불확실한 선형 시스템에 대한 Data-Driven LQR

심형보 (서울대학교 전기정보공학부)

> "선형 시스템 모델을 잘 모르고도 LQR 제어기를 만들 수 있다고요? LQR 제어의 궤환 이득 행렬을 구하려면 Riccati 방정식을 풀어야 하고 Riccati 방정식은 시스템 모델 정보를 필요로 하는데, 아니 어떻게..."

최근 주목받고 있는 강화학습을 이용한 궤환 제어법을 필두로 모델링이 필요없는 (model-free) 최적 제어에 대한 관심이 높아지고 있다. 어떻게 모델에 대한 정보 없이도 최적 제어가 가능할 지 가만히 생각해 보면, 시스템에 인가된 입력과 그 입력에 의한 모든 상태변수를 일정 시간 동안 측정이 가능하다고 할때 시스템의 정보는 이미 그 안에 들어있다고 볼 수 있다. 이렇게 측정된 정보(데이터)를 어떻게 활용하여 LQR 제어기를 설계할 수 있다는 것인지 알아보기로 하자. 본 편은 2017년에 발간된 Z. P. Jiang 교수의 책 Robust Adaptive Dynamic Programming 중에서 2장의 내용을 저자의 동의 하에 활용한 것이다.

1. 서론 : 다 아는 이야기

선형 시스템에 대한 Linear Quadratic Regulator (LQR)는 널리 알려진 최적 제어법 중 하나이다. 다시 한번 살펴보기 위해 controllable한 선형 시불변 (linear time-invariant) 시스템

$$\dot{x} = Ax + Bu, \qquad x \in \mathbb{R}^n, \quad u \in \mathbb{R}^m \tag{1}$$

을 생각하자. 이 시스템에 대하여 무한 구간[1] 비용 함수

$$J(x_0, u(\cdot)) = \int_0^\infty (x^T(t)Qx(t) + u^T(t)Ru(t))dt \tag{2}$$

가 최소가 되도록 하는 입력 함수 $u : \mathbb{R} \to \mathbb{R}^m$를 찾고자 한다. 윗 식 (2)에서 행렬 R은 positive definite, 행렬 Q는 positive semi-definite 행렬이며 동시에 $(A, \sqrt{Q})$가 observable한 것을 가정한다.[2] 이렇게 되면 비용 함수 J는 초기값 $x(0) = x_0$와 입력 함수 $u(\cdot)$의 함수가 되는 것을 확인하자. 왜냐하면, 식 (2)의 적분 중에 나오는 $x(t)$가 주어진 초기값 $x(0) = x_0$와 주어진 입력 함수 $u(\cdot)$로부터 식 (1)을 통해 얻어지는 시스템의 해이기 때문이다. 따라서, 주어진 초기값 $x(0) = x_0$에 대해 최적화 문제

$$\text{minimize}_{u(\cdot)} J(x_0, u(\cdot))$$

를 풀어 최적 입력인 $u^*(\cdot)$를 구하는 것이 LQR 문제가 된다. 개념적으로는 가능한 모든 시간 함수 $u(\cdot)$로 식 (2)의 J 값을 구해보아 가장 작은 J를 주는 입력을 고르는 문제이다. 따라서 그 결과물은 시간의 함수 $u^*(t)$로 얻어지게 되어 open-loop control이 되는 것이 일반적이나, 시스템이 선형 시불변이고 무한 구간 최적화 문제이기 때문에 최적 입력 u^*를

$$u^*(t) = -K^* x(t) \tag{3}$$

와 같이 표현할 수도 있음이 알려져 있다 [3]. 이는 시간 함수 $u^*(t)$가 상태변수 $x(t)$를 측정한 뒤 상수 행렬 K^*를 곱하는 것으로도 얻어질 수 있음을 의미하기 때문에, 자연스럽게 상태 궤환 제어가 가능하게 되어 모델 불확실성이나 외란이 작용하는 실제 시스템에 적용하기 적합한 형태가 되는 것이다. (더구나 이렇게 구한 모든 LQR 제어기 (3)은 gain margin이 0.5에서 무한대, phase margin이 $\pm 60°$까지 자동으로 보장됨이 알려져 있다 [3].)

위 식 (3)에 나오는 이득 행렬 K^*는

$$K^* = R^{-1}B^T P^* \tag{4}$$

로 주어지는데, 여기서 행렬 P^*는 Riccati 행렬식

$$A^T P^* + P^* A + Q - P^* B R^{-1} B^T P^* = 0 \tag{5}$$

[1] 식 (2)의 적분 구간이 무한 시간까지 주어져 있어서 무한 구간 (infinite horizon) LQR이라 불린다

[2] $\sqrt{Q}$는 제곱하면 Q가 되는 행렬을 표시하는 기호이다. 한편, positive (semi-)definite 행렬이라 말할 때 그 행렬이 대칭행렬이라는 의미를 포함하는 것으로 하자. 앞으로 $P > 0$는 행렬 P가 positive definite한 것으로, $P \geq 0$은 P가 positive semi-definite한 것으로 약속하자. $P > Q$는 $P - Q > 0$을 의미하고, $P \geq Q$는 $P - Q \geq 0$을 의미한다.

을 만족하는 positive definite한 행렬이며, 현재까지 이야기된 조건 하에서 이러한 행렬 P^*가 존재하고 유일함이 증명되어 있다. 또 하나 주목할 것은 최적 입력 u^*를 인가할 경우 시스템은

$$\dot{x} = (A - BK^*)x \tag{6}$$

가 되며 행렬 $A - BK^*$의 고유값이 모두 음의 실수부를 갖게 되어, 최적 입력 u^*는 시스템을 안정화하는 제어기임이 증명되어 있다. 한편, (5)에 의해

$$\begin{aligned}&(A-BK^*)^T P^* + P^*(A-BK^*) + Q + K^{*T}RK^* \\ &= A^T P^* + P^* A - 2P^* BR^{-1}B^T P^* + Q + P^* BR^{-1}B^T P^* \\ &= 0\end{aligned}$$

임을 주목하자. 이 식은 행렬 P^*에 대해 선형식인 Lyapunov 행렬식이고 행렬 $(A-BK^*)$가 Hurwitz이므로

$$P^* = \int_0^\infty e^{(A-BK^*)^T\tau}(Q + K^{*T}RK^*)e^{(A-BK^*)\tau}d\tau$$

로 주어진다.[3] 또한, 위 시스템 (6)이 초기값 x_0를 가질 때의 해를 x^*로 표기한다면,

$$\begin{aligned}J(x_0, u^*(\cdot)) &= \int_0^\infty (x^{*T}(t)Qx^*(t) + u^{*T}(t)Ru^*(t))dt \\ &= \int_0^\infty x^{*T}(t)(Q + K^{*T}RK^*)x^*(t)dt\end{aligned}$$

이고, $x^*(t) = e^{(A-BK^*)t}x_0$이므로 최소 비용 $J^*(x_0)$는

$$J^*(x_0) = J(x_0, u^*(\cdot)) = x_0^T P^* x_0$$

로 주어짐을 알 수 있다.

정리하면, LQR 제어를 하기 위해서는 우선 Riccati 행렬식 (5)를 풀어 P^*를 구한 다음, (4)를 통해 궤환 이득 행렬 K^*를 구하면 된다. 그런데 이 두 과정 모두 시스템 모델 정보, 즉, 행렬 A와 B를 필요로 한다. 그럼에도 불구하고, 이러한 모델 정보 없이 행렬 P^*와 K^*를 구할 수 있다는 것이 본 편의 핵심 주제이다.

2. 본론 : 흥미로운 이야기

2.1. Riccati 행렬식 대신 Lyapunov 행렬식을 반복적으로 풀어 P^*와 K^* 구하기

모델 정보 없이 P^*와 K^*를 구하기에 앞서, Riccati 행렬식은 미지 행렬 P^*에 대해 2차식이라는 점을 주목해 보자. 즉, 미지 행렬 P^*에 대한 선형 방정식이 아니기 때문에 Riccati 행렬식의 해 P^*를 구하는 것은 단순하지 않아, 이를 풀기 위한 여러가지 알려진 방법들을 사용해야 한다. 본 편에서는 그 중 Kleinman의 알고리듬 [1]을 사용하여 반복적인(iterative) 방법으로 구하는 방법을 소개한다.

0. $A - BK_0$가 Hurwitz 행렬이 되도록 만드는 행렬 $K_0 \in \mathbb{R}^{m\times n}$을 임의로 선정한다. $k = 0$이라 한다.
1. $A_k = A - BK_k$라 정하고, 다음 Lyapunov 행렬식

$$A_k^T P_k + P_k A_k + Q + K_k^T R K_k = 0 \tag{7}$$

을 풀어 positive definite 행렬 P_k를 얻는다.
2. $K_{k+1} = R^{-1}B^T P_k$라고 정하고, $k \leftarrow k+1$을 수행한 뒤 1번으로 간다.

위의 반복적인 방법에서 Lyapunov 행렬식 (7)은 Riccati 행렬식 (5)와 달리 미지 행렬 P_k에 대해 선형이기 때문에 쉽게 해를 구할 수 있다는 점을 주목하자. 위 방법을 통해 얻게 되는 성질은 다음과 같다.

Theorem 1: 모든 $k \geq 0$에 대하여

(a) $A - BK_k$는 Hurwitz 행렬이다.
(b) $P^* \leq P_{k+1} \leq P_k$
(c) $\lim_{k\to\infty} P_k = P^*$, $\lim_{k\to\infty} K_k = K^*$

증명: (a) $k \geq 1$의 경우에 대하여, 폐루프 시스템 $\dot{x} = A_k x = (A - BK_k)x$의 안정도를 보이기 위해, 식 (7)을 만족하는 positive definite P_k를 사용한 Lyapunov 함수 $V(x) = x^T P_k x$를 이용하여

$$\begin{aligned}\dot{V} &= x^T(PA_k + A_k^T P)x = -x^T(Q + K_k^T R K_k)x \\ &= -\|\sqrt{Q}x\|^2 - \|\sqrt{R}K_k x\|^2\end{aligned}$$

를 얻은 뒤 LaSalle 정리를 이용한다. 즉, 위 식으로부터 $x(t)$는 집합 $\{x : \sqrt{Q}x = 0, \sqrt{R}K_k x = 0\}$으로 수렴함을 알 수 있고, 이 집합에서는 $\dot{x} = A_k x = Ax$가 되기 때문에, $(A, \sqrt{Q})$의 observability로부터 largest invariant 집합이 원점 뿐임을 알 수 있다.

[3] Lyapunov 행렬식은 $A^T P + PA + Q = 0$의 형태로 주어지는데, 행렬 A가 Hurwitz라면, $P = \int_0^\infty e^{A^T\tau}Qe^{A\tau}d\tau$가 이 식을 만족한다. 왜냐하면,

$$A^T P + PA = A^T \int_0^\infty e^{A^T\tau}Qe^{A\tau}d\tau + \int_0^\infty e^{A^T\tau}Qe^{A\tau}d\tau A = \int_0^\infty \frac{d}{d\tau}\left(e^{A^T\tau}Qe^{A\tau}\right)d\tau = e^{A^T\tau}Qe^{A\tau}|_{\tau=0}^\infty = -Q$$

이기 때문이다.

(b) $k=1$일 때 식 (7)은

$$0 = A_1^T P_1 + P_1 A_1 + Q + K_1^T R K_1 \tag{8}$$

인 반면, $A_0 = A - BK_0 = A - BK_1 - B(K_0 - K_1) = A_1 - B(K_0 - K_1)$라는 점에 주목하면, $k=0$일때 식 (7)은

$$\begin{aligned} 0 &= A_0^T P_0 + P_0 A_0 + Q + K_0^T R K_0 \\ &= A_1^T P_0 + P_0 A_1 + Q + K_0^T R K_0 \\ &\quad - (K_0 - K_1)^T B^T P_0 - P_0 B (K_0 - K_1) \end{aligned} \tag{9}$$

와 같이 표현할 수 있다. 즉, 의도적으로 식 (9)에서 A_0 대신 A_1을 사용하였다. 이제 식 (9)에서 (8)을 빼면

$$\begin{aligned} 0 = A_1^T (P_0 - P_1) + (P_0 - P_1) A_1 + K_0^T R K_0 - K_1^T R K_1 \\ - (K_0 - K_1)^T B^T P_0 - P_0 B (K_0 - K_1) \end{aligned} \tag{10}$$

가 된다. 한편, 항등식

$$\begin{aligned} (K_0 - K_1)^T R K_1 + K_1^T R (K_0 - K_1) \\ = K_0^T R K_1 + K_1^T R K_0 - 2 K_1^T R K_1 \end{aligned}$$

을 식 (10)에 좌변을 더하고 우변을 빼면, 식 (10)은

$$\begin{aligned} 0 &= A_1^T (P_0 - P_1) + (P_0 - P_1) A_1 + (K_0 - K_1)^T R (K_0 - K_1) \\ &\quad + (K_0 - K_1)^T (R K_1 - B^T P_0) + (K_1^T R - P_0 B)(K_0 - K_1) \end{aligned}$$

이 된다. 그런데, $K_1 = R^{-1} B^T P_0$이므로 윗 식은 이제

$$0 = A_1^T (P_0 - P_1) + (P_0 - P_1) A_1 + (K_0 - K_1)^T R (K_0 - K_1)$$

가 된다. 이 식은 $(P_0 - P_1)$에 대해 Lyapunov 행렬식이고 A_1이 Hurwitz이기 때문에 (각주 3 참조)

$$P_0 - P_1 = \int_0^\infty e^{A_1^T \tau} (K_0 - K_1)^T R (K_0 - K_1) e^{A_1 \tau} d\tau$$

가 되고, 우변은 positive semi-definite한 행렬이기 때문에, $P_0 - P_1 \geq 0$이 성립한다.

같은 기법을 활용하면 $P_k \geq P_{k+1}$을 보일 수 있고, 최적 행렬 $P^* > 0$ 또한 식 (7)과 같은 $(A - BK^*)^T P^* + P^* (A - BK^*) + Q + K^{*T} R K^* = 0$을 만족하기 때문에, 역시 위의 기법을 활용하면 임의의 k에 대하여 $P_k \geq P^*$를 보일 수 있다. 이를 통해 (b)가 증명되고, (b)로부터 (c)가 따라 나오게 된다. //

그럼에도 불구하고 Kleinman의 방법은 여전히 시스템의 정보인 행렬 A와 B를 사용하고 있음에 주목하자.

2.2. A와 B의 정보 없이 데이터를 이용하여 P^*와 K^* 구하기

이제 시스템 모델 A와 B를 사용하지 않는 대신 실제 시스템을 일정 시간 임시 제어 입력으로 제어를 해 본 뒤, 이 때 얻은 상태변수의 데이터로부터 최적 제어에 필요한 P^*와 K^*를 얻는 두 가지 방법을 알아보려 한다. 두 방법 모두 앞 소절에서 살펴 본 Kleinman의 알고리듬이 중요한 역할을 하게 되는데, 우선 궤환 이득 K_0이 시스템을 안정화한다고 가정한다. On-policy learning이라 불리는 첫번째 방법은, K_0를 사용하여 시스템을 일정 시간 제어해 본 뒤 얻어진 데이터를 가지고 P_0과 K_1을 얻고, 다시 궤환 이득 K_1을 사용하여 시스템을 제어해 본 뒤 얻는 데이터를 이용하여 P_1과 K_2를 얻고, 이를 반복하여 P^*와 K^*를 근사적으로 얻는 방법이다. Off-policy learning이라 불리는 두번째 방법은, K_0를 사용하여 시스템을 일정 시간 제어해 본 뒤 얻어진 데이터를 반복적으로 재사용하여, 첫번째 방법처럼 여러번 다시 실험하지 않고도 P_k과 K_{k+1}을 반복적으로 얻어 P^*와 K^*를 근사적으로 얻는 방법이다.

이러한 방법을 시스템의 실제 운용 전 시험 운용에 적용해 최적 이득 K^*를 얻은 후 실제 운용에 사용할 수도 있고, 혹은 K^*가 근사적으로 얻어질 때까지의 좋지 않은 성능을 감내할 수 있다면 실제 운용에 바로 적용하여 시스템의 성능이 스스로 점점 좋아지는 것을 관찰할 수도 있을 것이다. 다만 최초의 K_0가 시스템을 안정화하는 궤환 이득이어야 한다는 점이 시스템 모델을 잘 모르는 상황에서 현실성이 떨어지는 가정일 수 있다. 하지만, 불확실한 시스템이 적어도 그 자체로 안정한 경우라면 K_0를 영행렬로 설정하면 된다. 물론 제어 성능이 떨어지더라도 시스템을 안정하게 하는 궤환 이득을 실험적으로 찾을 수 있는 경우도 있겠다.

* On-policy Learning

시스템을 일정 시간 제어해 본 뒤 얻어지는 데이터로부터 최적 궤환 이득을 얻기 위해서는 시스템에 관한 정보가 충실히 데이터에 실려 있어야 할 것이다. 이를 위해 최적 이득 K^*가 얻어질 때까지 시스템에 인가할 입력으로

$$u(t) = -K_k x(t) + e(t) \tag{11}$$

와 같이 $-K_k x$ 외에 추가적인 신호 e를 주입하도록 한다. 여기서 e는 exploration noise라고 불리는 의도적으로 주입하는 신호이며, 뒤에서 다시 논의하도록 한다. 위 제어 입력을 인가하게 되면 시스템은

$$\dot{x} = A_k x + Be$$

가 되며, $Q_k = Q + K_k^T R K_k$라고 했을 때,

$$\begin{aligned}\frac{d}{dt}(x^T P_k x) &= x^T(A_k^T P_k + P_k A_k)x + 2e^T B^T P_k x \\ &= -x^T Q_k x + 2e^T R K_{k+1} x\end{aligned} \tag{12}$$

가 됨을 알 수 있다. 이 식의 우변에서 A와 B가 사라졌다는 점에 주목하자. 이제 윗 식의 $2e^T R K_{k+1}x$ 항을 좌변으로 이항한 뒤, 양변을 시간 δ만큼 적분하면

$$\begin{aligned}&x^T(t+\delta)P_k x(t+\delta) - x^T(t)P_k x(t) \\ &-2\int_t^{t+\delta} e^T(\tau)RK_{k+1}x(\tau)d\tau = -\int_t^{t+\delta} x^T(\tau)Q_k x(\tau)d\tau\end{aligned} \tag{13}$$

을 얻는다. 만약 우리가 K_k만 알고 있고 시스템 행렬 A와 B를 모른다 하더라도, 윗 식에서 R과 Q_k, 그리고 입력 (11)을 인가하여 얻어진 시스템의 상태변수 $x(t)$를 알고 있기 때문에, 윗 식으로부터 P_k와 K_{k+1}을 구할 수 있을 것이란 생각이 든다.

사실 윗 식은 미지 행렬 P_k와 K_{k+1}에 대한 선형 방정식인데, 이를 더욱 명확히 보기 위해서 임의의 두 행렬을 이용해서 커다란 행렬을 만드는 Kronecker product라는 연산자와, 행렬을 열벡터(column vector)로 만들어주는 vec이라는 연산자에 대해 잠시 알아보자. 두 행렬 $M \in \mathbb{R}^{p\times q}$과 $N \in \mathbb{R}^{r\times s}$이 주어졌을 때 M과 N의 Kronecker product는, m_{ij}를 행렬 M의 (i,j)번째 원소라 할 때

$$M \otimes N = \begin{bmatrix} m_{11}N & \cdots & m_{1q}N \\ \vdots & \ddots & \vdots \\ m_{p1}N & \cdots & m_{pq}N \end{bmatrix} \in \mathbb{R}^{pr\times qs}$$

와 같이 정의된다. Kronecker product에 관한 한 가지 유용한 성질은 $(M\otimes N)(L\otimes J) = (ML\otimes NJ)$와 같은 행렬곱이 성립한다는 것이며 증명은 생략한다. 한편, 행렬 $M \in \mathbb{R}^{p\times q}$을 열벡터로 변환하는 연산자를 $\mathrm{vec}(M)$라고 하는데, $\bar{m}_i$를 행렬 M의 i번째 열이라 할 때

$$\mathrm{vec}(M) = \begin{bmatrix} \bar{m}_1 \\ \vdots \\ \bar{m}_q \end{bmatrix} \in \mathbb{R}^{pq\times 1}$$

와 같이 정의된다. 또한, 세 행렬 M, N, L의 곱에 관하여

$$\mathrm{vec}(MNL) = (L^T \otimes M)\mathrm{vec}(N)$$

이 성립함이 알려져 있다. 이는 행렬곱 MNL의 가운데 행렬을 열벡터로 만들고 싶을 때 사용할 수 있는 유용한 성질이다. (행렬곱 MN에서 N을 열벡터로 만들고 싶을 때는 단위 행렬 I를 사용하여 $MN = MNI$처럼 생각하면 된다.)

이제 식 (13)에서 미지 행렬인 P_k와 K_{k+1}을 열벡터로 만들기 위해 (13)의 양변에 연산자 vec을 인가하면

$$\begin{aligned}&\left(x^T(t+\delta)\otimes x^T(t+\delta) - x^T(t)\otimes x^T(t)\right)\mathrm{vec}(P_k) \\ &-2\int_t^{t+\delta}\left(x^T(\tau)\otimes e^T(\tau)R\right)d\tau\,\mathrm{vec}(K_{k+1}) \\ &= -\int_t^{t+\delta} x^T(\tau)Q_k x(\tau)d\tau\end{aligned} \tag{14}$$

가 된다. (우변의 경우 Q_k를 굳이 열벡터로 만들 필요가 없고, 마침 상수항이어서 vec 연산을 한 결과를 그 자신으로 두었다.) 이제 어떤 시간 수열 $\{t_i^k : i = 1,2,\cdots,l_k, k = 0,1,\cdots\}$이 있어 $t_i^k + \delta < t_{i+1}^k$과 $t_{l_k}^k + \delta < t_1^{k+1}$을 만족한다고 하자. 주어진 k에 대하여 각 $t = t_i^k$에 대한 식 (14)를 나열하여 한번에 쓰면

$$\begin{bmatrix} x^T\otimes x^T\big|_{t_1^k}^{t_1^k+\delta} & -2\int_{t_1^k}^{t_1^k+\delta}(x^T\otimes e^T R)d\tau \\ x^T\otimes x^T\big|_{t_2^k}^{t_2^k+\delta} & -2\int_{t_2^k}^{t_2^k+\delta}(x^T\otimes e^T R)d\tau \\ \vdots & \vdots \\ x^T\otimes x^T\big|_{t_{l_k}^k}^{t_{l_k}^k+\delta} & -2\int_{t_{l_k}^k}^{t_{l_k}^k+\delta}(x^T\otimes e^T R)d\tau \end{bmatrix} \begin{bmatrix} \mathrm{vec}(P_k) \\ \mathrm{vec}(K_{k+1}) \end{bmatrix}$$
$$= \begin{bmatrix} -\int_{t_1^k}^{t_1^k+\delta} x^T Q_k x d\tau \\ -\int_{t_2^k}^{t_2^k+\delta} x^T Q_k x d\tau \\ \vdots \\ -\int_{t_{l_k}^k}^{t_{l_k}^k+\delta} x^T Q_k x d\tau \end{bmatrix}$$

즉,

$$\Theta_k \begin{bmatrix} \mathrm{vec}(P_k) \\ \mathrm{vec}(K_{k+1}) \end{bmatrix} = \Xi_k \tag{15}$$

의 형태가 되며, 이는 주어진 k에 대해 각 시간 t_i^k $(i = 1,\cdots,l_k)$에서 식 (13)을 여러번 나열한 것이라고 볼 수 있다. 결국 식 (13)은 미지 행렬 P_k와 K_{k+1}에 대해 선형 방정식이란 뜻이며, 이 식은 (12)로부터 유래되었기 때문에 해 P_k와 K_{k+1}는 항상 존재하게 된다. 다만, 이 해가 유일할 조건은 Θ_k의 행이 모두 독립일 때라고 말하고 싶지만, 불행히도 Θ_k의 처음 n^2 행은 $x^T\otimes x^T$의 형태를 가지기 때문에 최대 $n(n+1)/2$개의 독립인 행밖에 가질 수가 없다. (왜 그런지 잘 생각해 보라!) 그런데, 마침 행렬 $P_k \in \mathbb{R}^{n\times n}$는 대칭행렬이기 때문에 이 행렬의 n^2 원소 중 자유롭게 정할 수 있는 것은 $n(n+1)/2$개뿐이다. 아닌 게 아니라, $\mathrm{vec}(P_k)$에서 대칭성 때문에 중복된 원소를 지운다고 하면, 식 (15)에서 행렬곱을 수행할 때, $\mathrm{vec}(P_k)$의 지워지는

원소에 대응하는 Θ_k의 해당 행도 함께 지울 수 있고, 마침 그 해당하는 행을 모두 제외하면 Θ_k 행렬이 모두 독립인 행을 가질 수 있게 됨을 알 수 있다. 정리하면, P_k의 대칭성 때문에, vec(P_k)에 존재하는 미지수의 실질적인 갯수는 $n(n+1)/2$이며, 따라서 식 (15)가 유일한 해를 가질 조건은

$$\text{rank}(\Theta_k) = \frac{n(n+1)}{2} + nm \tag{16}$$

라는 말이다.

그럼 언제 식 (16)이 만족될 것인가? 우선 Θ_k의 열 갯수 l_k가 $n(n+1)/2+nm$ 이상으로 충분히 커야하며, 동시에 Θ_k의 원소들 간의 규칙성이 없는 것이 유리하다는 것을 알 수 있다. 사실 이는 적응제어(adaptive control)의 persistent excitation 조건과 개념적으로는 동일한데, 결국 입력 항에 있는 exploration noise $e(t)$에 불규칙성이 많아, 이 입력으로 인해 야기되는 $\dot{x} = (A - BK_k)x + Be$의 해 $x(t)$도 불규칙성을 많이 가질수록, $e(t)$와 $x(t)$의 영향을 받는 Θ_k의 행이 독립이 될 가능성이 높아지는 것이다.

이제부터는 $e(t)$가 충분히 시스템을 흔들어 놓아 위 (16)의 가정이 모든 k에 대해 성립한다고 하자. 이를 바탕으로 최초의 궤환 이득 K_0로부터 출발하여, 시스템 모델 정보를 사용하지 않고 다음 방법으로 최적 K^*를 구할 수 있다.

(0): $k = 0$이라 하자.

(1): 실제 시스템에 $u(t) = -K_k x(t) + e(t)$라는 입력을 시간 $t_1^k \le t \le t_{l_k}^k + \delta$ 동안 인가하고 x와 e를 기록한다.

(2): 이를 통해 Θ_k과 Ξ_k을 구하고 Θ_k이 (16)을 만족하는지 확인한다. (만족하지 않는다면 l_k를 좀더 키우고 더 오랜 시간 실험을 계속하여 Θ_k의 추가적인 행을 만들어 본다.)

(3): 식 (15)를 풀어 P_k와 K_{k+1}을 구한다.

(4): $\|P_k - P_{k-1}\|$이 정해진 값보다 작으면 멈추고 아니면 $k \leftarrow k+1$하고 (1)번부터 반복한다.

이 방법은 시스템 모델 정보 A와 B 없이 Kleinman 알고리듬을 사용하고 있는 것이며, 따라서 l_k가 충분히 크고 exploration noise e가 잘 설계되어 식 (16)이 모든 k에 대해 성립한다고 하면, Theorem 1이 여전히 성립한다.

이 방법에서 식 (15)에서 Θ_k와 Ξ_k 원소에서의 $x(t)$는 궤환 이득 K_k가 사용되었을 때의 시스템 상태변수를 의미한다. 따라서, 이를 얻기 위해서는 매 k번째 시행마다 $u = -K_k x + e$를 실제 실험에 인가하여 $x(t)$를 얻어야만 한다. 이런 방식을 "on-policy learning"이라 한다. 그런데, 하나의 입력만을 실제 시스템에 인가하여 일정한 시간동안 데이터를 얻은 뒤, 그 데이터만을 반복적으로 활용하여 P^*와 K^*를 구하는 방법을 "off-policy learning"이라 하며, 이를 소개하고자 한다.

* Off-policy Learning

핵심 아이디어는 $u = -K_k x + e$ 대신 $u = u_0$라는 입력을 넣는 것인데, 이 경우

$$u = u_0 = -K_k x + (K_k x + u_0) = -K_k x + e$$

라고도 생각할 수 있으니, 이는 마치 매 k번째 실험에서 noise $e(t)$로 $(K_k x(t) + u_0(t))$가 사용된 것과 같이 간주될 수 있다. 결국 똑같은 신호 $x(t)$를 마치 $u = -K_k x + e_k$에 대한 시스템의 해이고 k번째 마다 서로 다른 $e_k = K_k x + u_0$에 의한 것이라고 간주하는 것이다.

이렇게 되면 식 (13)과 (14)에서 e 대신 $K_k x + u_0$로 치환하게 되고, 식 (14)의 좌변의 적분은 항등식

$$\begin{aligned} x^T \otimes e^T R &= x^T \otimes (x^T K_k^T + u_0^T) R \\ &= (x^T \otimes x^T)(I_n \otimes K_k^T R) + (x^T \otimes u_0^T)(I_n \otimes R) \end{aligned}$$

을 사용하면,

$$\begin{aligned} &\left(x^T(t+\delta) \otimes x^T(t+\delta) - x^T(t) \otimes x^T(t)\right) \text{vec}(P_k) \\ &- 2\int_t^{t+\delta} (x^T \otimes x^T) d\tau \, (I_n \otimes K_k^T R) \, \text{vec}(K_{k+1}) \\ &- 2\int_t^{t+\delta} (x^T \otimes u_0^T) d\tau \, (I_n \otimes R) \, \text{vec}(K_{k+1}) \\ &= -\int_t^{t+\delta} x^T \otimes x^T d\tau \, \text{vec}(Q_k) \end{aligned} \tag{17}$$

가 된다. 이제 $0 \le t_1 < t_2 < \cdots < t_l$에 대해

$$\delta_{xx} = \begin{bmatrix} x^T(t_1+\delta) \otimes x^T(t_1+\delta) - x^T(t_1) \otimes x^T(t_1) \\ \vdots \\ x^T(t_l+\delta) \otimes x^T(t_l+\delta) - x^T(t_l) \otimes x^T(t_l) \end{bmatrix}$$

$$\theta_{xx} = \begin{bmatrix} \int_{t_1}^{t_1+\delta} x^T \otimes x^T d\tau \\ \vdots \\ \int_{t_l}^{t_l+\delta} x^T \otimes x^T d\tau \end{bmatrix}, \quad \theta_{xu} = \begin{bmatrix} \int_{t_1}^{t_1+\delta} x^T \otimes u_0^T d\tau \\ \vdots \\ \int_{t_l}^{t_l+\delta} x^T \otimes u_0^T d\tau \end{bmatrix}$$

라고 하고, 이를 통해

$$\begin{aligned} \Theta_k &:= \left[\delta_{xx}, \quad -2\theta_{xx}(I_n \otimes K_k^T R) - 2\theta_{xu}(I_n \otimes R)\right] \\ \Xi_k &:= -\theta_{xx} \text{vec}(Q_k) \end{aligned}$$

라 정의하면, 식 (17)을 $t = t_1, t_2, \cdots, t_l$에 적용하여

$$\Theta_k \begin{bmatrix} \text{vec}(P_k) \\ \text{vec}(K_{k+1}) \end{bmatrix} = \Xi_k \tag{18}$$

를 얻는다.

행렬 Θ_i의 모양이 on-policy learning의 경우와 달라져 있지만,

$$\text{rank}\left(\begin{bmatrix}\theta_{xx}, & \theta_{xu}\end{bmatrix}\right) = \frac{n(n+1)}{2} + mn \qquad (19)$$

이 만족되면 Θ_i도 같은 rank를 가짐을 보일 수 있다. 따라서, 식 (19)가 만족되는 한 on-policy learning에서의 결과와 동일한 결과를 얻을 수 있다.

이제 off-policy learning이 어떻게 진행되는 지 살펴보자. 우선 $u_0(t)$라는 임시 제어 입력을 만들어 t_1부터 $t_l + \delta$까지 시스템에 인가하고 $x(t)$를 기록한다. 물론 입력 u_0의 한 예로 $-K_0x(t) + e(t)$를 써도 되고 다른 것을 써도 된다. 기록된 $x(t)$와 $u_0(t)$로부터 δ_{xx}, θ_{xx}, θ_{xu}를 만들고 가정 (19)가 성립될 때까지 l을 키운다. 식 (17)을 잘 보면, 이때 만든 δ_{xx}, θ_{xx}, θ_{xu}는 매 k번째 시행마다 새로 구할 필요가 없다는 것을 알 수 있다. 따라서, K_0로부터 시작하여, P_0과 K_1, 다시 P_1과 K_2를 계속해서 식 (17)로부터 얻어낼 수 있다.

3. 결론

본 편에서는 시스템 모델 정보 없이도 시스템의 입력과 상태변수 정보로부터 최적 제어 입력을 구해낼 수 있음을 살펴보았다. 본 편의 접근법은 Markov decision model을 사용하는 강화학습과 많은 유사성이 있으며, 연속 시간 공간에서 전개된 고전 제어에 익숙한 독자들이 쉽게 data-driven 제어의 분위기를 파악하는데 도움이 될 것으로 생각한다.

참고

본 편의 전체 내용은 현재 서울대학교 CDSL에 재학 중인 김정우 학생의 발표 자료와 참고문헌 [2]의 2장을 참고하여 작성하였다. 조판을 도와 준 장하민 학생에게 감사하며, 본 편의 LaTeX 조판본은 제어이론연구회 홈페이지에서 구할 수 있다.

REFERENCES

[1] D. L. Kleinman, "On an iterative technique for Riccati equation computations," *IEEE Trans. on Automatic Control*, pp. 114–115, 1968.

[2] Y. Jiang and Z. P. Jiang, *Robust Adaptive Dynamic Programming*, Wiley, 2017.

[3] B. D. O. Anderson and J. B. Moore, *Optimal Control: Linear Quadratic Methods*, Prentice-Hall, 1989.

심형보 교수는 2000년 서울대학교 전기공학부에서 박사학위를 받고 미국 산타바바라 소재 캘리포니아 주립대학에서 박사후 과정을 수료 후 현재 서울대학교에서 교수로 재직 중이다. Automatica, IEEE Transactions on Automatic Control 등 저널의 associate editor를 맡은 바 있다.