GRID TRADING MASTERY · PART 1A

Close System + Zone Tricks + Soft Martingale

ระบบปิด (ไม่มี SL) · Zone Waterfall · Bell Curve Density · Capital Front-Loading · r-parameter

แนวคิดจาก: Close System (พี่ต้าน Mudley Group)

แก่นของ PART นี้

Close System หมายถึง: วาง capital ทั้งหมดที่ต้องการ *ก่อน* เข้าเล่น จนถึง "floor" หรือ give-up point — ไม่มี stop-loss นอกจากทุนหมด ข้อได้เปรียบ: ไม่โดน stop hunt, ไม่ต้อง top-up กลางทาง, รู้ maximum loss ล่วงหน้า

```
C_floor = Σ(i=1 to N) Q_i × P_i [ทุน reserve ถึง floor]
```

1A.1 Close System คืออะไร

Grid ส่วนใหญ่ที่สอนกันทั่วไปมี stop-loss — ถ้าราคาลงถึงจุดหนึ่งก็ปิด position ยอมขาดทุน

Close System คือแนวคิดตรงกันข้าม:

หลักการ Close System (ระบบปิด)

1. **กำหนด give-up point (floor)** — จุดที่ถ้าราคาลงถึง คุณยอมรับการถือ inventory ทั้งหมด (หรือ capital หมด)
2. **วาง capital ทั้งหมด ล่วงหน้า** — คำนวณว่าต้องใช้เงินเท่าไหร่ถ้าราคาลงถึง floor และวาง capital นั้นไว้ก่อน
3. **ไม่มี stop-loss ระหว่างทาง** — ยกเว้น capital หมดถึง floor เท่านั้น
4. **รู้ worst-case ล่วงหน้า** — maximum loss = capital ที่วางไว้ทั้งหมด (ไม่มีเซอร์ไพรส์)

ทำไมไม่มี SL กลาง grid?

ปัญหาของ SL กลาง grid: ถ้า BTC ลงจาก $100k → $95k → SL triggered → ขาดทุน $5k → จากนั้น BTC ฟื้นกลับ $105k — คุณเสียโอกาสทำกำไรทั้งหมด

Close System ไม่มี SL กลาง → ถ้า BTC ลง $95k ก็รอ → ถ้ากลับ $105k → กำไร | ถ้าไม่กลับเลย → ขาดทุนเท่า capital ที่ reserve ไว้

Trade-off: ยอมรับ maximum loss ที่ชัดเจน เพื่อแลกกับโอกาสไม่โดน stop hunt

1A.2 Floor Capital — การคำนวณ

Floor = ราคาต่ำสุดที่ยอมรับให้ grid ทำงานถึง (เช่น $80,000 สำหรับ BTC ที่ $100,000)

```
สูตร Capital ที่ต้องการถึง Floor (Equal Weight):
C_floor = Q × Σ(P_i)   สำหรับทุก i ที่ P_i อยู่ระหว่าง [floor, entry]

ตัวอย่าง: BTC entry = $100k, floor = $90k, step = $2k, Q = 0.01 BTC/level
  Levels: $98k, $96k, $94k, $92k, $90k  (5 levels ด้านล่าง)
  C_floor = 0.01 × ($98k + $96k + $94k + $92k + $90k)
          = 0.01 × $470k
          = $4,700

นี่คือทุน "Reserve" ที่ต้องมีก่อนเปิด grid
ถ้าไม่มีครบ $4,700 → ไม่เปิด grid (ไม่ใช่ลดขนาด order)

สูตร Soft Martingale (ขนาด Q เพิ่มขึ้นตาม level):
Q_i = Q_0 × r^(i-1)   โดย r ∈ [1.1, 1.5], i = 1,2,3,...,N
C_floor_martingale = Σ(i=1 to N) Q_i × P_i
                   = Σ (Q_0 × r^(i-1)) × P_i   [ต้องคำนวณ case-by-case]
```

🔁 **Running Example: คำนวณ Floor Capital**

Level	ราคา	Q (BTC)	Capital ที่ใช้	Cumulative
Entry	$100,000	—	—	—
L1	$98,000	0.010	$980	$980
L2	$96,000	0.010	$960	$1,940
L3	$94,000	0.010	$940	$2,880
L4	$92,000	0.010	$920	$3,800
L5 (floor)	$90,000	0.010	$900	**$4,700**

สรุป: ต้องมีทุน $4,700 reserve ก่อนเปิด grid | ถ้า BTC ลงถึง $90k โดยไม่กลับ → unrealized loss = $4,700 – (0.05 BTC × $90k) = $4,700 – $4,500 = $200 เท่านั้น เพราะถือ BTC ไว้

1A.3 Zone Waterfall — Active / Standby / Floor

ปัญหาใหญ่ของ Close System: ต้องวาง capital ทั้งหมดไว้เฉยๆ (idle) รอราคาลงถึง — **Zone Waterfall แก้ปัญหานี้**

ZONE WATERFALL — ACTIVE / STANDBY / FLOOR

ACTIVE ZONE — 50%
Deploy เป็น Grid Orders ตอนนี้

▼

STANDBY ZONE — 30%
ฝาก lending / T-bill รับดอกเบี้ย

▼

FLOOR ZONE — 20%
สำรองฉุกเฉิน — ไม่แตะยกเว้นกรณีสุดท้าย

การไหลของทุน

Grid fill → คืนทุน + กำไร
ส่งกลับมาใช้ Active Zone ใหม่

↓

กำไรส่วนเกิน → Standby
ฝากรับดอกเบี้ย 4–8% APY

↓

Floor: ไม่แตะจนถึง
give-up point (Close System)

Active (50%) + Standby (30%) + Floor (20%) = 100% — ทุนทุกบาทมีบทบาท

ไม่ต้องใช้ทุนทั้งหมดเป็น grid orders — Standby หาดอกเบี้ยระหว่างรอ

วิธีทำงานของ Zone Waterfall

1. เริ่มต้น: Active Zone deploy grid orders ตาม price range ปัจจุบัน
2. เมื่อราคาลง → Active Zone depletes → Standby เลื่อนมาเป็น Active ใหม่
3. เมื่อ Standby depletes → Floor เลื่อนมา (นี่คือ warning signal)
4. เมื่อ Floor depletes → Close System ถึง give-up point → หยุด grid

ข้อดี: Standby Zone ฝากรับดอกเบี้ยได้ขณะรอ → ลด opportunity cost ของ capital ที่ idle

```
def zone_waterfall_capital(total_capital, zone_ratios=(0.5, 0.3, 0.2)):
    """คำนวณ capital ใน each zone"""
    active_ratio, standby_ratio, floor_ratio = zone_ratios
```

```
    active_capital   = total_capital * active_ratio   # deploy เป็น
orders
    standby_capital  = total_capital * standby_ratio  # ฝาก
lending/yield
    floor_capital    = total_capital * floor_ratio    # ไม่แตะ

    # Standby yield (สมมติ 5% APY)
    standby_yield_daily = standby_capital * 0.05 / 365
    return {
        "active": active_capital,
        "standby": standby_capital,
        "floor": floor_capital,
        "standby_daily_yield": standby_yield_daily
    }

# ตัวอย่าง: total = $10,000
# active=$5,000 · standby=$3,000 · floor=$2,000
# standby_daily_yield = $3,000 × 5%/365 = $0.41/วัน (ลด opportunity
cost)
```

> ⚠️ **Counterparty Risk ของ Lending Protocol**
>
> Yield 4–8% APY จาก Aave/Compound/lending platforms **ไม่ใช่ risk-free**:
>
> - **Smart contract risk:** bug หรือ exploit → เงินอาจหายทั้งหมด (เกิดกับ Euler Finance $197M ปี 2023)
> - **Liquidity risk:** ในช่วง market crash → ถอน USDT ออกไม่ได้ทันที (utilization rate สูง)
> - **เหมาะสำหรับ:** Standby capital ที่รอ > 30 วัน และยอมรับ smart contract risk ได้
> - **ทางเลือกที่ safer กว่า:** CEX savings product (Binance Earn, OKX Earn) — counterparty risk ต่ำกว่าแต่ yield ต่ำกว่า

1A.4 Buy-Only Zone Asymmetry — Trick ที่ทรงพลัง

ZONE TRICK #1 — ASYMMETRIC ZONE

Grid ทั่วไป: วาง zone −2σ ถึง +2σ (symmetric)

Zone Trick: วาง zone −2σ ถึง *+1σ* (ไม่ถึง +2σ) สำหรับ Buy-Only Grid

ทำไม: ราคาอยู่เหนือ +1σ น้อยมาก — วาง grid สูงเกิน +1σ เหมือน "ซื้อบนดอย" ที่ไม่ค่อย fill หรือ fill แล้วราคาไม่กลับลงมา

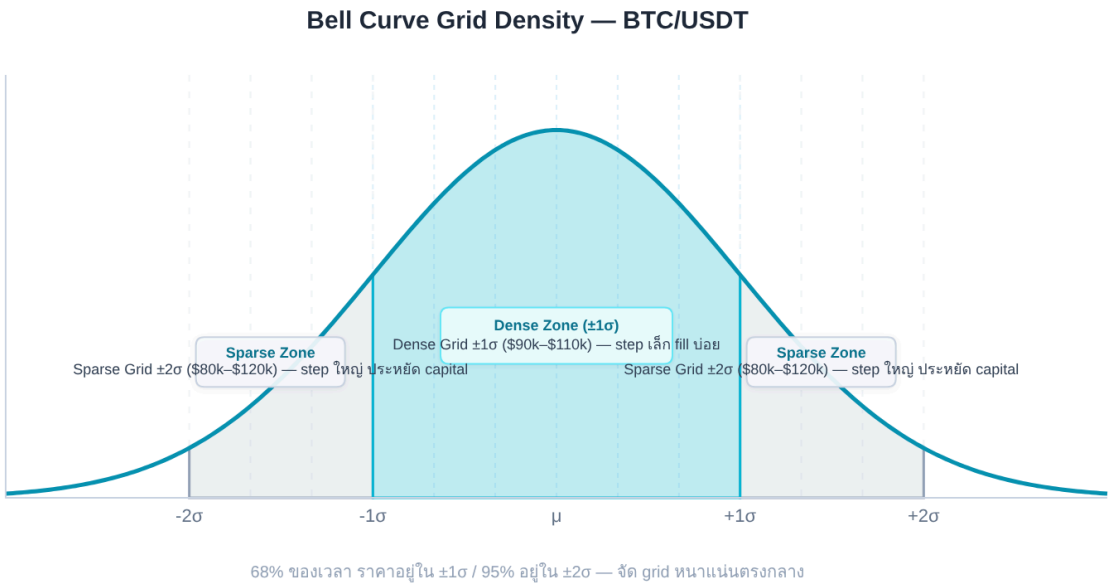
ข้อดีของ −2σ ถึง +1σ: ประหยัดทุนได้ ~25% (ตัดส่วนบนออก) + ลด inventory risk บนสุด

```
สมมติ BTC mean = $100k, σ = $10k (10%)
Zone ทั่วไป (−2σ ถึง +2σ): $80k ถึง $120k  (range $40k)
Zone Trick   (−2σ ถึง +1σ): $80k ถึง $110k  (range $30k)

Capital ที่ประหยัด: Grid levels $110k-$120k ไม่ต้องวาง
  → ถ้า step = $2k → ประหยัด 5 levels × capital ต่อ level
  → ≈ 25% less capital สำหรับ upper zone ที่ "ไม่ค่อย fill"

ทดสอบ: ในช่วง 1 ปี BTC อยู่เหนือ +1σ เท่าไหร่?
  → ตามทฤษฎีปกติ: ~16% ของเวลา (1 − 84th percentile)
  → จริงๆ BTC: ≈ 15-20% (ใกล้เคียง) เพราะ crypto skewed right
  → สรุป: 80% ของเวลาราคาอยู่ใน −2σ ถึง +1σ = Zone ที่เลือก
```

1A.5 Bell Curve Grid Density



Grid หนาแน่นที่ mean — Step เล็กในโซนที่ราคาอยู่บ่อย ประหยัด capital ในโซนที่ราคามาน้อย

ZONE TRICK #2 — VARIABLE STEP (BELL CURVE DENSITY)

แทนที่จะใช้ step เท่ากันทุก level ลองปรับ step ตามระยะห่างจาก mean:

- ใกล้ mean (±0.5σ): step เล็ก → fill บ่อย → cashflow ดี
- ห่าง mean (1σ–2σ): step ใหญ่ขึ้น → fill น้อย → ประหยัด capital

```
def variable_step(price, mean, sigma, base_step):
    """คำนวณ step size ตามระยะห่างจาก mean (bell curve density)"""
    z = abs(price - mean) / sigma  # z-score ของ price
    multiplier = 1 + z * 0.5      # step เพิ่ม 50% ต่อ 1σ ที่ห่างออกไป
    return base_step * multiplier

# ตัวอย่าง: mean=$100k, σ=$10k, base_step=$2k
# ที่ $100k (z=0):   step = $2,000 × 1.0  = $2,000  (dense)
# ที่ $95k  (z=0.5): step = $2,000 × 1.25 = $2,500
# ที่ $90k  (z=1.0): step = $2,000 × 1.5  = $3,000
# ที่ $80k  (z=2.0): step = $2,000 × 2.0  = $4,000  (sparse)

# ผลลัพธ์: cashflow จาก center สูง, capital ที่ extremes น้อย
```

ข้อดีของ Bell Curve Density

- Cashflow concentration: 70%+ ของรอบ grid อยู่ใน ±1σ จาก mean → step เล็กในโซนนี้ให้ cashflow สูง
- Capital efficiency: ไม่ต้องใส่ทุนเยอะที่ extreme level ที่ fill น้อย
- Natural risk reduction: ถ้าราคาออกไปที่ extreme, step ใหญ่ → ซื้อน้อยกว่าในวิธี equal step

1A.6 Capital Front-Loading — ยืมจาก Upper Zone

ZONE TRICK #3 — CAPITAL FRONT-LOADING

ถ้าราคาอยู่ที่ mean และ upper zone ว่างเปล่า (ไม่มี sell order fill) — ทำไมต้องปล่อยทุนนั้นนอนเฉยๆ?

Idea: ยืม capital จาก upper zone (ที่ยังไม่ถูกใช้) มาเปิด grid ในราคาปัจจุบัน — เมื่อ sell order fill ที่ upper zone ทีหลัง → คืนทุนอัตโนมัติ

```
ตัวอย่าง Capital Front-Loading:
  Grid zone: $90k ถึง $110k · Step $2k · Q = 0.01 BTC · ราคาปัจจุบัน =
$100k

  ปกติ:
    - BUY orders ที่ $98k, $96k, $94k, $92k, $90k (ด้านล่าง)
    - SELL orders ที่ $102k, $104k, $106k, $108k, $110k (ด้านบน)
    - Capital ใน BUY side = $4,700  (ตามที่คำนวณ)
    - Capital ใน upper zone = 0 (ยังไม่มี inventory)

  Front-Loading:
    - เปิด BUY orders เพิ่มจาก $100k ลงมา (aggressive, เต็ม zone)
    - รวม BUY จาก $100k ลงถึง $90k = 5 + 1 orders เพิ่มขึ้น
    - ใช้ capital จาก "upper zone allocation" มา deploy เพิ่ม
    - เมื่อ SELL orders fill ที่ upper zone → คืน capital

  ข้อดี: deploy capital productive มากขึ้น | ข้อเสีย: ถ้าราคาลงก่อน = inventory
เยอะกว่าปกติ
```

ความเสี่ยงของ Front-Loading

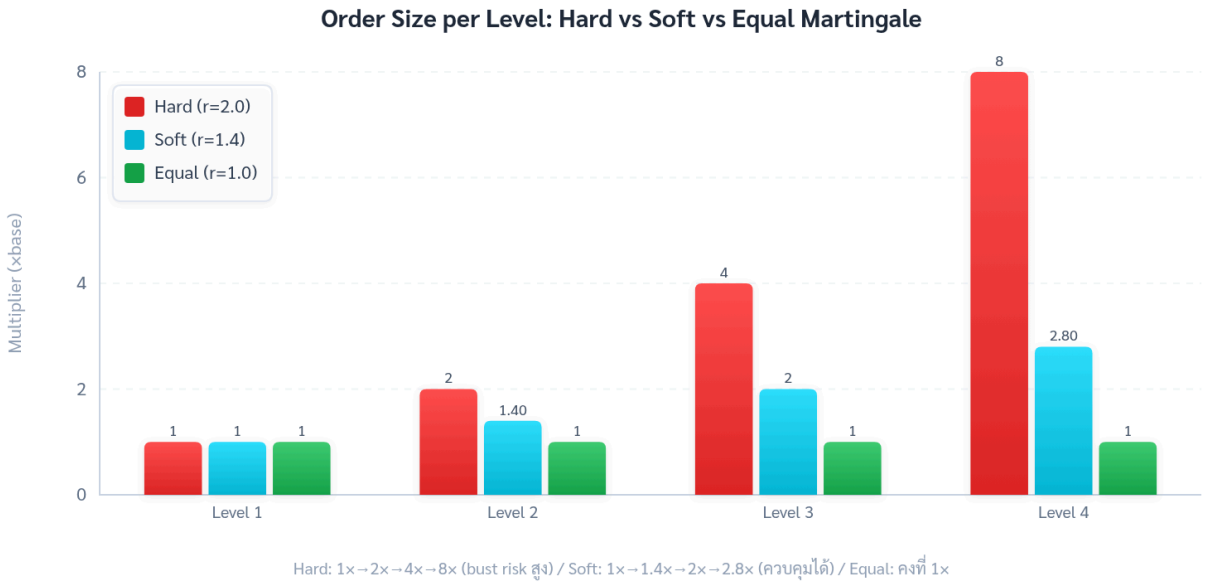
ถ้าราคาลงก่อนที่ upper zone จะ fill → front-loaded capital กลายเป็น "ขาดทุน unrealized" มากกว่าปกติ

วิธีลด risk: Front-load เฉพาะเมื่อ H < 0.45 (mean-reverting ชัดเจน) และ ATR ต่ำกว่าค่าเฉลี่ย (ตลาดเงียบ)

1A.7 Soft Martingale — r-parameter

Martingale ดั้งเดิม: เพิ่ม size เป็น 2×, 4×, 8× ทุกครั้งที่ราคาลง → capital เพิ่มแบบ exponential จนอาจ bust

Soft Martingale: ใช้ r ∈ [1.0, 2.0] (ปรับได้) แทน r = 2 ตายตัว



Soft Martingale (r=1.4) เพิ่ม size ช้ากว่า Hard Martingale (r=2) — capital requirement ต่างกันมาก

```
def soft_martingale_sizes(n_levels, base_q, r):
    """คำนวณขนาด order ในแต่ละ level ด้วย soft martingale"""
    sizes = [base_q * (r ** i) for i in range(n_levels)]
    return sizes

# Hard Martingale r=2.0 (n=5 levels):
# [1×, 2×, 4×, 8×, 16×] × base_q
# Total capital ∝ 1+2+4+8+16 = 31× base_q (สูงมาก)

# Soft Martingale r=1.4 (n=5 levels):
# [1×, 1.4×, 2.0×, 2.7×, 3.8×] × base_q
# Total capital ∝ 1+1.4+2.0+2.7+3.8 = 10.9× base_q (ต่ำกว่ามาก)

# Equal Weight r=1.0 (baseline):
# [1×, 1×, 1×, 1×, 1×] × base_q
# Total capital ∝ 5× base_q (ต่ำสุด, safe)
```

```
def calculate_capital_requirement(prices, base_q, r):
    """คำนวณ capital ทั้งหมดสำหรับ soft martingale grid"""
    total = 0
    for i, price in enumerate(prices):
        q_i = base_q * (r ** i)
        total += q_i * price
    return total
```

r parameter	ชื่อ	Capital ที่ Level 5 (relative)	เหมาะกับ
1.0	Equal Weight	1×	Conservative, baseline
1.2	Gentle Soft	2.1×	มั่นใจ mean-reversion เล็กน้อย
1.4	Moderate Soft	3.8×	สมดุล (แนะนำ)
1.6	Aggressive Soft	6.6×	มั่นใจ mean-reversion มาก
2.0	Hard Martingale	16×	ไม่แนะนำ — bust risk สูง

Stress Test ก่อนเลือก r — BTC ลง 40% ต้องใช้ทุนเท่าไหร่?

ตัวอย่าง: Grid 5 levels · Q=0.001 BTC · r=1.4 · Price $100k

Level ที่ Fill	Q (Soft Martingale r=1.4)	Capital Deployed	Cumulative BTC
1 ($98k)	0.001	$98	0.001
2 ($96k)	0.0014	$134	0.0024
3 ($94k)	0.00196	$184	0.00436
4 ($92k)	0.00274	$252	0.00710
5 ($90k)	0.00384	$346	0.01094
รวม	—	**$1,014**	**0.01094 BTC**

ถ้า BTC ลงต่อถึง $60k (–40% จาก $100k): Inventory loss = 0.01094 × ($100k – $60k) = **–$437.6**

กฎ: ก่อนเลือก r ให้ทำ stress test: Close System capital ต้อง ≥ (cumulative Q ทุก level × worst-case price drop)

r > 1.5 เสี่ยงมากใน bear market — Close System อาจหมดก่อนถึง floor

ข้อจำกัดสำคัญของ Soft Martingale

Soft Martingale ไม่ได้ลบ risk ออก — มันแค่ *ลดความเร็ว* ที่ capital เพิ่มขึ้น

ต้องใช้ร่วมกับ Close System เสมอ: คำนวณ capital requirement ทั้งหมดถึง floor ก่อน แล้วค่อยตัดสินใจว่า r เท่าไหร่ที่ capital พอ

Rule: ถ้า capital ที่ต้องการถึง floor > 50% ของ net worth → ลด r ลงหรือลด N

1A.8 Step Pyramid — เพิ่ม Step แทน Size ในช่วง Trend

IDEA ใหม่ — STEP PYRAMID (ต่างจาก SIZE PYRAMID)

ในช่วง uptrend ชั่วคราว (H ขึ้นแต่ยังไม่ถึง 0.6): แทนที่จะเพิ่ม size ของ order (martingale) → **เพิ่ม step size** แทน

- **Size Pyramid:** step คงที่, size เพิ่ม → inventory risk เพิ่ม
- **Step Pyramid:** size คงที่, step เพิ่ม → inventory risk เท่าเดิม แต่ fill น้อยลง

ผล: cashflow คล้ายกัน (step ใหญ่ × fill น้อย ≈ step เล็ก × fill เยอะ) แต่ inventory accumulation ช้าลงในช่วง trend

```
def step_pyramid(base_step, current_h, h_normal=0.45, h_max=0.60):
    """
    ปรับ step size ตาม Hurst Exponent:
    - H ต่ำ (mean-reverting): step = base_step (หรือเล็กกว่า)
    - H สูง (trending): step เพิ่มขึ้น เพื่อหยุดการ accumulate inventory เร็วเกินไป
    """
    if current_h <= h_normal:
        return base_step
    elif current_h >= h_max:
        return base_step * 2.0  # ช่วง trending — double step (fill น้อยลง 50%)
    else:
        # interpolate
        ratio = (current_h - h_normal) / (h_max - h_normal)
        return base_step * (1.0 + ratio * 1.0)

# ตัวอย่าง: base_step = $2,000
# H=0.45 → step = $2,000  (ปกติ)
# H=0.52 → step = $2,467  (เริ่ม trend)
# H=0.60 → step = $4,000  (trending ชัด → ไม่ค่อย fill)
# H>0.60 → พิจารณา pause grid (ดู Part 4)
```

1A.9 Running Example รวม — Close System + Soft Martingale

🔁 **Running Example: BTC/USDT Close System สมบูรณ์**

Parameters: Entry = $100k · Floor = $90k · Step = $2k · r = 1.3 · Base Q = 0.01 BTC

Level	ราคา	Q (BTC)	Capital ($)	Cumulative ($)	กำไรถ้า Sell
L1	$98,000	0.010	$980	$980	+$20
L2	$96,000	0.013	$1,248	$2,228	+$26
L3	$94,000	0.017	$1,598	$3,826	+$34
L4	$92,000	0.022	$2,024	$5,850	+$44
L5 (floor)	$90,000	0.029	$2,610	**$8,460**	+$58

Capital ที่ต้องมี (Close System): $8,460

Zone Waterfall: Active $4,230 (ใช้งาน) · Standby $2,538 (lending ~5% APY → $0.35/วัน) · Floor $1,692 (สำรอง)

Max Drawdown (worst case): ถ้า BTC ลงถึง $90k ตรงๆ = unrealized ~$1,500 (เพราะถือ BTC ที่ซื้อมาทั้งหมด)

Daily cashflow (ประมาณ, 1.5 รอบ/วัน, เฉลี่ย): ≈ 1.5 × ($30 avg profit) = $45/วัน = 0.53%/วัน บน $8,460

1A.10 แบบฝึกหัด

แบบฝึกหัด Part 1A

▶ ข้อ 1: Close System BTC entry $100k, floor $85k, step $2k, Q = 0.01 BTC — ต้องการ capital เท่าไหร่?

▶ ข้อ 2: ทำไม Buy-Only zone ถึงแนะนำ −2σ ถึง +1σ แทน −2σ ถึง +2σ?

▶ ข้อ 3: Soft Martingale r=1.4, base_q=0.01 BTC, 5 levels — capital รวมมากกว่า Equal Weight กี่%?

▶ ข้อ 4: ทำไม Step Pyramid ดีกว่า Size Pyramid ในช่วง uptrend?

▶ ข้อ 5 (Challenge): ออกแบบ Close System + Soft Martingale สำหรับ $20,000 capital — BTC $100k, floor $80k, step $2k