

* 칼만 필터

(데이터가 불확실하고, 노이즈가 포함되어 있을 때
 '진짜 상태'를 최대한 정확하게 찾아주는 계산 방법)

① 기본 원리 → 센서와 예측 중 더 정확한 쪽에 가중치 ↑

(예측 (Prediction) : 이전까지의 정보로 현재 상태 추정
 업데이트 (Update, 보정) : 데이터 (+노이즈)로 예측값 수정)

② 예측 (Prediction) → 과거의 추정값과 시스템 규칙을 바탕으로

(이전 상태 추정값 : 이전 시점에서 알고 있던 상태 (x, u 등)와 시스템 모델 규칙을 이용하여 새로운 상태 추정값 계산
 오차 공분산 : 잡음, 예측 오차까지 고려하여 예측값이 얼마나 믿을 만 한지, 오차가 얼마나 큰지 상태 추정값의 오차 범위 계산)

③ 업데이트 (Update, 보정)

(혁신 (Innovation), 잔차 (Residual) : 예측값과 관측값의 차
 → 보정할 필요성 판단, 낮을 수록 좋은 예측
 칼만 이득 (Kalman Gain) : 예측값과 측정값의 신뢰도 (오차 공분산)를 바탕으로 어느 쪽을 얼마나 신뢰할지 결정하는 가중치
 → 측정값의 신뢰도 ↑ & 예측값의 신뢰도 ↓ → 칼만 이득 ↑
 상태 추정값 업데이트 : 기존 예측값에 칼만 이득과 혁신값 곱
 → $\hat{x}_t = \hat{x}_{t-1} + K_t(x_t - H\hat{x}_{t-1})$, 예측값 + gain × 혁신
 오차 공분산 업데이트 : 상태 추정값 신뢰도 강화
 → $P_t = (I - K_t H)P_{t-1}$, 추정 상태의 불확실성

* 예측 및 업데이트 과정 → 순차적 (스텝별, 이산적)으로 동작

① 예측 단계 상태 추정값 활용 : $\hat{x}_t = A\hat{x}_{t-1} + Bu_t + \tilde{w}_t$
 (\hat{x}_{t-1} : 이전 시점 업데이트 완료된 상태 추정값
 A : 상태 전이 행렬 (시스템 모델)
 B : 제어 입력 행렬
 u_t : 현재 시점 제어 입력 벡터
 noise)

② 예측 단계 오차 공분산 : $P_t = A \cdot P_{t-1} \cdot A^T + Q$

(P_{t-1} : 이전 시점의 오차 공분산
 A : 상태 전이 행렬 (시스템 모델)
 Q : 시스템 자체에서 발생하는 내부 노이즈 공분산
 → P_{t-1} 을 A를 이용해 현재 시점으로 '이동'
 → 시스템 노이즈로 오차가 더 커진 부분을 Q로 더해 계산
 → 즉 '추가적인 불확실성'이 Q로 반영

③ 측정값 오차 공분산 : $R = E[uv^T]$

→ 센서로부터 유도 / 통계 분석으로 실증적 측정

④ 혁신(잔차) : $y_t = z_t - H\hat{x}_t$

(z_t : 실제 측정값 (+노이즈)
 H : 관측 행렬, 상태를 측정값으로 변환

⑤ 혁신 공분산 계산 : $S_t = H \cdot P_t \cdot H^T + R$

→ P_t 를 H를 이용해 예측값의 관측 공간에서 오차 범위 변환
 → 측정 오차 공분산 R을 더해 두 오차의 합이 될.

⑥ 칼만 이득 계산 : $K_t = P_t \cdot H^T \cdot S_t^{-1}$ 역행렬

→ 혁신 공분산의 역행렬로, 예측 오차 대비 측정오차 비교

⑦ 상태 추정 업데이트 : $\hat{x}_t = \hat{x}_{t-1} + K_t y_t$

⑧ 오차 공분산 업데이트 : $P_t = (I - K_t H)P_{t-1}$

→ $I - K_t H$ 는 혁신값을 반영해 불확실성이 줄어드는 정도 계산
 즉, 측정값 신뢰도가 높을 수록 ↓

비선형 시스템 상태 추정

* 서론

• 상태 변화나 관측 모델

(현실에서는 대부분의 시스템이 비선형 (로봇 팔, 드론, GPS 등)
앞장의 선형 칼만 필터로는 오차 누적 부정확/불안정)

* EKF (Extended Kalman Filter, 확장)

(비선형 함수를 매 시점마다 1차 미분 (야코비안 행렬)로
선형 근사하여 칼만 필터 수식에 적용)

* 야코비안 행렬 (Jacobian Matrix)

(다변수 함수의 각 변수가 미세하게 변경될 때,
다변수 함수가 어떻게 변화하는지! (편미분))

→ 즉, 작은 입력 변화로 인한 출력 변화의 민감도
EKF에선, 비선형 상태/관측 모델을 선형 함수로 근사

ex (딥러닝 역전파(Backpropagation)에서 기울기 구할 때 사용
벡터함수의 속을 단위/기저 벡터로!)

* 테일러 전개 → 다양한 함수를 $x=a$ 기준 다항함수로!

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!} (x-a)^n$$

→ $f(x)$ 는 $x=a$ 에서 무한정 미분 가능해야 한다.

(특정 점($x=a$)근처 구간을 다항함수의 형태로 근사 가능,
1차 근사 시 작은 구간에서 직선처럼, 선형 근사!

→ 1차 근사 : $f(x) \approx f(a) + \underbrace{J_f(a)(x-a)}$
↳ 애가 야코비안 행렬

* 선형화 과정의 부정확, 성능 저하 가능성 → UKF

* UKF (Unscented Kalman Filter, 변형)

(야코비안 행렬/테일러 전개 대신, 상태분포(ECL, 공분산)을
대표하는 샘플점(시그마 포인트)들을 뽑아 비선형 함수에 통과,
결과를 통계적으로 재조합해 상태분포를 더 정확하게 선형화)

↳ 가중 합

↳ 새로운 평균과 공분산 추정

→ n 차원 상태 벡터에서 $2n+1$ 개의 시그마 포인트 선택

각 포인트는 가중치가 부여됨. 대푯값 계산 시 반영