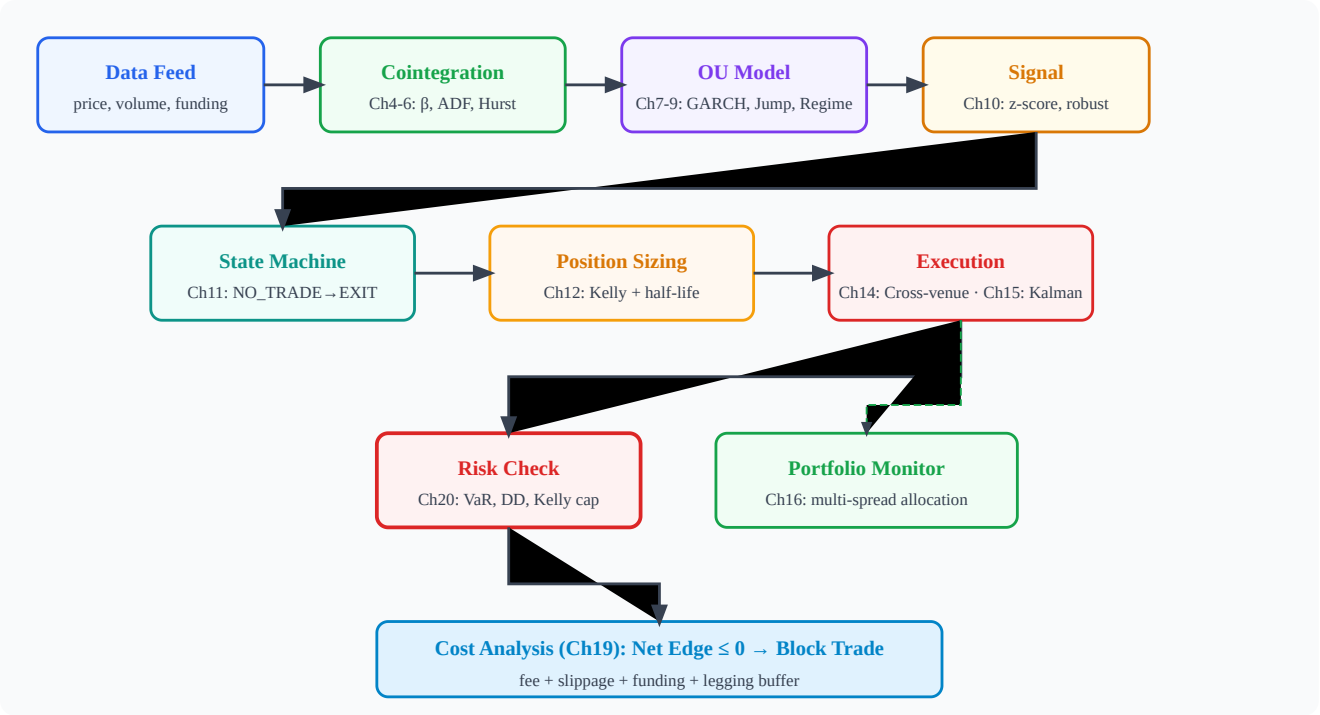
- ▶ ข้อ 1: ถ้า σ_spread = 3bps และ total cost = 4bps — ควรเทรดที่ z=2.0 ไหม?
- ▶ ข้อ 2: วิธีลด fee cost จาก 17bps (taker) เป็น rebate (maker) — ทำได้อย่างไร?
- ▶ ข้อ 3: ถ้าต้องการ minimum net edge 5bps — คำนวณ minimum z_entry ที่ต้องใช้

Statistical Arbitrage · บทที่ 20 — บทสรุป

Risk Management & Portfolio: สรุป Framework ทั้งเล่ม

จาก cointegration ถึง risk limits — ภาพรวมของ stat arb system ที่สมบูรณ์
ทุกชิ้นส่วนทำงานร่วมกัน — เพิกเฉยชิ้นเดียวทำให้ทั้งระบบพัง
แก่นของบทนี้ — บทสรุป
Stat arb ไม่ใช่แค่สูตรคณิตศาสตร์ — มันคือระบบที่ต้องการ **data quality + statistical rigor + execution discipline + risk management** ทำงานพร้อมกัน ขาดด้านใดด้านหนึ่งทำให้ edge หายไป

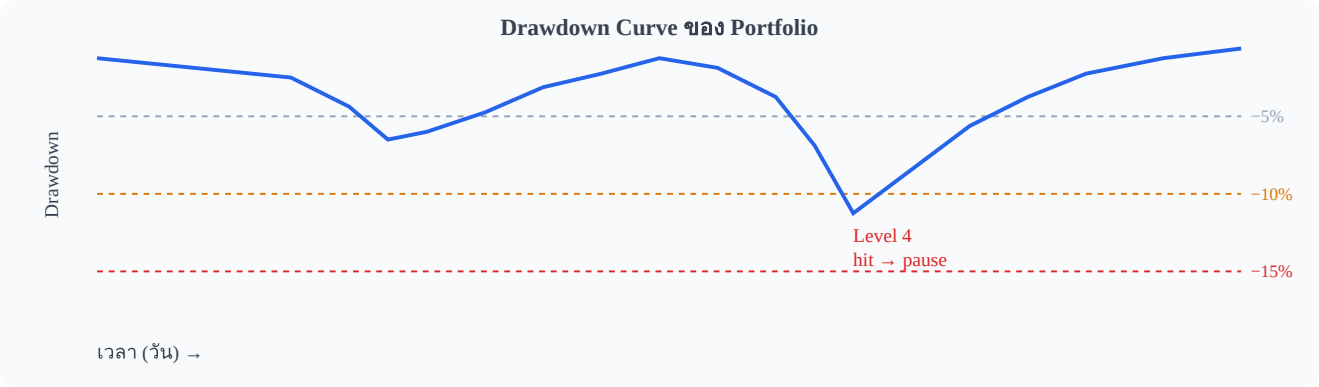
20.1 Framework Overview Diagram



Architecture ของ Stat Arb System — ทุก module เชื่อมกันเป็น pipeline จาก data ถึง execution

20.2 Drawdown Limits

Hierarchy ของ drawdown limits: Level 1 — Per Trade: max loss = entry_z × σ_spread × size × 1.5 # ×1.5 = jump-risk buffer Level 2 — Per Day: max daily PnL loss = 2% ของ total capital Level 3 — Per Strategy: max drawdown = 10% ของ strategy capital Level 4 — Portfolio: max portfolio drawdown = 15% → system pause เมื่อ hit limit: Level 1 → close trade Level 2 → close all trades, no new entries ถึงวันถัดไป Level 3 → reduce size 50%, review parameters Level 4 → full stop, manual review required



Drawdown curve — เมื่อแตะ level 4 (-15%) ระบบหยุด รอ manual review และ recalibration

20.3 VaR/CVaR ของ Spread Position

VaR_95 = σ_spread × 1.645 × size × sqrt(hold_periods) CVaR_95 = E[loss | loss > VaR_95] ≈ σ_spread × 2.063 × size × sqrt(hold_periods) (สำหรับ Normal distribution) ตัวอย่าง: σ_spread = 0.075% (7.5bps ต่อ period) size = $100,000 hold = 1 period VaR_95 = 0.00075 × 1.645 × 100,000 = $123 CVaR_95 = 0.00075 × 2.063 × 100,000 = $155 (expected loss ใน worst 5%)

20.4 Black Swan Scenarios

Scenario	Detection	Response	Recovery
Flash crash (>10% in 5m)	price_change > 5× σ_daily	Cancel all orders, close positions at market	Wait 30m, recheck cointegration
Exchange outage	API timeout > 30s	Hedge single leg on alternative venue	Close hedge when API restores
Cointegration breakdown	ADF p-value > 0.10 rolling	Enter INVALIDATED state immediately	Refit model with 30-day rolling data
Funding rate spike (>0.5%/8h)	Rate change > 3× historical σ	Reduce size 50%, widen stop-loss	Normalize after 1 funding period

20.5 Pseudo-code risk_check()

def risk_check(portfolio, new_trade, limits): """ portfolio: current open positions + daily PnL new_trade: proposed trade parameters limits: RiskLimits object (max_size, daily_stop, max_corr, max_var) """ # 1. Check trade size if new_trade.notional > limits.max_single_size: return "REJECT", "Size exceeds limit" # 2. Check daily drawdown if portfolio.daily_pnl < -limits.daily_stop: return "REJECT", "Daily stop hit" # 3. Check correlation with existing positions for pos in portfolio.positions: corr = rolling_correlation(new_trade.spread, pos.spread, window=30) # 30 trading days if corr > limits.max_correlation: return "REJECT", f"Correlation {corr:.2f} with {pos.id}" # 4. Check portfolio VaR hypothetical = portfolio.positions + [new_trade] new_var = compute_portfolio_var(hypothetical, confidence=0.95) if new_var > limits.max_var * portfolio.total_capital: return "REJECT", f"VaR {new_var:.1%} exceeds limit" # 5. Check edge (from Ch19) edge_result = compute_edge(new_trade) if not edge_result["viable"]: return "REJECT", f"Net edge {edge_result['net_edge_pct']:.2f}% <= 0" return "APPROVE", "All checks passed"

20.6 สรุปบทเรียนทั้งเล่ม

Ch.1–3
พื้นฐาน Synthetic Process
$\varepsilon = \log(P_A) - \beta \cdot \log(P_B)$ คือ spread ที่ mean-revert; OU process อธิบาย dynamics; half-life = $\ln(2)/\kappa$
Ch.4–6
Statistical Foundation
OLS/Kalman β; Engle-Granger cointegration; Hurst exponent $H<0.5$ = mean-reverting
Ch.7–9
Advanced OU Models
GARCH conditional volatility; Jump-Diffusion fat tail; Regime-Switching Markov chain
Ch.10
Robust Z-score
Median/MAD robust ต่อ outlier; rolling window z-score; entry/exit band design
Ch.11
State Machine
7-state machine ป้องกัน double-entry, race condition; INVALIDATED เมื่อ coint หาย
Ch.12
Position Sizing
Kelly $f^* = \text{edge}/\sigma^2$; half-life scaling; jump discount; max drawdown constraint
Ch.13
Perpetual Basis
Basis = perp − spot; fair basis จาก carry cost; entry เมื่อ excess > $k \cdot \sigma$
Ch.14–15
Execution
Cross-venue leg risk; synchronization; Kalman dynamic β real-time update
Ch.16
Portfolio
MV optimal weights; correlation heatmap; equal-weight robust baseline
Ch.17–18
Applications
Commodity calendar spread (storage/convenience yield); Options parity arbitrage
Ch.19
Cost Analysis
Gross − fee − spread − slippage − funding − legging = Net edge; compute before every trade
Ch.20
Risk Management
4-level drawdown limits; VaR/CVaR; black swan procedures; portfolio heat monitoring
Ch.21
Spot↔CFD Swap Arb
Spot↔CFD Swap Arbitrage บน MT5: ขยายกรอบเดิมไปสู่ CFD/FX/โลหะ
Ch.22
Options-Based Stat Arb
IV Surface · Box Spread · IV Skew · Cross-venue IV (Deribit↔Bybit) · Vol Risk Premium

20.7 Next Steps

เส้นทางจากทฤษฎีสู่ Live Trading

1. **Backtest ครบทุก module** — test บน historical data ≥ 1 ปี รวม regime ต่างๆ ตรวจว่า Sharpe, max drawdown อยู่ใน target
2. **Paper Trade 1–3 เดือน** — รัน system โดยไม่ใช้เงินจริง บันทึก slippage จริง vs model, latency จริง, execution quality
3. **Live Trading ด้วย Small Size** — เริ่มด้วย 10% ของ target size เพิ่มทีละน้อยเมื่อ P&L confirmed
4. **Monitor และ Recalibrate** — refit OU parameters ทุกสัปดาห์; ตรวจ cointegration ทุกวัน; review regime detection หลัง major events
5. **Expand Strategies** — เพิ่ม spread ใหม่เข้า portfolio เมื่อมีทุนและ operational capacity พอ

Golden Rules ของ Stat Arb

- ไม่มี edge ที่ "ถาวร" — ต้อง monitor และ adapt ตลอดเวลา
- Cost analysis ก่อนทุก trade — ไม่ assume
- Risk limits คือ non-negotiable — ไม่ override เพราะ "ครั้งนี้ต่างออกไป"
- Regime awareness เสมอ — ตลาดมีสภาวะต่างกัน parameter ชุดเดียวไม่พอ
- Execution quality กำหนด profitability เท่ากับ signal quality

แบบฝึกหัดสุดท้าย — Capstone

▶ คำถาม: Bybit และ Lighter ประกาศ merge platform — spread หายไปทันที ระบบควร respond อย่างไร?

จบ Statistical Arbitrage — Bybit ↔ Lighter BTC Perp Framework

บทที่ 1–22 · ครอบคลุมตั้งแต่ Synthetic Process จนถึง Live Risk Management, Spot↔CFD Swap Arbitrage และ Options-Based Stat Arb

Statistical Arbitrage · บทที่ 21

Spot ↔ CFD Swap Arbitrage บน MT5

Swap Rate = Interest Rate Alpha | Broker Compliance | Multi-Asset Universe
FX · Metals · Indices · Crypto CFD — ทุกอย่างที่ได้กำไร
แก่นของบทนี้ — อ่านก่อน
CFD ≠ crypto perp — ไม่มี funding rate แต่มี **swap overnight** ซึ่งเป็นดอกเบี้ยที่จ่ายหรือรับทุกคืน กำไรมาจาก 2 ส่วน: (1) swap differential ระหว่าง 2 ขา และ (2) ε mean reversion ตามปกติ ข้อระวังสำคัญ: MT5 มี netting restriction → ต้องใช้ 2 broker แยกกัน
ε = log(P_A) − β·log(P_B) | Alpha = Swap_net + ε_reversion

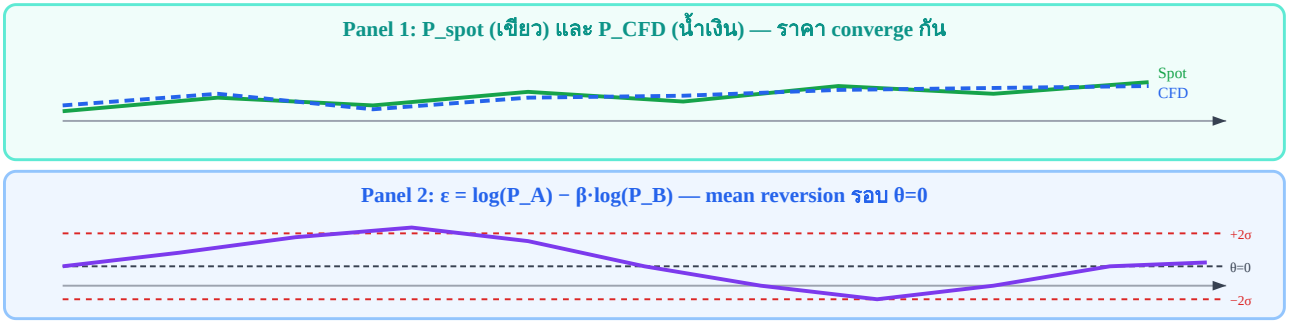
21.1 Spot ↔ CFD Mechanics

CFD (Contract for Difference) บน MT5 แตกต่างจาก crypto perpetual futures ในหลายมิติ แต่แก่นของ Statistical Arbitrage ยังเหมือนเดิม: ε = log(P_A) − β·log(P_B) mean-reverts กลับค่ากลาง สิ่งที่ต่างคือ **แหล่ง alpha เพิ่มเติม** ในรูปของ swap overnight

CFD vs Crypto Perp — ความต่างที่สำคัญ

- **CFD price ≈ Spot price + financing adjustment** — ปรับทุกวัน ไม่ใช่ทุก 8h แบบ perp
- **ε = log(P_A) − β·log(P_B)** ยังใช้แบบเดิม (อ้างอิง ch4) — ไม่ต้องเปลี่ยน framework
- **Arbitrage driver:** ส่วนต่าง swap rate ระหว่าง 2 ขา = "Interest Rate Alpha" — มาจาก central bank rate differential ไม่ใช่ demand sentiment

	Crypto Perp (Bybit)	CFD (MT5)
Financing mechanism	Funding rate ทุก 8h	Swap overnight ทุกวัน
Rate basis	Premium index (demand-driven)	Central bank interest rate differential
Rate size	0.01–0.10% / 8h	0.001–0.05% / day
Rate predictability	เปลี่ยนทุก 8h (volatile)	ค่อนข้างคงที่ (ผูกกับ CB rate)
Price transparency	Order book สาธารณะ	Broker sets spread (ไม่ fully transparent)



Panel บน: P_spot และ P_CFD เคลื่อนที่ใกล้กัน | Panel ล่าง: ε oscillates รอบ θ=0 ภายใน ±2σ band

21.2 Swap Rate Mechanics ใน MT5

Swap overnight คือดอกเบี้ยที่จ่ายหรือรับสำหรับการถือ CFD position ข้ามคืน — broker จะ rollover ตำแหน่งทุกเที่ยงคืนเวลา server (ส่วนใหญ่ 00:00 EET) และคิด/ให้ swap points ตามที่ระบุใน symbol specification

สูตรการคำนวณ Swap Amount
Swap_amount = Position_Lots × Contract_Size × Swap_Points / 10

Swap_Points = ค่าจาก MT5 symbol specification (points ต่อวัน) — ค่า *ลบ* = จ่าย, ค่า *บวก* = รับ
Contract_Size = ขนาดสัญญาต่อ lot (เช่น 100 oz สำหรับ XAUUSD)
หมายเหตุ: สูตรนี้ใช้สำหรับ FX/Commodity ที่คำนวณเป็น points — บาง broker ระบุ swap เป็น % ต่อวัน (crypto CFD) ซึ่งคำนวณต่างกัน ต้องเช็ค specification
ต้อง check specification ใน MT5: คลิกขวา symbol → Properties → Swap Long / Swap Short

ตัวอย่าง positive swap ที่พบบ่อยในตลาด (ค่าตัวอย่าง — ต้องเช็คกับ broker จริงก่อนเสมอ):

Instrument	ทิศทาง	เหตุผล	ตัวอย่าง swap/day/lot
XAUUSD (Gold CFD)	Short	Gold yield < USD yield → short receives rate	+$0.5–2.0
USOIL	Short (backwardation)	Futures curve downward sloping	+$1.0–3.0
USDHUF / USDTRY	Short USD	HUF/TRY interest rate สูงกว่า USD มาก	+$5.0–15.0
BTC CFD / ETH CFD	ขึ้นกับ broker	Varies — ต้องเช็ค specification	-$1.0–+$1.0

⚠️ คำเตือน: Swap Rates เปลี่ยนได้

Swap rates เปลี่ยนได้ทุกสัปดาห์ตามนโยบาย broker และ central bank rate cycle — ต้อง check specification ก่อนเปิด position เสมอ อย่าใช้ค่าจากสัปดาห์ก่อนโดยไม่ verify ใหม่

def compute_swap_harvest(position_lots, contract_size, swap_points_per_day, days_held, price=None, pip_value=0.1): """ คำนวณ swap income จากการถือ CFD ข้ามคืน swap_points_per_day: ค่าจาก MT5 symbol spec (pips/day) ค่าลบ = จ่าย swap | ค่าลบสำหรับ short = รับ swap (ขึ้นกับทิศ) """ swap_per_lot_per_day = swap_points_per_day * pip_value total_swap = position_lots * swap_per_lot_per_day * days_held return { "swap_per_day": position_lots * swap_per_lot_per_day, "total_swap": total_swap, "annualized_pct": (total_swap / (position_lots * contract_size * (price or 1))) * 365 / days_held * 100 }
ตัวอย่างหลัก: XAUUSD Short — Swap Harvest คำนวณ

Position: **Short 1 lot XAUUSD CFD** (100 oz), ราคา = $2,000/oz
Swap specification: Swap Short = **−1.20 swap points/day** (ค่าลบสำหรับ short = broker จ่ายให้เรา)
Pip value สำหรับ XAUUSD = $1.00 ต่อ pip ต่อ lot (1 pip = $0.01 × 100 oz = $1.00)

Swap per day = 1 lot × 1.20 × $1.00 = **$1.20/day**
ต่อเดือน (22 trading days) = $1.20 × 22 = **$26.40/month per lot**
ต่อปี (252 days) = $1.20 × 252 = $302.40 ≈ **1.5% annualized** บน notional $20,000 (1 lot × 10 oz margin)

21.3 Broker Compliance — กฎที่ต้องระวัง

นี่คือบทที่สำคัญที่สุดใน ch21 — การไม่อ่าน TOS ก่อน trade อาจทำให้ account ถูก ban และกำไรถูกยึดทั้งหมด อ่านทุกข้อด้านล่างอย่างละเอียด

❌ Netting Restriction — ปัญหาหลักของ MT5

MT5 default = Netting mode: ไม่สามารถถือ Buy และ Sell ของ symbol เดียวกันพร้อมกันใน account เดียว — ถ้าเปิด Buy แล้วเปิด Sell จะหักล้างกันทันที

- → **ต้องใช้ 2 broker แยกกัน:** Broker A สำหรับ leg หนึ่ง, Broker B สำหรับอีก leg
- → บาง broker มี "Hedging" account (MT4/MT5) ที่อนุญาต hedge ใน account เดียว แต่ต้องตรวจ TOS ก่อนเสมอ
- → การใช้ 2 broker แยกกันคือวิธีที่ถูกต้องและปลอดภัยที่สุด

❌ TOS Anti-Arbitrage Clauses

หลาย broker มีข้อห้ามใน Terms of Service ครอบคลุมคำต่อไปนี้:

- "arbitrage trading", "latency arbitrage", "swap scalping", "tick scalping"
- "exploiting pricing errors", "taking advantage of system delays"

→ **อ่าน TOS ของทุก broker ที่จะใช้ก่อนเสมอ** โดยเฉพาะหัวข้อ Prohibited Trading Practices
→ ถ้าเจอข้อห้าม → หา broker ที่ไม่มีข้อห้ายนี้ (มีอยู่หลาย broker โดยเฉพาะ ECN/STP broker)

⚠️ Minimum Hold Time

บาง broker กำหนด minimum position hold time (เช่น ≥ 2 นาที หรือ ≥ 1 ชั่วโมง) เพื่อป้องกัน latency arbitrage
→ **กระทบ intraday spread arb** — ถ้า hold น้อยกว่า minimum จะถูก void profit หรือ ban
→ **ไม่กระทบ overnight swing strategy** ของ ch21 เพราะเราถือข้ามคืน (หลาย ชม. ถึงหลายวัน)

⚠️ Requote และ Execution Risk

MT5 CFD execution ช้ากว่า crypto exchange มาก (200ms–2s vs <100ms)
→ **Legging risk สูงกว่า** — ขา A เข้าแล้วขา B ราคาเปลี่ยนก่อนจะ fill
→ **ใช้ limit orders แทน market orders เสมอ**ที่ทำได้ เพื่อลด slippage และ requote
→ เปิดทั้งสองขาภายใน 1–2 วินาทีหากเป็นไปได้

✅ สิ่งที่ทำได้อย่างถูกต้อง

- ✅ ถือ position ข้ามคืนเพื่อเก็บ swap (overnight swing) — เป็น legitimate investment activity
- ✅ ใช้ 2 broker แยกกัน — ไม่ผิด TOS เพราะแต่ละ broker ไม่รู้ว่าอีกด้านอยู่ที่ไหน
- ✅ Exit เมื่อ z-score กลับค่ากลาง — เป็น mean reversion trade ปกติ
- ✅ ใช้ limit orders เพื่อรับ spread แทนจ่าย spread

🚫 คำเตือนสำคัญ — อ่านก่อน trade

อย่า trade Swap Arb บน broker เดียวกันโดยใช้ hedge account — ผิด TOS ของเกือบทุก broker และอาจถูก account ban พร้อมยึดกำไรทั้งหมด กฎนี้ใช้กับทุก broker ที่มี anti-arbitrage policy ไม่ว่าจะเรียกว่า "hedging account" หรืออะไรก็ตาม — ถ้า 2 ขาอยู่ใน broker เดียวกัน broker มองเห็นทั้งคู่และจะระบุได้ว่าเป็น arbitrage

21.4 Asset Universe บน MT5

MT5 เปิดโอกาสให้ trade หลาย asset class ในแพลตฟอร์มเดียว แต่ละ category มี half-life และ friction ต่างกัน ต้องผ่าน Pair Selection Pipeline (ch5 §5.8) ก่อนเสมอ

Category	Y_t	X_t	Alpha Driver	Half-life ε	Friction (est.)
Metals	XAUUSD (Gold)	XAGUSD (Silver)	Commodity ratio + backwardation swap	3–10 วัน	40–70 bps
FX Crosses	EURUSD CFD	GBPUSD CFD	USD leg ร่วม → synthetic EURGBP	1–5 วัน	20–40 bps
Indices	US500 CFD	NAS100 CFD	US equity sector correlation	2–7 วัน	30–60 bps
Crypto CFD	BTCUSD CFD	ETHUSD CFD	Crypto market cycle β	0.5–3 วัน	60–120 bps

กรณีพิเศษ: FX Pairs — "4 สกุลเงิน แต่ USD หักล้างกัน"

EURUSD ↔ GBPUSD = Synthetic EURGBP

- **Long EURUSD + Short GBPUSD** = Long EUR / Short GBP / USD ≈ cancel (หักล้างกัน)
- **$\varepsilon = \log(EURUSD) - \beta \cdot \log(GBPUSD) \approx \log(EURGBP)$** สำหรับ $\beta \approx 1$
- ถ้า **$\beta \neq 1$** → มี residual USD exposure → ต้อง normalize notional ใน USD ก่อน sizing
- ตัวอย่าง: $\beta = 1.15$ → Long $100k EURUSD + Short $115k GBPUSD → USD ≈ 0

ข้อดีของใช้ majors แทน cross rate: EURUSD และ GBPUSD มี bid-ask แคบกว่า EURGBP โดยตรงมาก (10–20 bps vs 30–50 bps) → friction ต่ำกว่า → edge ดีกว่า

ต้องผ่าน Pair Selection Pipeline ก่อนเสมอ

ใช้ Pair Selection Pipeline จากบทที่ 5 §5.8 กับทุกคู่ก่อนทุกครั้ง — ตรวจ: (1) cointegration test (ADF/KPSS), (2) half-life ε, (3) ρ (correlation), (4) AR(1) coefficient, (5) breakeven σ check ทุกขั้น

ตอน

21.5 ε Quality และ Walk-Forward Calibration

การคำนวณ ε ใช้ OLS/TLS β เหมือนกับที่อธิบายใน ch4 ไม่ต้องเปลี่ยน framework แต่ **มีสิ่งที่ต่างจาก crypto** ที่ต้องระวัง:

ปัญหา θ Drift จาก Central Bank Rate Cycle

Financing spread ใน CFD เปลี่ยนตาม central bank rate cycle:
→ ถ้า Fed ขึ้น/ลดดอกเบี้ย → swap rate เปลี่ยน → ค่ากลาง θ ของ ε drift ได้
→ **Walk-Forward Calibration (ch3 §3.9) สำคัญยิ่งกว่าใน crypto** — ต้อง re-calibrate บ่อยกว่า
→ อย่าใช้ static θ หรือ static β ที่ fit ครั้งเดียวนาน > 3 เดือนโดยไม่ validate ใหม่

Recommended calibration window สำหรับ MT5 แต่ละ category:

Strategy type	Train window	Test window	Slide
FX intraday	30 วัน	10 วัน	10 วัน
Metals swing	90 วัน	30 วัน	30 วัน
Indices	60 วัน	20 วัน	20 วัน
Crypto CFD	45 วัน	15 วัน	15 วัน

ตัวอย่าง: BTC/ETH CFD Walk-Forward

β = 0.0444, ρ = 0.9745, AR(1) = 0.9602, Half-life = 17h
→ ใช้ Crypto CFD row: Train 45 วัน, Test 15 วัน, Slide ทุก 15 วัน
→ Half-life 17h = overnight swing — เหมาะสำหรับ strategy ที่ถือข้ามคืน
→ Walk-forward re-calibrate β ทุก 15 วัน เพื่อจับ drift ของ crypto market cycle

21.6 Breakeven Analysis สำหรับ MT5

Friction ของ MT5 CFD ต่างจาก crypto exchange — ต้องเข้าใจ structure ก่อน calculate edge:

Component	Crypto (Bybit)	MT5 ECN	MT5 Market Maker
Commission	0.055% + 0.030% (taker)	$3–7/lot (separate)	Embedded in spread
Bid-ask spread	0.002–0.004%	0.5–2 pips FX; wider metals	1–3 pips FX; wider metals
Maker option	✅ ลดเหลือ 0.02%	บางครั้ง (limit orders)	❌
Overnight	สามารถ +/−	Swap (positive เมื่อ harvesting)	Swap
Execution speed	<100ms	200ms–2s	500ms–3s

Breakeven σ Formula (เหมือน ch19 §19.7 / ch4 §4.5)
σ_stat ≥ total_friction / (z_entry − z_exit)

total_friction = bid-ask + commission + max(0, swap_net) — ถ้า swap เป็น income ให้ *หักออก* จาก friction
z_entry และ z_exit คือ entry/exit threshold ในหน่วย σ (เช่น 2.0 และ 0.5)

ตัวอย่างคำนวณ Breakeven σ สำหรับ 3 คู่หลัก

1. XAUUSD / XAGUSD

Friction ≈ 50 bps → breakeven σ = 50 / (2.0 − 0.5) = **33 bps**
σ_stat empirical ≈ 50–80 bps → **net edge = 17–47 bps per half-cycle** ✅

2. BTC / ETH CFD

Friction ≈ 80 bps → breakeven σ = 80 / (2.0 − 0.5) = **53 bps**
σ_stat ≈ 150–200 bps → **net edge = 97–147 bps** ✅ (แต่ fat tail → ต้องใช้ robust z-score)

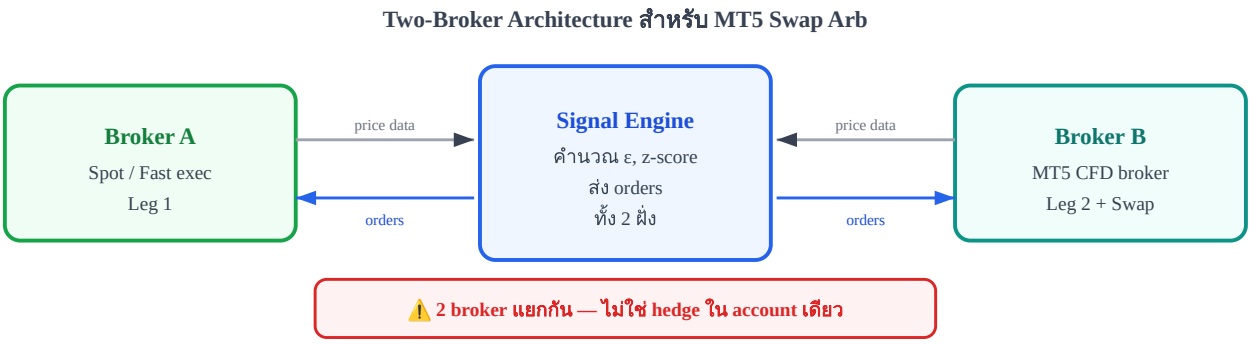
3. EURUSD / GBPUSD

Friction ≈ 25 bps → breakeven σ = 25 / (2.0 − 0.5) = **17 bps**
σ_stat ≈ 30–50 bps → **net edge = 13–33 bps** ✅

def mt5_breakeven_sigma(bid_ask_pct, commission_pct, swap_net_pct, entry_z=2.0, exit_z=0.5): """ คำนวณ breakeven σ_stat สำหรับ MT5 CFD pairs swap_net_pct: negative ถ้ารับ swap (income), positive ถ้าจ่าย ถ้า swap_net_pct < 0 → หักออกจาก friction (เป็นรายได้) """ total_friction = bid_ask_pct + commission_pct + swap_net_pct breakeven_sigma = total_friction / (entry_z - exit_z) return { "total_friction_bps": round(total_friction * 100, 1), "breakeven_sigma_bps": round(breakeven_sigma * 100, 1) } # ตัวอย่าง XAUUSD/XAGUSD (bid_ask=40bps, commission=20bps, swap_income=−10bps): # mt5_breakeven_sigma(0.004, 0.002, -0.001, 2.0, 0.5) # → total_friction = 0.004 + 0.002 − 0.001 = 0.005 = 50 bps # → breakeven_sigma = 50 / 1.5 = 33.3 bps ← ตรงกับ §21.6

21.7 Practical Setup: Two-Broker Configuration

การ implement Swap Arb บน MT5 ต้องใช้ 2 broker แยกกัน โดย Signal Engine เป็นตัวกลางคำนวณ ε และส่ง order ไปทั้งสองฝั่ง:



Signal Engine เชื่อมต่อทั้ง 2 broker แยกกัน — รับ price feed → คำนวณ ε → ส่ง order ไปแต่ละ broker

Data Synchronization

การ synchronize ข้อมูลจาก 2 broker ต้องทำอย่างระมัดระวัง:

- ใช้ **same timeframe** — bar close time ต้องตรงกัน (เช่น H1 close ทั้งคู่)
- ใช้ **same timezone reference** — แปลงเป็น UTC ก่อน align
- ตรวจ **missing bars** — หยุดเทรดถ้า data gap เกิน 2 bars ติดกัน
- ใช้ **mid-price** (bid+ask)/2 สำหรับคำนวณ ε — ไม่ใช้ bid หรือ ask อย่างเดียว

Basic MQL5 EA Structure (CFD Leg Execution)

// MQL5 EA skeleton for CFD leg execution // สำหรับ Broker B (MT5 CFD leg) — Broker A ใช้ API แยก input double ZEntryThreshold = 2.0; input double ZExitThreshold = 0.5; input double LotSize = 0.1; void OnTick() { double epsilon = ComputeEpsilon(); // คำนวณ ε จาก 2 prices double z_score = ComputeZScore(epsilon); // z = (ε − θ) / σ if (z_score > ZEntryThreshold && !HasPosition()) OpenShort(LotSize); // Short CFD leg (Broker B) else if (z_score < -ZEntryThreshold && !HasPosition()) OpenLong(LotSize); // Long CFD leg (Broker B) else if (MathAbs(z_score) < ZExitThreshold && HasPosition()) ClosePosition(); // ปิดเมื่อ z กลับค่ากลาง } double ComputeEpsilon() { double price_a = GetBrokerAPrice(); // รับราคาจาก Broker A (shared memory/pipe) double price_b = SymbolInfoDouble(_Symbol, SYMBOL_BID); return MathLog(price_a) - g_beta * MathLog(price_b); } double ComputeZScore(double eps) { return (eps - g_theta) / g_sigma; // g_theta, g_sigma update จาก walk-forward }

Position Sizing และ Legging Risk

- **Kelly framework (ch12)** ยังใช้ได้ — แต่ต้อง adjust สำหรับ higher friction ของ MT5
- แนะนำ half-Kelly หรือ quarter-Kelly สำหรับ MT5 เพราะ execution uncertainty สูงกว่า
- **Legging risk:** MT5 ช้ากว่า crypto มาก → ใช้ limit orders ที่ target entry price เพื่อลด slippage
- ถ้า leg หนึ่ง fill แล้วอีก leg ไม่ fill ภายใน timeout (เช่น 5 วินาที) → ยกเลิก leg แรกด้วย

ตัวอย่างหลัก: 3 คู่ที่น่าสนใจ

ตัวอย่างที่ 1: XAUUSD Gold Swap Harvest

Setup: Broker A — Gold spot บน exchange | Broker B — Short XAUUSD CFD บน MT5

พารามิเตอร์	ค่า
Swap received (short CFD)	$1.20/day per lot (backwardation)
ε Half-life	5 วัน
σ_stat	0.6% (60 bps)
Total friction	50 bps
Breakeven σ	50 / 1.5 = 33 bps ✅

Net alpha per trade: (σ_stat − BE_σ) × (z_entry − z_exit) = (0.60% − 0.33%) × 1.5 = **40 bps** จาก mean reversion
+ swap income: $1.20/day × 5 วัน (avg hold) = $6.00 per lot per trade
→ รวม alpha ≈ 0.54% + swap เสริม — เป็น compound edge ที่น่าสนใจ
ตัวอย่างที่ 2: BTC / ETH CFD (ตาม screenshot คู่หลัก)

Cointegration stats: β = 0.0444, ρ = 0.9745, AR(1) = 0.9602, Half-life = 17h

พารามิเตอร์	ค่า
Pair Selection Pipeline (ch5 §5.8)	✅ ผ่าน (swing only, robust z-score)
MT5 friction estimate	~80 bps
Breakeven σ	80 / 1.5 = 53 bps
σ_stat empirical	≈ 170 bps (200 bps >> 53 bps ✅)

Strategy: overnight swing, z_entry = 2.0, ใช้ MAD-based robust z-score เพราะ fat tail
Walk-forward: calibrate ทุก 15 วัน (Crypto CFD row จากตาราง §21.5)

ตัวอย่างที่ 3: EURUSD ↔ GBPUSD (Synthetic EURGBP)

β = 1.15 — ไม่ใช่ 1.0 เพราะ GBPUSD มี pip value ต่างกัน (GBPUSD pip มีค่าสูงกว่า)

พารามิเตอร์	ค่า
USD Normalization	Long $100k EURUSD + Short $115k GBPUSD (β-adjusted)
Net exposure	Long ~€74k / Short ~£55k / Net USD ≈ +$15k (residual ~13%)
ε interpretation	≈ synthetic EURGBP; ρ = 0.96; ADF p ≈ 0.02 ✅
Friction estimate	25 bps
Breakeven σ	25 / 1.5 = 17 bps
σ_stat typical	35–45 bps >> 17 bps ✅

ข้อดี: EURUSD+GBPUSD มี spread แคบกว่า EURGBP โดยตรง → friction ต่ำที่สุดในบรรดา 3 ตัวอย่าง
ข้อระวัง: ต้อง normalize notional ตาม β เสมอ — ถ้าใช้ equal notional จะมี residual USD exposure

21.8 แบบฝึกหัด

โจทย์ 21.1 — คำนวณ Swap Income

Short 2 lots XAUUSD CFD
Swap specification: Swap Short = −1.5 swap points/day (ค่าลบ = รับ swap)
สำหรับ XAUUSD: 1 lot = 100 oz, swap calculation: lots × swap_points × pip_value
Pip value สำหรับ XAUUSD = $1.00 ต่อ pip ต่อ lot (1 pip = $0.01 × 100 oz = $1.00)
ถาม: swap income ต่อวันเท่าไหร่? ต่อเดือน (22 trading days)?

▶ ดูคำตอบ

โจทย์ 21.2 — Broker Rule Check

Broker X มีข้อกำหนดใน TOS ว่า: *"Minimum position hold time = 5 นาที. Positions closed before 5 minutes will have profits voided."*
ถาม: กลยุทธ์ไหนของ ch21 ยังทำได้กับ Broker X? กลยุทธ์ไหนทำไม่ได้?

▶ ดูคำตอบ

โจทย์ 21.3 — Breakeven Check สำหรับ BTC/ETH CFD

BTC/ETH CFD บน MT5:
σ_stat = 2.0%, total friction = 80 bps, z_entry = 2.0, z_exit = 0.5
ถาม: (1) breakeven σ = เท่าไหร่? (2) trade ได้ไหม? (3) net edge ต่อ trade คือกี่ bps?

▶ ดูคำตอบ

อ้างอิงและบทที่เกี่ยวข้อง

บทที่เกี่ยวข้องในตำราชุดนี้

- **ch3 §3.9:** Walk-Forward Calibration — ใช้โดยตรงสำหรับ MT5 re-calibration schedule
- **ch4 §4.5:** Breakeven σ และการคำนวณ total friction — framework เดิม ใช้ได้กับ MT5
- **ch5 §5.8:** Pair Selection Pipeline — ต้องใช้กับทุกคู่ใน §21.4 ก่อนเสมอ
- **ch12:** Kelly Position Sizing — ปรับ half-Kelly สำหรับ MT5 execution uncertainty
- **ch19 §19.7:** Breakeven Analysis framework — เหมือนกันกับ §21.6

แหล่งอ้างอิงภายนอก

- **MetaTrader 5 Documentation** — "Swap Calculation" ใน MQL5 Reference: *SymbolInfoDouble(symbol, SYMBOL_SWAP_LONG/SHORT)* และ Contract Specification
- **Hull, J.C. (2022)** — *Options, Futures, and Other Derivatives*, 11th ed. — Ch.5 (Cost of Carry), Ch.3 (Futures Pricing) สำหรับ fair value framework
- **Vidyamurthy, G. (2004)** — *Pairs Trading: Quantitative Methods and Analysis* — OLS/TLS β estimation และ cointegration framework

Ch.21 Spot↔CFD Swap Arbitrage บน MT5 | ← **Ch.20** | → **Ch.22**

Statistical Arbitrage · บทที่ 22

Options-Based Stat Arb

IV Surface · Box Spread · Cross-venue IV · Vol Risk Premium
ต่อยอดจาก Put-Call Parity (บทที่ 18)
แก่นของบทนี้ — อ่านก่อน
แทนที่ราคา P_A, P_B → ใช้ **Implied Volatility IV_A, IV_B**
ε_IV = IV_A − β·IV_B ยังคง mean-revert เหมือน ε ราคา — ใช้ pipeline เดิมทั้งหมด (ADF, half-life, z-score)
แหล่งกำไรใหม่: IV mispricing จาก skew, cross-venue, vol risk premium
ε_IV = IV_A − β·IV_B | Pipeline: ADF → half-life → z-score (เหมือน ch3–ch10)
Prerequisites — ต้องเข้าใจก่อนอ่านบทนี้

- **ch3** OU process, half-life, mean-reversion — ε_IV fit OU เหมือนกัน
- **ch4 §4.2b** OLS β = Cov/Var — β_IV คำนวณแบบเดียวกัน
- **ch5** ADF + Pair Selection Pipeline — ใช้กับ IV pairs ได้ตรงๆ
- **ch10** z-score entry/exit — เหมือนกันทุกประการ
- **ch12** Kelly sizing — f* × 0.3 สำหรับ short vol strategy
- **ch18 §18.8** ε_PCP — starting point สำหรับ options-based ε
- **pm-part1** PCP fundamentals + synthetic construction

Glossary — คำศัพท์ Options ที่ใช้ในบทนี้

- **ATM (At-the-Money)**: strike K = spot price S — เป็น strike ที่ liquid ที่สุด
- **Straddle**: Long Call(K) + Long Put(K) — กำไรเมื่อ vol สูง ไม่สนทิศทาง
- **Vega**: sensitivity ของ option price ต่อ IV change — Long option = Long Vega
- **Gamma**: sensitivity ของ delta ต่อ S change — Short straddle = Short Gamma
- **Delta hedge**: ซื้อ/ขาย underlying เพื่อให้ net delta ≈ 0 (ป้องกัน directional risk)
- **Vol Smile/Skew**: IV ที่แตกต่างกันตาม strike K — OTM puts มักมี IV สูงกว่า OTM calls
- **25Δ**: options ที่มี absolute delta = 0.25 — OTM แต่ยัง liquid พอ

22.1 IV เป็น Process ใหม่ — สร้าง ε บน Vol Surface

Implied Volatility (IV) คือค่า σ ที่แทนลงใน Black-Scholes แล้วได้ราคา option ที่ตลาดซื้อขายจริงพอดี — IV ไม่ใช่ historical volatility แต่คือ *market's forward-looking expectation of volatility* ณ ขณะนั้น

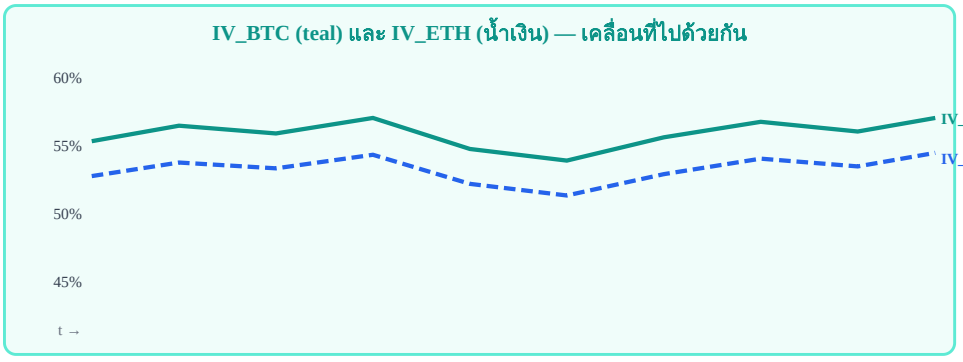
จุดสำคัญ: IV ของ asset ที่มีความสัมพันธ์กัน (เช่น BTC และ ETH) ควรเคลื่อนที่ไปด้วยกัน — เมื่อ crypto market มี risk-off event ทั้งสองจะพุ่งขึ้นพร้อมกัน และเมื่อตลาดสงบ ทั้งสองจะลดลงพร้อมกัน

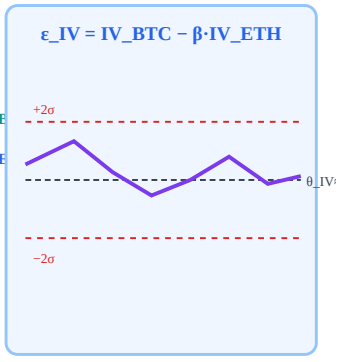
ด้วยเหตุนี้:

สมการหลัก — IV Spread Process
ε_IV,t = IV_A(K,T,t) − β·IV_B(K,T,t)

โดยที่ K คือ strike (เลือก ATM เพื่อ liquidity สูงสุด), T คือ **constant-maturity** (เช่น 30d interpolated — ไม่ใช่ fixed-calendar expiry ที่ T หดทุกวัน), t คือ timestamp
β_IV คำนวณจาก OLS: β_IV = Cov(IV_A, IV_B) / Var(IV_B) — เหมือนกับ บทที่ 4 §4.2b ทุกประการ
θ_IV = E[ε_IV] = ค่ากลางของ spread (อาจไม่ใช่ 0 เพราะ structural premium ต่างกัน)

	ε ราคา	**ε_IV**
Input	log P_A, log P_B	IV_A(K,T), IV_B(K,T)
สร้างด้วย	OLS β (ch4)	OLS β_IV
mean ของ ε	θ ≈ 0 (log-price)	θ_IV (vol level differential)
mean-revert เพราะ	cointegration (ch5)	vol surfaces ของ correlated assets linked
test ด้วย	ADF + half-life + z-score	ADF + half-life + z-score — เหมือนกัน
trade ด้วย	z-score บน ε (ch10)	z-score บน ε_IV — pipeline เดิมทั้งหมด





ซ้าย: IV_BTC (teal) และ IV_ETH (น้ำเงิน) เคลื่อนที่ไปด้วยกัน | ขวา: ε_IV oscillates รอบ θ_IV ≈ 3% ภายใน ±2σ band
ทำไม IV ของ BTC กับ ETH ควร Cointegrate?

- BTC และ ETH ขับเคลื่อนโดย **crypto market sentiment** เดียวกัน — ความกลัวหรือความโลภกระทบทั้งคู่พร้อมกัน
- Institutional vol sellers มักขาย **volatility ทั้งสองพร้อมกัน** เพราะ hedge ด้วย BTC และ ETH ในสัดส่วนใกล้เคียงกัน
- Risk-off event (เช่น regulatory news, macro shock) กระทบ **IV ทั้งคู่พร้อมกัน** ในทิศทางเดียว
- → ε_IV = IV_BTC − β·IV_ETH stationary ได้ ถ้า pair เลือกถูก และ β calibrate ด้วย OLS บน window ที่เหมาะสม

§22.1 ใช้ IV ของ assets ต่างกัน → §22.2 ดู case พิเศษที่ ε ควรเป็น 0 อย่างแน่นอน: Box Spread ▶

22.2 Box Spread Stat Arb — ε_box

Box Spread คือ synthetic bond ที่สร้างจาก options 4 legs โดยรวม long bull call spread กับ short bull put spread ที่ K เดียวกัน สิ่งที่ได้คือ fixed payoff = K2 − K1 ณ expiry ไม่ว่า underlying จะเป็นเท่าไหร่

พื้นฐาน: Bull Call Spread คืออะไร?

Bull Call Spread = Long Call(K1) + Short Call(K2) ที่ K1 < K2 — payoff = max(S−K1, 0) − max(S−K2, 0) ซึ่งมีค่าสูงสุด = K2−K1
Box Spread = Long Bull Call Spread (K1,K2) + Short Bull Put Spread (K1,K2) = ผลรวมที่ได้ K2−K1 แน่นอนทุก scenario ไม่ว่า S จะเป็นเท่าไหร่ → Box = synthetic zero-coupon bond ที่ maturity = K2−K1

Fair Value ของ Box Spread
V_box = (K2 − K1) · e^{−rT}
Market box price = (C_K1 − P_K1) − (C_K2 − P_K2)
ε_box = Market_box − (K2 − K1)·e^{−rT}

เนื่องจาก payoff ของ box = K2 − K1 แน่นอน fair value จึงต้องเป็น present value ของ spread นั้น ε_box ควร = 0 ในตลาดที่ไม่มี friction — เมื่อ ε_box ≠ 0 คือ arbitrage opportunity
สูตร Box Spread Market box price = (C_K1 − P_K1) − (C_K2 − P_K2) Fair box value = (K2 − K1) · e^{−rT} ε_box = Market_box − Fair_box

ε_box	ความหมาย	Action
> 0 (แพงไป)	ตลาดประเมิน box สูงกว่า fair value	Short box: **Sell C(K1) + Buy P(K1) + Buy C(K2) + Sell P(K2)**
< 0 (ถูกไป)	ตลาดประเมิน box ต่ำกว่า fair value	Long box: **Buy C(K1) + Sell P(K1) + Sell C(K2) + Buy P(K2)**
≈ 0	Fair — ไม่มีความไม่สมดุล	ไม่มีสัญญาณ รอ

ตัวอย่างหลัก: BTC Box Spread บน Deribit

ข้อมูล:
K1 = 90,000, K2 = 100,000 (spread = $10,000)
T = 7 วัน = 7/365, r = 5%/ปี
Deribit quotes: C(90k) = $6,200, P(90k) = $1,050, C(100k) = $2,400, P(100k) = $2,350

คำนวณ Market Box Price Market box = (C_K1 − P_K1) − (C_K2 − P_K2) = (6200 − 1050) − (2400 − 2350) = 5150 − 50 = $5,100 # คำนวณ Fair Box Value Fair box = 10000 × e^{−0.05×7/365} = 10000 × 0.99904 = $9,990 # ε_box ε_box = 5100 − 9990 = **−$4,890**

→ ε_box **ลบมาก** → ตลาดประเมิน box ต่ำกว่า fair value $4,890
→ ถ้า total friction ≤ $4,890 → **Long box**: Buy (C_K1−P_K1), Sell (C_K2−P_K2) รับกำไร $4,890 ณ expiry

หมายเหตุ: ε_box = −$4,890 นี้ใหญ่ผิดปกติและเกิดจากการที่ option quotes ที่ให้ไปนั้น **ละเมิด PCP โดยตั้งใจ** เพื่อสาธิต — ราคาเหล่านี้ไม่สามารถ coexist พร้อมกันในตลาดจริงได้ ในตลาดปกติ ε_box จะอยู่ที่ ±$50–200 เท่านั้น และ arb trader จะเข้ากดให้เป็น 0 ทันที

ข้อจำกัดของ Box Spread Arb

- ต้องการ **4 legs พร้อมกัน** → friction สูง 4× เทียบกับ single leg trade
- Liquidity ของ deep ITM/OTM options **ต่ำมาก** → bid-ask กว้าง → ε_box ที่ profitable จริงหายาก
- ดีที่สุดกับ **near-ATM options ที่ liquid** — strike ที่ volume สูง
- Crypto: ใช้ Deribit เพราะ **European options** (ไม่มี early exercise risk) — American options จะทำให้ fair value ซับซ้อนขึ้น
- ε_box ≠ 0 ที่คงอยู่มักเกิดจาก execution friction ไม่ใช่ market error — ต้องวัด friction จริงก่อน

ดูพื้นฐาน Box Spread เพิ่มเติม: *arb-part1* · *arb-part2* · *arb-part3*

Box Spread วัด mispricing ที่ PCP level → §22.3 ขยับขึ้นดู mispricing บน vol surface: IV Skew ▶

22.3 IV Skew Stat Arb — Risk Reversal (RR_25Δ)

IV Skew หมายถึง IV ของ options ที่ strike ต่างกันนั้นไม่เท่ากัน — put ที่ strike ต่ำมักมี IV สูงกว่า call ที่ strike สูง เพราะ demand ซื้อ protective put สูงกว่า ปรากฏการณ์นี้สะท้อนอยู่ใน **vol smile curve ที่เอียงซ้าย**

ทำไมต้องวัด IV ข้ามสไตรก์? — PCP Constraint

Put-Call Parity (บทที่ 18 §18.1) บังคับว่า IV_call(K,T) = IV_put(K,T) สำหรับ **K เดียวกัน, T เดียวกัน** เสมอ — ถ้าต่างกัน จะมี riskless arbitrage ทันที ดังนั้น ε_skew จึงวัดไม่ได้ที่ K เดียวกัน
→ "Skew" ต้องวัดโดยเปรียบ IV *ข้ามสองสไตรก์ที่ต่างกัน*
→ วิธีมาตรฐานของตลาด: **Risk Reversal 25Δ** — เปรียบ IV ที่ 25Δ put (strike ต่ำ) กับ IV ที่ 25Δ call (strike สูง)

Risk Reversal 25Δ — นิยาม IV Skew มาตรฐาน
RR_25Δ = IV_call(Δ=+0.25) − IV_put(Δ=−0.25)

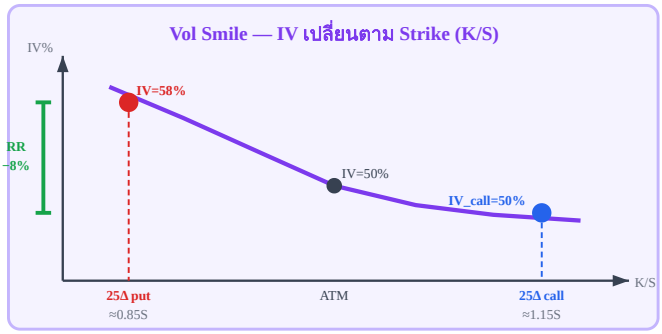
25Δ put: strike ≈ 80–90% ของ spot (OTM put ทางซ้ายของ smile curve)
25Δ call: strike ≈ 110–120% ของ spot (OTM call ทางขวาของ smile curve)

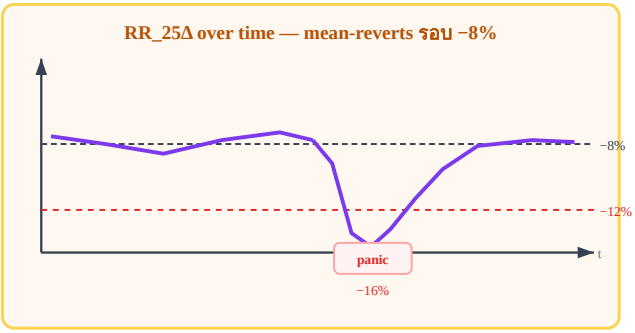
ปกติ: RR_25Δ ≈ −5% ถึง −10% สำหรับ BTC (กว้างกว่า FX/equity เพราะ downside skew เรื้อรัง)
Panic spike: RR_25Δ ลงถึง −15% ถึง −25% แล้ว mean-revert กลับ

ε_RR = RR_25Δ − θ_RR (θ_RR = mean ปกติ เช่น −8% สำหรับ BTC)

หมายเหตุ: ตลาด FX ใช้ convention ตรงข้าม (RR = IV_call − IV_put > 0 เมื่อ call แพงกว่า) — บทนี้ใช้ convention เดียวกับ equity/crypto ที่ RR < 0 เมื่อ put skew มี

ทฤษฎีบังคับว่า IV_call = IV_put ที่ K เดียวกัน → ในตลาดจริง smile curve เอียงทำให้ IV ต่างกัน ข้ามสไตรก์ → **เทรด gap นั้นเมื่อ spike ผิดปกติ**





ซ้าย: Vol smile เอียงซ้าย (pure skew) — 25Δ put (K≈85%S) IV=58% > 25Δ call (K≈115%S) IV=50% → RR_25Δ = IV_call−IV_put = 50%−58% = −8% | ขวา: RR_25Δ mean-reverts รอบ −8% spike ลงถึง −16% ระหว่าง panic แล้วกลับ

สถานการณ์	RR_25Δ	ε_RR = RR − θ_RR	Action
Normal	≈ −5% ถึง −10% (BTC)	ε_RR ≈ 0	ไม่มีสัญญาณ — รอ
Panic: put demand พุ่ง	spike ลง −12% ถึง −20%	ε_RR ≪ 0, z < −2	Long RR: Buy 25Δ Call + Sell 25Δ Put
Post-panic: normalize	กลับ −3% ถึง −5%	ε_RR ≈ 0	Close position รับกำไรจาก reversion

ตัวอย่างหลัก: BTC 25Δ Risk Reversal — Panic Event

สภาวะปกติ: BTC spot S = $95,000, T = 30 วัน

25Δ put: strike ≈ 85% ของ spot = $80,750 (IV สูงกว่า ATM เพราะ demand ป้องกัน downside) # 25Δ call: strike ≈ 115% ของ spot = $109,250 (IV ต่ำกว่า ATM) IV_put(25Δ) = 58% IV_call(25Δ) = 50% RR_25Δ = IV_call − IV_put = 50% − 58% = **−8%** ← normal (BTC range −5% ถึง −10%) Historical θ_RR = −8%, σ_RR = 2% ε_RR = −8% − (−8%) = 0% z_RR = 0 → ไม่มีสัญญาณ

ช่วง panic (flash crash −15%):

Spot ลงเหลือ ≈ $80,750 → 25Δ strikes ขยับตาม IV_put(25Δ) = 73% # put demand พุ่งแรง — ผู้คนแห่ซื้อ downside protection IV_call(25Δ) = 57% # call IV ขึ้นน้อยกว่ามาก RR_25Δ = IV_call − IV_put = 57% − 73% = **−16%** ← spike! ε_RR = −16% − (−8%) = −8% z_RR = −8% / 2% = **−4.0** ← >> 2σ → Signal!
ทำไม Long RR? — เหตุผลเบื้องหลังการเทรด

ในช่วง panic: put IV พุ่งสูงผิดปกติ → put *ราคาแพง* relative to fair value
→ **ขาย put** (เก็บ premium ที่แพงเกินไป) + **ซื้อ call** (ถูกกว่า relative)
→ รอ panic ผ่านไป → skew normalize → RR กลับ −8% → ปิด position รับกำไร

⚠️ Long Call + Short Put = net **positive delta** (long underlying exposure)
→ **ต้อง delta-hedge:** ขาย BTC futures จำนวน ≈ net_delta units ทันทีที่เปิด position
→ ไม่เช่นนั้นจะเป็นการ bet ทิศทาง ไม่ใช่ skew arb

Action: **Long RR** — Buy 25Δ Call + Sell 25Δ Put บน Deribit + Short BTC-PERP ≈ net_delta units (delta hedge) → รอ panic subsides → RR_25Δ กลับ −8% → Close ทุก leg → กำไร ≈ Δ_ε_RR × Vega (per 1% IV change)

IV Skew Selection Algorithm: *vp-part4*

§22.3 เปรียบ IV ข้ามสไตรก์ใน exchange เดียว → §22.4 เปรียบ IV ของ option เดียวกันข้าม exchanges ▶

22.4 Cross-venue IV Stat Arb — Deribit ↔ Bybit

สำหรับ underlying เดียวกัน (BTC), strike K เดียวกัน, expiry T เดียวกัน — IV ควรเท่ากันระหว่าง exchanges ในทางทฤษฎี แต่ในทางปฏิบัติ IV แตกต่างกันได้จาก liquidity structure และ market maker behavior ที่ต่างกัน

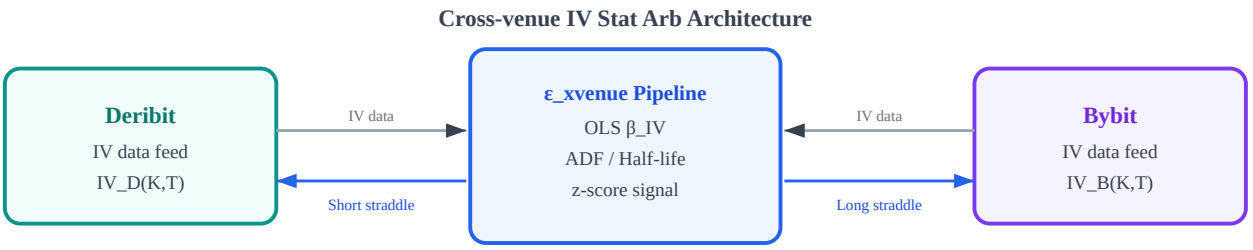
ε Cross-venue — Process เดิม Pipeline เดิม
ε_xvenue = IV_Deribit(K,T) − β·IV_Bybit(K,T)

β ≈ 1 (same underlying) แต่มี small deviation จาก structural liquidity premium
→ ใช้ OLS คำนวณ β_IV จาก historical data เหมือน บทที่ 4 §4.2b
→ ε_xvenue มี θ_IV บวก = structural alpha (Deribit IV สูงกว่าเรื้อรัง)

Apply Full Stat Arb Pipeline (ต่อจากบทก่อน)

1. **Scatter plot** IV_Deribit vs IV_Bybit → ρ ≥ 0.85? (ch5)
2. **OLS β_IV** = Cov(IV_D, IV_B) / Var(IV_B) (ch4 §4.2b)
3. **ADF** บน ε_xvenue → p < 0.05? (ch5)
4. **Half-life** → อยู่ใน target horizon? (ch3)
5. **z-score** entry/exit (ch10)

ตัวอย่างหลัก: BTC IV Cross-venue — 90 วัน Hourly Data
Pipeline Results β_IV = 0.98 # Deribit ≈ Bybit แต่สูงกว่าเล็กน้อย θ_IV = +0.8% # Deribit IV สูงกว่า Bybit 0.8% โดยเฉลี่ย σ_IV = 1.2% # standard deviation ของ ε_xvenue Half-life = 8 ชม. # φ = 0.92 → OU process ADF p = 0.021 # ✅ stationary # Entry Signal ε_xvenue > θ + 2×σ = 0.8% + 2.4% = 3.2% → Deribit แพง relative → Short Deribit straddle + Long Bybit straddle # Exit Signal ε_xvenue < θ + 0.5×σ = 0.8% + 0.6% = 1.4%



Pipeline รับ IV feed จากทั้ง Deribit และ Bybit → คำนวณ ε_xvenue → ส่ง trade signal: Short straddle (Deribit) + Long straddle (Bybit)
ทำไม Deribit IV > Bybit IV เรื้อรัง

- **Deribit** = dominant crypto options exchange → vol sellers มากกว่า → bid-ask แคบกว่า → IV คำนวณได้แม่นยำกว่า
- **Bybit** = newer options market → market makers demand higher premium เพราะ uncertainty สูงกว่า → IV underpriced relative to Deribit
- → ε_xvenue มี **positive drift θ_IV = structural alpha** — ε_xvenue = θ + noise
- → mean-revert รอบ θ_IV บวก ไม่ใช่ 0

Cross-venue concept: *eye-part3*

§22.1–22.4 เปรียบ IV ระหว่างสองสิ่ง → §22.5 เปรียบ IV กับ realized volatility ที่เกิดขึ้นจริง ▶

22.5 Vol Risk Premium — IV − RV

Vol Risk Premium (VRP) คือปรากฏการณ์ที่ Implied Volatility (IV) สูงกว่า Realized Volatility (RV) โดยเฉลี่ยอย่างต่อเนื่อง — เพราะผู้ซื้อ options ยินดีจ่ายเพิ่มเป็น insurance premium แหล่งกำไรคือการ *ขาย volatility* รับ IV เมื่อ IV สูงกว่า RV ที่คาด

RV_Nd (Realized Volatility N วัน) = annualized standard deviation ของ daily log-returns ใน N วันที่ผ่านมา: `RV = √(252/N · Σ r_t²)` โดย r_t = log(S_t/S_{t-1})

VRP Process — Mean-Reversion แหล่งที่สาม
ε_VRP = IV_t − RV_t (ปกติ > 0 = positive alpha เรื้อรัง)

เมื่อ ε_VRP spike สูงผิดปกติ (panic-driven IV explosion) → ขาย vol รอ normalize
เมื่อ ε_VRP ลบ (vol surprise — realized > implied) → ซื้อ vol ป้องกัน
→ ใช้ z-score บน ε_VRP เหมือนกับ ε ราคา — pipeline เดิมทั้งหมด

	BTC/ETH Options	S&P500 Options
Average VRP	10–20% (IV − RV)	3–5%
Regime: normal	ε_VRP ≈ +15%	ε_VRP ≈ +4%
Regime: crisis	ε_VRP → −20% (IV explodes)	ε_VRP → −30% ★
Mean-reversion τ½	≈ 5–15 วัน	≈ 10–20 วัน

★ S&P500 ε_VRP ลบมากกว่าในช่วง crisis เพราะ IV_normal ต่ำกว่า BTC มาก (VIX ≈ 15% vs BTC IV ≈ 60%) เมื่อ realized vol พุ่งพร้อมกัน อัตราส่วน RV/IV ของ S&P500 จึงเกิน 1 ได้มากกว่า

Trade	Short straddle เมื่อ z > 2σ	Short straddle เมื่อ z > 2σ

ตัวอย่างหลัก: BTC VRP Signal — Normal vs Bull Run

สภาวะปกติ:

IV_30d = 65% RV_30d = 48% # realized ใน 30 วันที่ผ่านมา ε_VRP = 65% − 48% = +17% Historical mean ε_VRP = +13%, σ_VRP = 6% z_VRP = (17% − 13%) / 6% = +0.67 ← ไม่ถึง 2σ → ยังไม่ trade

ช่วง bull run (IV พุ่งไม่สมเหตุผล):

IV_30d = 90% RV_30d = 55% ε_VRP = 90% − 55% = +35% z_VRP = (35% − 13%) / 6% = **+3.67** ← >> 2σ → Signal! Action: **ขาย vol** → Short straddle BTC (Sell ATM Call + Sell ATM Put) หรือ Short variance swap (ถ้ามี) Position size: Kelly f* × 0.3 (conservative เพราะ tail risk)
ความเสี่ยง VRP Trade — อ่านก่อนเทรด

- **Tail risk:** vol spike ใน crisis ทำ loss ไม่จำกัด — Short straddle = Short gamma = unlimited downside
- **Black swan:** RV พุ่งเกิน IV ได้ — ε_VRP negative หนัก ทำให้ loss ใหญ่มาก
- **Stop-loss บังคับ:** ต้องมี stop เมื่อ ε_VRP ลดลงต่ำกว่า mean อย่างชัดเจน (เช่น z_VRP < −1)
- **Kelly sizing อนุรักษ์นิยม:** f* ≤ 30% ของ Kelly เต็ม (ch12) เพราะ left tail ของ P&L ไม่สมมาตร
- **Delta hedge ต่อเนื่อง:** Short straddle มี gamma risk — ต้อง re-hedge delta ทุก 30–60 นาที

Vol Clustering และ GARCH forecast: *vol-part4*

22.6 Friction & Breakeven สำหรับ Options

Options มี friction structure ที่ซับซ้อนและสูงกว่า perpetual futures มาก — ต้องเข้าใจ component ทุกตัวก่อนคำนวณ edge:

Component	Perp (Bybit)	Options (Deribit)
Commission	0.055% taker	0.03% taker (maker 0%)
Bid-ask spread	0.002–0.004% ของ notional	0.5–3% ของ option price (สูงมาก)
Slippage	ต่ำ (deep liquid)	สูง โดยเฉพาะ OTM options
Greeks hedge cost	ไม่มี	Delta hedge ทุกวัน (Gamma cost สะสม)
Overnight financing	Funding rate (อาจ + หรือ −)	ไม่มี (IV baked in เป็น annualized)
สรุป friction	~13 bps ทั่วไป	~100–300 bps (5–20× สูงกว่า perp)

Breakeven σ_IV Formula (เดิมจาก ch19 §19.7)
Breakeven σ_IV = total_friction / (z_entry − z_exit)

เงื่อนไข: σ_IV ≥ breakeven σ_IV → มี edge
สำหรับ options: total_friction สูงกว่า perp 5–10× → ต้องเลือกเฉพาะ signal ที่ deep enough
ตัวอย่าง: Breakeven Calculation สำหรับ Options Strategy
Parameters: z_entry = 2.0 z_exit = 0.5 total_friction = 200 bps ใน IV-space (= 2% IV change; ต้องใช้หน่วยเดียวกับ σ_IV) Breakeven σ_IV = 200 / (2.0 − 0.5) = 200 / 1.5 = **133 bps = 1.33%** → ต้องการ σ_IV ≥ 1.33% ถึงจะ trade ได้ → เทียบ: Perp breakeven σ ≈ 13 bps — options ต้องการ σ สูงกว่า ~10×
สรุป: เมื่อไหร่จึง Trade Options Stat Arb ได้

- **ε_IV หรือ ε_skew หรือ ε_VRP ≥ 1–3%** (ขึ้นอยู่กับ friction จริง) — ไม่ใช่ทุก signal
- ATM options ที่ **liquid** เท่านั้น — OTM options มี bid-ask กว้างเกินไป
- Deribit BTC/ETH เป็น venue ที่ดีที่สุดสำหรับ crypto เพราะ European + ลึกที่สุด
- ต้องวัด friction จริงล่วงหน้า — อย่าประมาณ friction ต่ำเกินไป

22.7 Practical Setup — Data, Code, Position Sizing

Data Sources

Data	Source	Free/Paid
BTC/ETH option prices + IV	Deribit API (REST + WebSocket)	Free
IV Surface historical	Deribit `/api/v2/public/get_volatility_index_data`	Free
Cross-venue comparison	Bybit Options API	Free
RV calculation	Spot price (Binance/Bybit) คำนวณเอง	Free

Pseudo-code: IV Stat Arb Pipeline

1. Fetch ATM IV จาก Deribit def get_atm_iv(currency, expiry): # Deribit API: /api/v2/public/get_order_book # หา ATM strike ที่ |delta| ≈ 0.5 # return (IV_call + IV_put) / 2 (mid IV) ... # 2. สร้าง ε_IV time series btc_iv = [get_atm_iv("BTC", expiry) for expiry in expiry_list] eth_iv = [get_atm_iv("ETH", expiry) for expiry in expiry_list] # OLS β_IV = Cov(IV_BTC, IV_ETH) / Var(IV_ETH) — ch4 §4.2b beta_iv = np.cov(btc_iv, eth_iv)[0,1] / np.var(eth_iv, ddof=1) epsilon_iv = np.array(btc_iv) - beta_iv * np.array(eth_iv) # 3. z-score (ch10) mean_iv = epsilon_iv.mean() std_iv = epsilon_iv.std() z_iv = (epsilon_iv[-1] - mean_iv) / std_iv # 4. Entry/Exit Signal if z_iv > 2.0: # BTC IV แพงเทียบกับ ETH IV → Short BTC vol, Long ETH vol # Short BTC straddle (Sell ATM call + Sell ATM put บน Deribit) # Long ETH straddle (Buy ATM call + Buy ATM put บน Deribit/Bybit) signal = "SELL_BTC_IV_BUY_ETH_IV" elif z_iv < -2.0: # BTC IV ถูกเทียบกับ ETH IV → Long BTC vol, Short ETH vol signal = "BUY_BTC_IV_SELL_ETH_IV" else: signal = "NO_SIGNAL" # รอ # 5. Exit if abs(z_iv) < 0.5: signal = "EXIT" # z กลับค่ากลาง

Position Sizing กับ Options — Vega-equivalent

Options position size ไม่ควรใช้ dollar-notional เหมือน perp เพราะ sensitivity ต่อ IV ต่างกัน — ต้องใช้ **Vega-equivalent**:

Vega-Neutral Sizing

- **Vega** ≈ $S\cdot\sqrt{T}\cdot N'(d_1)/100$ — ค่าเปลี่ยนของ option price ต่อ 1 percentage-point IV change (BS Vega มาตรฐาน = $S\cdot\sqrt{T}\cdot N'(d_1)$ คือ per 1.00 change ใน σ; หาร 100 เพื่อให้ได้ per 1% move; √T คือ annualized แล้ว | ATM approximation: $N'(d_1) \approx 0.40$)
- Target: **Vega_A ≈ β_IV · Vega_B** ระหว่างสองขา (Vega-neutral)
- ตัวอย่าง: ถ้า Vega_BTC = $200/% และ Vega_ETH = $50/% และ β_IV = 1.05 → ต้องการ BTC: 1 lot, ETH: 200/(50×1.05) ≈ 3.81 lots → rounded = 4 lots
- **Kelly adjustment:** f* × 0.3 เพราะ tail risk ของ short options (ch12)

Vega-neutral position sizing def vega_neutral_size(vega_a, vega_b, beta_iv, kelly_fraction=0.3): # ratio เพื่อให้ Vega_A = beta_iv * Vega_B ratio = vega_a / (vega_b * beta_iv) # ใช้ lots_b = 1 เป็น base → คำนวณ lots_a lots_b = 1.0 lots_a = lots_b * ratio # Kelly: ลด size ลง 70% สำหรับ short vol strategy return { "lots_a": lots_a * kelly_fraction, "lots_b": lots_b * kelly_fraction }

22.8 เชื่อมหนังสือทั้งชุด

บทนี้ต่อยอดจากหลายบทในตำราชุดนี้โดยตรง — ทุก concept เดิมยังใช้ได้ เพียงแต่ *input เปลี่ยนจากราคาเป็น IV*:

หนังสือ / บท	Content ที่ใช้ใน ch22
pm-part1	PCP fundamentals, synthetic construction — พื้นฐาน options ทั้งหมด
arb-part1/arb-part2/arb-part3	Box Spread concept — §22.2 ขยายเพิ่ม stat arb layer บน top
vol-part4	Vol Clustering, IV mean reversion — basis ทางทฤษฎีของ §22.5 VRP
vp-part4	IV Skew Selection Algorithm — §22.3 ใช้โดยตรง
eye-part3	Cross-venue IV gap as arb signal — §22.4 ขยาย full pipeline
statarb-ch3	OU process, half-life — ε_IV ก็ fit OU เหมือนกัน
statarb-ch4	OLS β_IV = Cov/Var — §22.1, §22.4
statarb-ch5	ADF + Pair Selection Pipeline — §22.4 ใช้ full pipeline
statarb-ch10	z-score entry/exit — ใช้กับ ε_IV เหมือนกันทุกประการ
statarb-ch12	Kelly sizing — f* × 0.3 สำหรับ short vol (§22.5, §22.7)
statarb-ch18 §18.8	PCP — §18.8 ε_PCP คือ starting point ที่ ch22 ต่อยอด
statarb-ch19	Breakeven σ framework — §22.6 ขยาย to options friction

บทสรุป: Options Stat Arb = Same Pipeline, New Input

แก่นของ ch22 คือ **ไม่ต้องเรียน pipeline ใหม่** — เพียงแต่เปลี่ยน input จากราคา log(P) เป็น IV และ understand ว่า IV process มี 4 แหล่ง alpha ใหม่:

1. **§22.1** ε_IV = IV cross-asset spread (BTC vs ETH IV)
2. **§22.2** ε_box = Box spread deviation จาก fair value
3. **§22.3** ε_RR = deviation ของ RR_25Δ จาก mean θ_RR (cross-strike IV asymmetry)
4. **§22.4** ε_xvenue = Cross-exchange IV gap
5. **§22.5** ε_VRP = IV − RV premium spike

ทั้งหมดนี้ใช้ ADF, half-life, z-score, Kelly เหมือนกันทุกประการ

แบบฝึกหัด

โจทย์ 22.1 — ε_IV Pipeline

BTC ATM IV (30d) วันนี้ = 62%, ETH ATM IV (30d) = 56%
Historical parameters: β_IV = 1.05, θ_IV = +3%, σ_IV = 2%

(ก) คำนวณ ε_IV,today

(ข) คำนวณ z-score

(ค) ควร trade ไหม? ถ้าใช่ เปิด position ไหน?

▶ ดูคำตอบ

โจทย์ 22.2 — Breakeven IV

Options strategy parameters: total_friction = 250 bps, z_entry = 2.0, z_exit = 0.5
Observed σ_IV ของ ε_IV = 1.8%

(ก) คำนวณ Breakeven σ_IV

(ข) σ_IV = 1.8% ≥ breakeven? trade ได้ไหม? margin of safety เป็นอย่างไร?

▶ ดูคำตอบ

โจทย์ 22.3 — VRP Signal

IV_30d = 80%, RV_30d = 45%
Historical parameters: mean ε_VRP = 15%, σ_VRP = 8%

(ก) คำนวณ ε_VRP และ z-score

(ข) แนะนำ action พร้อม position sizing และ stop-loss

▶ ดูคำตอบ

โจทย์ 22.4 — Box Spread (§22.2)

BTC options บน Deribit: K1 = 90,000, K2 = 95,000 (spread = $5,000), T = 14 วัน, r = 5%/ปี
Quotes: C(90k) = $6,200 · P(90k) = $1,050 · C(95k) = $3,800 · P(95k) = $2,750
Total friction per box = $40

(ก) คำนวณ Fair box value

(ข) คำนวณ Market box price และ ε_box

(ค) ควร Long box หรือ Short box? ระบุ 4 legs และ net edge หลัง friction

▶ ดูคำตอบ

โจทย์ 22.5 — Risk Reversal (§22.3)

BTC S = $90,000, T = 30 วัน. Historical: θ_RR = −8%, σ_RR = 2%
Observe: IV_put(25Δ) = 71%, IV_call(25Δ) = 60%

(ก) คำนวณ RR_25Δ และ ε_RR

(ข) คำนวณ z_RR — ควร trade ไหม?

(ค) ถ้า trade: ระบุ legs และอธิบายว่าต้อง delta-hedge อย่างไร

▶ ดูคำตอบ

Ch.22 Options-Based Stat Arb | ← Ch.21: MT5 Swap Arb | Ch.23: Kalman & t-Distribution →

Statistical Arbitrage · บทที่ 23 — บทเสริม Advanced

Kalman Filter & t-Distribution

Dynamic β_t | Predict-Update Cycle | Fat Tails | Critical Values
สัญญาณซื้อขายใต้ Normal vs t-Distribution
แก่นของบทนี้ — อ่านก่อน
Kalman Filter แก้ปัญหา β_t ที่เปลี่ยนตามเวลา — ปรับ β ทุก tick แทน rolling window
t-Distribution แก้ปัญหา fat tails ของ ε crypto — threshold ที่ถูกต้องกว่า z-score ปกติ
ทั้งสองเทคนิคเสริมกัน: Kalman ให้ β_t ที่แม่นยำ → t-distribution แปลง ε เป็น signal ที่น่าเชื่อถือกว่า
t_stat = ε_t / σ_t → เทียบกับ t_{α/2}(ν) ไม่ใช่ z_{α/2} = 1.96
Prerequisites — ต้องเข้าใจก่อนอ่านบทนี้

- บทที่ 3 §3.3 Half-life และ AR(1) ของ ε
- บทที่ 4 §4.2 OLS สำหรับ β (static β foundation)
- บทที่ 5 §5.6 Shapiro-Wilk, §5.7 Crossing Statistics
- บทที่ 10 §10.3 Robust Z-score (MAD-based)
- บทที่ 22 §22.3 Risk Reversal, §22.5 VRP (ε ที่มี fat tails)

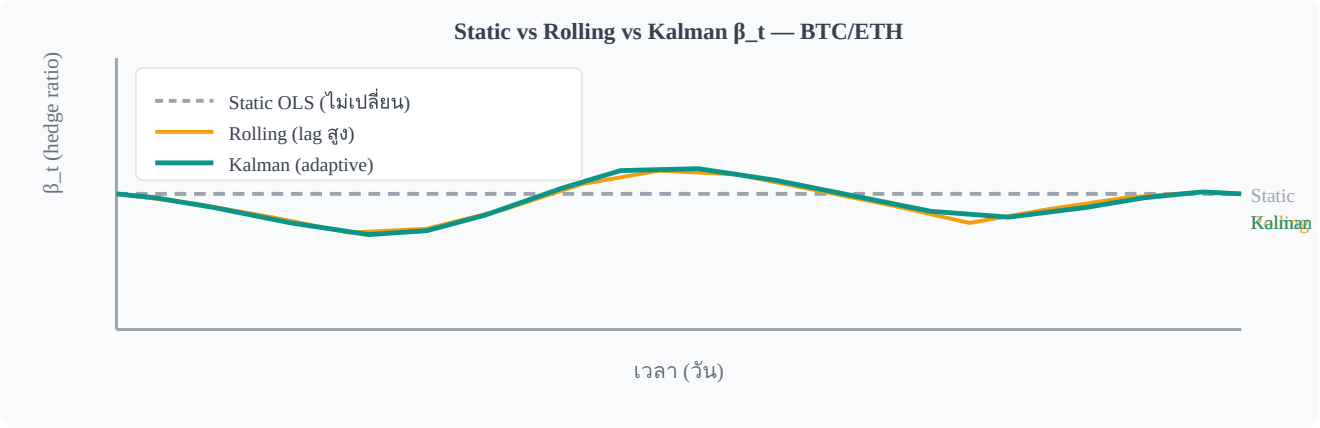
23.1 ปัญหาของ β คงที่ — ทำไม OLS ไม่พอ

ใน บทที่ 4 เราประมาณ β ด้วย OLS บน lookback window เดียว เช่น 250 วัน โมเดลถือว่า β ไม่เปลี่ยนแปลงตลอดช่วงเวลานั้น แต่ในทางปฏิบัติ hedge ratio ของ crypto pairs เช่น BTC/ETH ขยับตามรอบตลาด

สามข้อจำกัดของ Static OLS β

1. **Window selection dilemma:** Window สั้น (30 วัน) → β ตอบสนองเร็วแต่ noisy มาก | Window ยาว (250 วัน) → smooth แต่ lag สูง lag เกิดขึ้นเมื่อระบอบตลาดเปลี่ยน
2. **Regime change blindness:** เมื่อ correlation ระหว่าง BTC/ETH ลดลงชั่วคราว (เช่น altcoin season) OLS β ที่ fit บน data เก่าจะให้ ε ที่ biased — z-score ผิดพลาด ทำให้เข้า trade ผิด side
3. **Abrupt structural break:** Hard fork, exchange hack, macro event → β เปลี่ยนทันที แต่ OLS ใช้เวลาหลายสัปดาห์กว่าจะ update — ช่วงนั้น strategy ขาดทุนทุก trade

Rolling-window β เป็นทางออกแรก: คำนวณ β ใหม่ทุกวันบน lookback window ที่เลื่อนไป แต่ยังเลือก window ได้ยาก และยัง lag



รูป 23.1 — Kalman β_t (เขียว) ตอบสนองรวดเร็วกว่า rolling window (เหลือง) โดยไม่มี lag สูง

Kalman Filter แก้ทั้งสามปัญหาในคราวเดียว: ปรับ β ทุก step, ไม่ต้องเลือก window, และ update ทันทีเมื่อ data ใหม่เข้า

23.2 Kalman Filter — ทฤษฎีและวงจร Predict-Update

Kalman Filter คือ **recursive Bayesian estimator** — ในแต่ละ step เราประมาณ state ล่าสุดจากการผสม prediction จาก model กับ observation ใหม่ น้ำหนักที่ให้แต่ละส่วนขึ้นอยู่กับความไม่แน่นอนสัมพัทธ์ของทั้งสอง

State-Space Model สำหรับ β_t

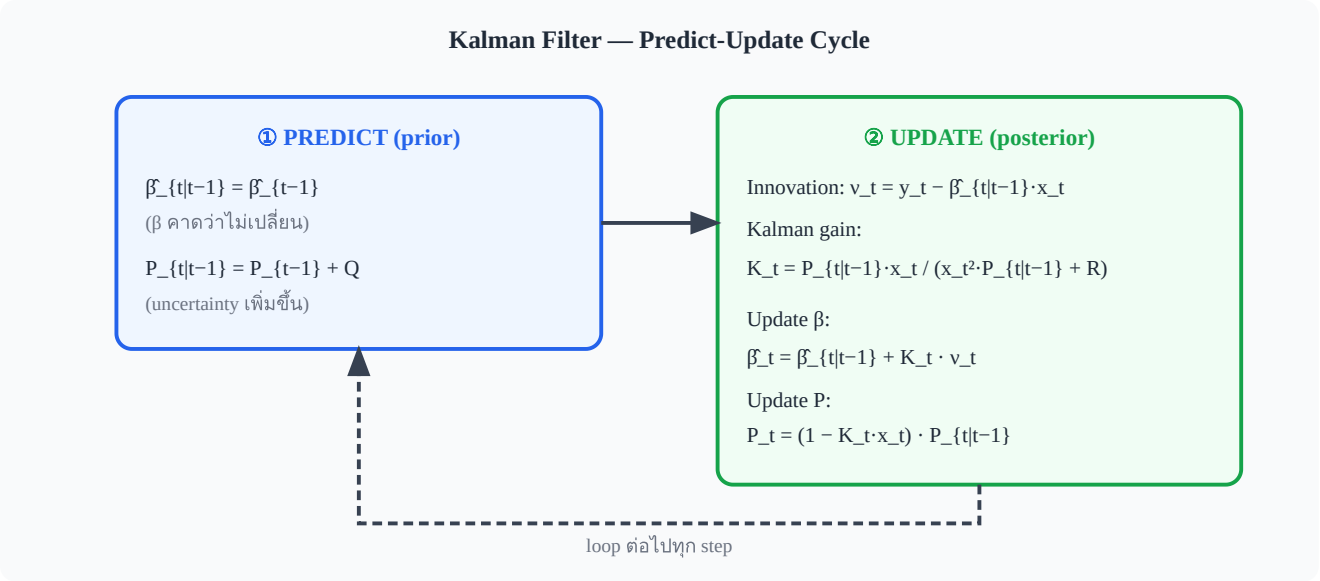
กำหนด state equation และ observation equation:

// State equation (β เปลี่ยนช้าๆ แบบ random walk)
β_t = β_{t−1} + w_t, w_t ~ N(0, Q) // Q = process noise variance

// Observation equation (ราคาที่สังเกตได้)
y_t = β_t · x_t + v_t, v_t ~ N(0, R) // R = observation noise variance

- **Q (Process noise):** ควบคุมความเร็วที่ β_t เปลี่ยนได้ — ค่ามากขึ้น → β ตอบสนองเร็วขึ้นแต่ noisy มากขึ้น
- **R (Observation noise):** สะท้อน noise ใน price relationship — ค่ามากขึ้น → filter เชื่อ observation น้อยลง เชื่อ model มากขึ้น
- **P_t (Error covariance):** ความไม่แน่นอนของ estimate β̂_t — ยิ่งสูง filter ยิ่งให้น้ำหนัก observation ใหม่มากขึ้น

วงจร Predict-Update ทุก time step



รูป 23.2 — วงจร Predict-Update ของ Kalman Filter ทำซ้ำทุก time step

ตีความ Kalman Gain K_t

K_t อยู่ระหว่าง 0 ถึง 1 เสมอ:

- **K_t → 1:** เชื่อ observation ใหม่เกือบทั้งหมด (P สูง หรือ R ต่ำ) → β ปรับตาม data ใหม่มาก
- **K_t → 0:** เชื่อ prediction เกือบทั้งหมด (P ต่ำ หรือ R สูง) → β เปลี่ยนน้อยมาก
- **K_t ≈ 0.5:** ถ่วงน้ำหนักเท่ากัน — filter อยู่ในสมดุล

23.3 Kalman Filter สำหรับ Dynamic β_t — BTC/ETH

นำทฤษฎีในข้อ §23.2 มาใช้กับ BTC/ETH pair จาก บทที่ 5 §5.8 — BTC เป็น y_t (dependent), ETH เป็น x_t (independent)

ตัวอย่างหลัก — BTC/ETH Dynamic β_t

Setup: BTC = y_t, ETH = x_t | Static OLS: $\hat{\beta} = 0.0444$ | $\rho = 0.9745$ | AR(1) $\varphi = 0.9602$ | Half-life = 17h

Kalman parameters: $Q = 1\times10^{-5}$ (β เปลี่ยนช้า), R = 0.001 (noise ปานกลาง)

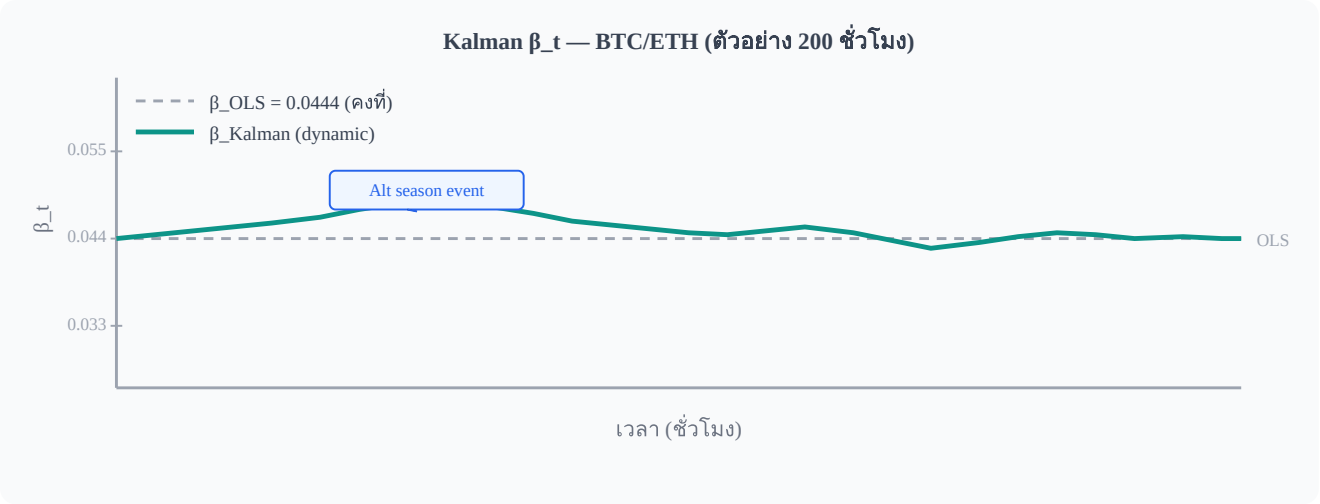
Initial conditions: $\hat{\beta}_0 = 0.0444$ (ค่าจาก OLS), $P_0 = 0.01$

Pseudo-code: Kalman Filter สำหรับ β_t

```
# ตั้งค่าเริ่มต้น
beta_hat = 0.0444 # initial β (จาก OLS)
P = 0.01 # initial error covariance
Q = 1e-5 # process noise (tuning parameter)
```

```
R = 0.001 # observation noise (tuning parameter)
beta_series = []

for t in range(len(prices)):
y_t = log_btc[t]
x_t = log_eth[t]

## ① PREDICT
beta_prior = beta_hat # β̂_{t|t-1}
P_prior = P + Q # P_{t|t-1}

## ② UPDATE
innovation = y_t - beta_prior * x_t # ν_t (residual)
S = x_t**2 * P_prior + R # innovation variance
K = P_prior * x_t / S # Kalman gain
beta_hat = beta_prior + K * innovation
P = (1 - K * x_t) * P_prior

## คำนวณ ε_t ด้วย β_t ที่ update แล้ว
epsilon_t = y_t - beta_hat * x_t
beta_series.append(beta_hat)
```

หลังจากวน loop เสร็จ `beta_series` คือ time series ของ β_t ที่ปรับตาม data ทุก step โดยอัตโนมัติ



รูป 23.3 — β_t ของ Kalman Filter (เขียว) เลื่อนขึ้นระหว่าง altcoin season แล้วกลับสู่ระดับปกติ ขณะที่ OLS คงที่ตลอด

ข้อดีเชิงปฏิบัติ — ε_t ที่แม่นยำกว่า

เมื่อใช้ Kalman β_t แทน OLS β:

- ε_t = y_t − β_t·x_t มี **variance ต่ำกว่า** และ **mean ≈ 0 สม่ำเสมอกว่า**
- z-score threshold ยังใช้เดิม (±2σ) แต่ **false signals ลดลง** เพราะ ε ไม่มี drift จาก stale β
- Half-life ของ ε_t สั้นลง → reversion เร็วขึ้น → สามารถ trade ได้ถี่ขึ้นหรือ hold time ลดลง

23.4 ข้อจำกัดและการ Tune Kalman Filter

Kalman Filter มีพลัง แต่ต้องเข้าใจข้อจำกัดก่อนนำไปใช้จริง

⚠ ข้อจำกัดสำคัญ 4 ข้อ

1. **Q/R tuning:** ไม่มีสูตรตายตัวสำหรับ Q และ R — ต้องทำ walk-forward validation (ch3 §3.9) เพื่อหาค่าที่ให้ Sharpe ratio สูงสุดบน out-of-sample data
2. **Initial conditions sensitivity:** ค่าเริ่มต้น β_0 และ P_0 ส่งผลต่อ 50–100 step แรก — burn-in period นี้ควรตัดออกจาก live trading ค่อยๆ converge
3. **Linearity assumption:** Model นี้สมมติ β เปลี่ยนแบบ linear random walk (Gaussian) — ถ้า jump discontinuity เกิดขึ้น (เช่น delisting) Kalman จะ overreact หรือ underreact
4. **Single-factor limitation:** Filter ข้างบน estimate β เดียว (slope) โดยถือ intercept ≈ 0 สำหรับ model ที่มีทั้ง α และ β ต้องขยายเป็น state vector 2×1 (Kalman เวอร์ชัน multivariate)

แนวทาง Tune Q และ R

Parameter	ค่าต่ำ	ค่าสูง	Rule of thumb
Q (process noise)	β เปลี่ยนช้า — smooth แต่ lag	β ตอบสนองเร็ว — แต่ noisy	เริ่มที่ 1e-5 ถึง 1e-4; หาจาก grid search
R (observation noise)	เชื่อ price มาก — β วิ่งตาม	เชื่อ model มาก — β stable	ประมาณจาก Var(ε) ของ OLS residual
P_0 (initial P)	confident เรื่อง β_0	uncertain เรื่อง β_0	ถ้าไม่แน่ใจ ใส่ $P_0 = 1.0$ แล้ว burn-in

EM Algorithm สำหรับ Q/R (Advanced)

สามารถประมาณ Q และ R จาก data โดยตรงด้วย **Expectation-Maximization (EM)** บน log-likelihood ของ state-space model — ไม่ต้อง tune ด้วยมือ แต่ต้องการ training window ยาวพอ (อย่างน้อย 500+ observations) ห้องสมุด Python ที่ใช้ได้: `pykalman`, `statsmodels.tsa.statespace`

เส้นทางจาก §23.1–23.4 ครอบคลุม Dynamic β ด้วย Kalman Filter ครบถ้วน ส่วนต่อไป §23.5–23.8 จะแก้อีกปัญหาหนึ่งที่เกี่ยวข้องกัน — fat tails ของ ε ที่ทำให้ z-score แบบ Normal ไม่น่าเชื่อถือ

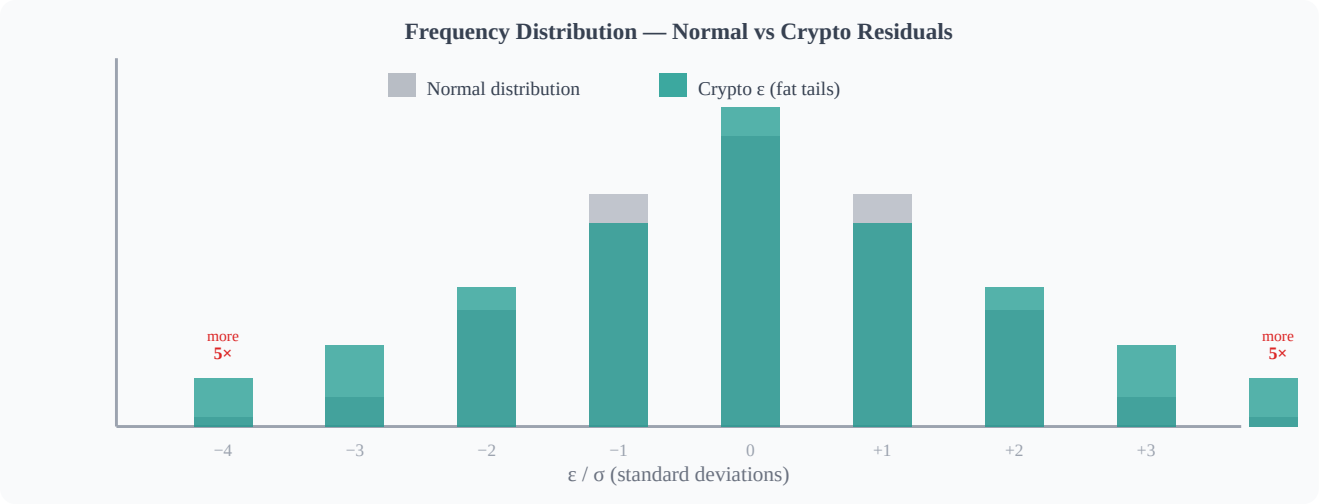
23.5 ปัญหา Fat Tails ใน Crypto Residuals

ε ของ crypto pairs และ options residuals (เช่น ε_RR จาก §22.3) มักมีหางที่หนากว่าการแจกแจงปกติมาก — ปัญหานี้ทำให้ z-score ที่คำนวณจาก Normal distribution ให้ false signals มากเกินจริง

Kurtosis — วัดความ "heavy" ของหาง

Kurtosis (excess) = E[(ε/σ)⁴] − 3

- **Normal distribution:** excess kurtosis = 0 (kurtosis = 3)
- **BTC/ETH residuals:** excess kurtosis ≈ 3–9 (kurtosis 6–12) — เหตุการณ์ 4σ เกิดถี่กว่า Normal หลายเท่า
- **ε_RR (Risk Reversal):** excess kurtosis ≈ 5–12 — IV spike ระหว่าง fear events สร้าง extreme ε



รูป 23.4 — ที่ ±4σ crypto residuals เกิดถึง 5× บ่อยกว่า Normal — z-score 2.0 แบบปกติให้ false entries มากเกิน

ผลต่อ Trading Signals

ถ้าใช้ z-score threshold = ±2.0 โดยสมมติ Normal distribution:

- ตาม Normal: P(|z| > 2.0) = 4.6% → คาด 4.6% ของ observations เป็น signal
- สำหรับ crypto ε ที่มี excess kurtosis = 6: P(|z| > 2.0) ≈ 8–12% จริงๆ → **signals เกือบ 2–3× มากเกินที่ควรจะเป็น**
- ส่วนใหญ่ที่เกินมาคือ **false signals** จากหาง → เข้า trade ตอน momentum ยังแรง แล้วขาดทุน

ทางออก: ใช้ **t-distribution ที่มี degrees of freedom ν ต่ำ** เพื่อสะท้อน fat tails ที่แท้จริง

23.6 t-Distribution — สูตรและ Degrees of Freedom

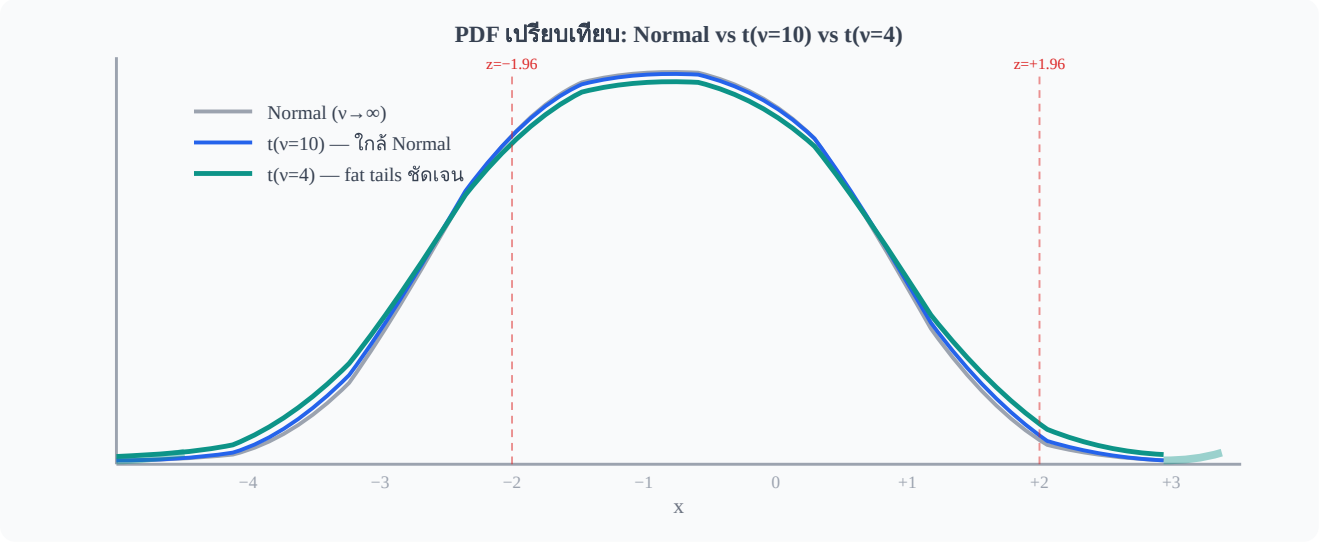
t-distribution (Student's t) มีรูปทรงคล้าย Normal แต่มีหางหนักกว่าที่ปรับได้ด้วยพารามิเตอร์ **degrees of freedom ν (nu)**

Probability Density Function

$$f(t ; \nu) = \Gamma((\nu+1)/2) / (\sqrt{(\nu\pi)} \cdot \Gamma(\nu/2)) \cdot (1 + t^2/\nu)^{-(\nu+1)/2}$$

เมื่อ $\nu \to \infty$: $f(t;\nu) \to$ Normal PDF N(0,1)

เมื่อ ν ↓ 1 : หางหนักมาก (Cauchy distribution)
สำหรับ crypto: ν ≈ 3–6 ให้ผลที่ fit data ดีที่สุด



รูป 23.5 — t(ν=4) มีหางหนักกว่า Normal ชัดเจน — area นอก ±1.96 ใหญ่กว่า 2× สำหรับ crypto ν ≈ 3–6

คุณสมบัติสำคัญของ t-distribution

ν (degrees of freedom)	ลักษณะ	Excess Kurtosis	Variance
ν = 2	หางหนักมาก	ไม่มีที่สิ้นสุด	ไม่มีที่สิ้นสุด
ν = 3	หางหนาก	ไม่มีที่สิ้นสุด	3.0
ν = 4	fat tails — เหมาะ crypto	6.0	2.0
ν = 6	moderate fat tails	3.0	1.5
ν = 10	ใกล้ Normal	1.0	1.25
ν → ∞	Normal distribution	0	1.0

ตัวอย่างหลัก — ε_RR (Risk Reversal) Options Residual

ε_RR จาก §22.3 ของ BTC options มี distribution ที่ fat-tailed เนื่องจาก IV spikes ระหว่าง market stress:

- BTC 30-day IV สามารถ jump จาก 50% → 120% ภายใน 48 ชั่วโมง (Black Thursday 2020, FTX collapse 2022)
- MLE estimate ν ≈ 4–5 สำหรับ ε_RR บน 1-year data
- ถ้าใช้ Normal threshold z = 2.0 → จะ flag ε_RR ที่ IV spike ว่าเป็น "entry signal" ทั้งที่จริงๆ คือ regime change ที่ควรหลีกเลี่ยง
- t(ν=4) threshold = 2.78 → กรองสัญญาณ IV spike ออกไปได้ดีกว่า

23.7 Critical Values — Normal vs t-Distribution

ความแตกต่างใน threshold นี้มีผลโดยตรงต่อจำนวนครั้งที่ strategy เข้า trade

Distribution	Two-tailed α=5%	Two-tailed α=2%	Two-tailed α=1%
Normal (z)	**1.960**	2.326	2.576
t (ν = 30)	2.042	2.457	2.750
t (ν = 10)	2.228	2.634	3.169
t (ν = 6)	2.447	2.969	3.707
t (ν = 4)	**2.776**	3.747	4.604
t (ν = 3)	3.182	4.541	5.841

ตีความตาราง

ถ้าใช้ t(ν=4) แทน Normal สำหรับ two-tailed α=5% (threshold แบบ "95% confidence"):

- Normal threshold: |z| ≥ **1.96**
- t(ν=4) threshold: |t| ≥ **2.78** → เข้มกว่า 42%
- ความหมาย: ε ต้องห่างจาก mean มากกว่า 42% จึงจะเรียกว่า "significant" — กรอง noise ออกมากขึ้น

การประมาณ ν จากข้อมูล (MLE)

หา ν ที่เหมาะสมกับ ε โดยใช้ Maximum Likelihood Estimation:

```
from scipy.stats import t as t_dist
import numpy as np

# MLE fit ของ t-distribution บน ε_RR หรือ ε BTC/ETH
epsilon = np.array([...]) # residuals series

# fit returns (df, loc, scale) — df คือ ν ที่ fit ดีที่สุด
nu, loc, scale = t_dist.fit(epsilon, floc=0) # fix loc=0 (mean-zero ε)

print(f"Estimated ν = {nu:.2f}") # ตัวอย่าง: ν ≈ 4.2

# critical value สำหรับ two-tailed 5%
threshold = t_dist.ppf(0.975, df=nu) # ตัวอย่าง: 2.74
print(f"Use threshold ±{threshold:.3f} instead of ±1.960")
```

ตัวอย่างหลัก — MLE บน ε_RR BTC Options

Input: ε_RR series จาก 1 ปีของ BTC 25Δ Risk Reversal (≈250 daily observations)

MLE result: $\hat{\nu} \approx 4.3$ → ยืนยันว่า fat-tailed

New threshold: t_{0.025}(ν=4.3) ≈ 2.72 (แทน 1.96)

ผลต่อ strategy:

- Signals ลดลงจาก ≈28 ครั้ง/ปี (Normal) → ≈17 ครั้ง/ปี (t-dist)
- False positive rate ลดลงจาก ≈8% → ≈4.8% (ใกล้เคียงทางทฤษฎี α=5% จริงๆ)
- Win rate ต่อ trade สูงขึ้น แต่ trade น้อยลง — net Sharpe ratio ดีขึ้น

23.8 เปรียบเทียบ Trading Signals: Normal vs t-Distribution

ส่วนนี้แสดงผลต่างเชิงปฏิบัติของการใช้ Normal vs t-distribution ในการสร้าง trading signals บน pipeline เดิม

Scenario	Normal (z=2.0)	t(ν=4, t=2.78)	ผล
ε_RR = −12% (σ_RR=2%, θ=−8%)	z = (−12−(−8))/2 = −2.0 → **ENTER**	t = −2.0 < 2.78 → **WAIT**	t กรอง marginal signal
ε_RR = −15% (σ_RR=2%, θ=−8%)	z = −3.5 → **ENTER**	t = −3.5 > 2.78 → **ENTER**	ทั้งคู่เห็นตรงกัน
ε_BTC = +2.3σ (half-life event)	z = 2.3 → **ENTER**	t = 2.3 < 2.78 → **WAIT**	t กรอง noise trade
ε_BTC = +3.1σ (IV spike event)	z = 3.1 → **ENTER**	t = 3.1 > 2.78 → **ENTER**	ทั้งคู่เห็นตรงกัน
ε_PCP = −1.8σ (PCP violation)	z = 1.8 → **WAIT**	t = 1.8 → **WAIT**	ทั้งคู่ผ่าน — ถูกต้อง

สรุปผลต่างเชิงปฏิบัติ

- **t-distribution ช่วยลด false positives** โดยเฉพาะสัญญาณที่ z อยู่ในโซน 2.0–2.7 — ซึ่งใน crypto เป็น noise มากกว่า signal
- Signals ที่ผ่าน t-threshold มักอยู่ใกล้ขอบเขต extreme ของ distribution จริงๆ → higher conviction trades
- สำหรับ **options ε (ε_RR, ε_PCP)**: ควรใช้ t เสมอเพราะ fat tails จาก IV dynamics รุนแรงกว่า spot ε
- สำหรับ **spot pair ε** ที่ผ่าน Shapiro-Wilk (ch5 §5.6): ใช้ Normal ก็ได้ — แต่ถ้า kurtosis > 5 → เปลี่ยนเป็น t

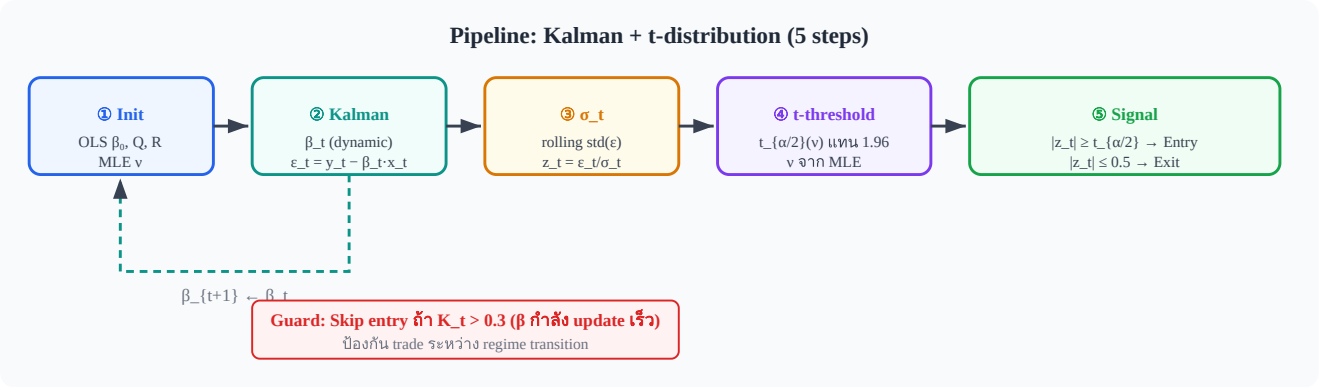
§23.5–23.8 แสดงว่า t-distribution ให้ threshold ที่สะท้อน tail risk ของ crypto จริงมากกว่า ส่วนต่อไป §23.9 จะรวม Kalman Filter กับ t-distribution เป็น pipeline เดียว

23.9 Pipeline รวม: Kalman β_t + t-Distribution Signal

Pipeline ต่อไปนี้รวม 2 เทคนิคจาก §23.1–23.8 เข้าด้วยกัน ให้ signal ที่แม่นยำและน่าเชื่อถือกว่า static β + Normal threshold

5-Step Pipeline: Kalman + t-distribution

1. **Initialization:** OLS β บน training window (≥250 obs) → ได้ β_0 เริ่มต้น และ Var(ε) สำหรับ R; ตั้ง Q จาก walk-forward validation
2. **Kalman update (ทุก tick/bar):** รัน predict-update cycle → ได้ β_t และ ε_t = y_t − β_t·x_t
3. **Estimate σ_t:** คำนวณ rolling std ของ ε_t (lookback 50–100 bars) → ได้ z_t = ε_t / σ_t
4. **t-distribution threshold:** ใช้ ν ที่ประมาณจาก MLE บน training window → threshold = t_{α/2}(ν) แทน 1.96
5. **Entry/Exit signal:**
   - Entry: |z_t| ≥ t_{α/2}(ν) AND Kalman gain K_t < 0.3 (β stable, ไม่อยู่ระหว่าง fast update)
   - Exit: |z_t| ≤ 0.5 (ε กลับ mean)
   - Stop: |z_t| ≥ 3 × t_{α/2}(ν) (ε วิ่งไกลเกิน — ไม่ใช่ reversion)



รูป 23.6 — Pipeline 5 ขั้นตอน: Kalman ให้ β_t แม่นยำ → t-distribution กำหนด threshold ที่เหมาะ

Pseudo-code: Pipeline รวม

```
# === Initialization (ครั้งเดียวบน training data) ===
beta_hat, P = ols_fit(y_train, x_train) # β0, P0
Q, R = tune_kalman(y_train, x_train) # walk-forward
nu = mle_t_fit(epsilon_train) # ν จาก MLE
threshold = t_dist.ppf(0.975, df=nu) # เช่น 2.74
epsilon_buf = [] # rolling buffer สำหรับ σ_t

# === Live Trading Loop ===
for bar in live_stream:
y_t, x_t = bar['log_y'], bar['log_x']

## ② Kalman update
P_prior = P + Q
innovation = y_t - beta_hat * x_t
S = x_t**2 * P_prior + R
K = P_prior * x_t / S
beta_hat = beta_hat + K * innovation
P = (1 - K * x_t) * P_prior
epsilon_t = y_t - beta_hat * x_t

## ③ σ_t estimate
```

```
epsilon_buf.append(epsilon_t)
if len(epsilon_buf) > 100: epsilon_buf.pop(0)
sigma_t = std(epsilon_buf)
z_t = epsilon_t / sigma_t

## ⑤ Signal
if abs(z_t) >= threshold and K < 0.3: # Entry
signal = 'LONG_Y' if z_t < 0 else 'SHORT_Y'
elif abs(z_t) <= 0.5: # Exit
signal = 'CLOSE'
elif abs(z_t) >= 3 * threshold: # Stop
signal = 'STOP'
```

23.10 Cross-Reference ทั้งเล่ม

บทนี้สร้างขึ้นบนฐานของหลายบทก่อนหน้า และเพิ่มเติมให้บทที่จะใช้ต่อ:

บท	เชื่อมโยงกับ ch23
บทที่ 3 §3.9	Walk-forward calibration — ใช้สำหรับ Q/R tuning ของ Kalman
บทที่ 4 §4.2b	OLS β เป็นค่าเริ่มต้น β_0 ของ Kalman Filter
บทที่ 5 §5.6	Shapiro-Wilk: ถ้า $p \geq 0.05$ ใช้ Normal; ถ้า $p < 0.05$ ใช้ t-distribution
บทที่ 5 §5.7	Crossing rate φ: AR(1) coefficient เดียวกับที่ใช้ประมาณ Half-life
บทที่ 10 §10.3	Robust Z-score (MAD): ทางเลือกแทน σ-based z เมื่อ outlier มาก
บทที่ 12	Kelly criterion: ปรับ position size ตาม win rate จาก t-distribution threshold
บทที่ 18 §18.8	ε_PCP: ตัวอย่าง fat-tailed options residual ที่ใช้ t-distribution
บทที่ 22 §22.3	ε_RR: Risk Reversal residual — ตัวอย่างหลักสำหรับ MLE ν ใน §23.7
บทที่ 22 §22.5	ε_VRP: Vol Risk Premium — ประโยชน์จาก Kalman เพราะ β_IV เปลี่ยนตาม term structure

โจทย์บทที่ 23

โจทย์ 23.1 — Kalman Filter 1 Step

ให้ค่า: $\hat{\beta}_0 = 0.0444$, $P_0 = 0.01$, $Q = 1\times10^{-5}$, $R = 0.001$
Data ใหม่: $y_1 = 10.520$ (log BTC), $x_1 = 7.953$ (log ETH)

(ก) คำนวณ P_{1|0} (prior covariance)
(ข) คำนวณ Kalman gain K_1
(ค) คำนวณ $\hat{\beta}_1$ (updated estimate) และ ε_1

▶ ดูคำตอบ

โจทย์ 23.2 — MLE ประมาณ ν

ε_{RR} ของ BTC 25Δ Risk Reversal มีสถิติต่อไปนี้จาก 252 observations:
Mean = 0.0%, Std = 2.1%, Skewness = −0.3, **Kurtosis (excess) = 5.8**

(ก) จาก kurtosis = 5.8 ประมาณ ν คร่าวๆ ได้เท่าไร? (ใช้สูตร excess kurtosis = $6/(\nu-4)$ สำหรับ $\nu > 4$)
(ข) ถ้า $\nu \approx 5$ ให้คำนวณ two-tailed threshold ที่ α=5% (ใช้ตารางใน §23.7)
(ค) เปรียบเทียบกับ Normal z = 1.96 — threshold ต่างกันกี่ %?

▶ ดูคำตอบ

โจทย์ 23.3 — เปรียบเทียบ Normal vs t Signal

BTC/ETH pair: $\varepsilon_t = +2.4\sigma$ (คำนวณจาก Kalman β_t), $\nu = 4$ สำหรับ t-distribution

(ก) Normal (threshold = 1.96): เข้า trade หรือ wait?
(ข) t(ν=4) (threshold = 2.78): เข้า trade หรือ wait?
(ค) ถ้าหลังจากนี้ ε เพิ่มขึ้นเป็น +3.2σ — ทั้งสองระบบตัดสินใจอย่างไร?

▶ ดูคำตอบ

โจทย์ 23.4 — Full Pipeline

ให้ข้อมูล BTC/ETH: ε_{RR} series มี $\nu = 4$, $\sigma_{RR} = 2\%$, $\theta_{RR} = -8\%$
Observe $\varepsilon_{RR} = -14.5\%$ ($RR_{25\Delta} = -14.5\%$) พร้อมกับ Kalman gain $K_t = 0.18$

(ก) คำนวณ z_t (ใช้ mean-adjusted ε)
(ข) เปรียบเทียบกับ t(ν=4) threshold = 2.776 — ควร enter trade ไหม?
(ค) ถ้า $K_t = 0.45$ แทน — pipeline ตัดสินใจอย่างไร?

▶ ดูคำตอบ

Ch.23 Advanced: Kalman Filter & t-Distribution | ← Ch.22: Options-Based Stat Arb