자, LQR을 짧게 복습하자면, 선형시스템을 만족하면서 내가 원하는 궤적 (x_t)를 따라가게 하는 최적의 u_t 찾기라고 볼수있다.
이때, 목적함수에 따라 u_t는 (nominal을 0으로 설정할때) Q와 R의 설정에 따라 u_t를 작게 쓰는게 얼마나 중요한지에 대한 가중치가 정해진다.
우선순위는 다음과 같다. ① 선형시스템 만족 ② nominal (guide) 따라가기

2번과 3번의 중요도의 대소를 따질수 있는가?

그번은 x와 u를 각각 guide로 줄수있다는 생각으로부터 나온 틀린 발상이였어서 지우고 3번을 2번으로 옮김

nominal 한 값을 guide로 볼수있는가?
아니면 출발값으로 봐야하나?
둘다로 볼수 있는건 아닌가?

iLQR에선 출발값. LQR에선 가이드.
두 경우 모두에 쓸수 있는 한국어 번역은 "기준값"

LQR의 중요가정은 다음과 같다.

목적함수) (줄일수록 좋은것) $\frac{1}{2}x_N^TQ_Nx_N+\frac{1}{2}\sum_{k=1}^{N-1}\left(x_k^TQ_kx_k+u_k^TR_ku_k\right)$

만약 1차항과 상수항이 붙는다면 어찌되는가?
이름의 quadratic이 목적함수를 가르킨건데, 1차항과 상수항이 붙으면 이는 quadratic이 아닌가?

최적 cost-to-go) $V_k=\min_{u_k}\frac{1}{2}\left(x_k^TQ_kx_k+u_k^TR_ku_k\right)+V_{k+1}(x_{k+1})=\underline{\frac{1}{2}x_k^TS_kx_k}$

1차항 (S_kx_k), 상수항이 붙으면 어찌될까?

선형시스템) $x_{k+1}=\underbrace{A_kx_k+B_ku_k}_{\text{시간에 따라 변하는 선형시스템이어도 가능.}}$

positive-definite) Q_k는 real-positive-semidefinite.
R_k은 real-positive-definite

왜 서로 조건이 다른지?

두조건의 필요성) Q_k가 positive-semidefinite가 아니면 목적함수에서 $x_k^TQ_kx_k$가 음수가 될수도 있다.
R_k가 positive-definite가 아니면 $u_k^TR_ku_k$가 0이 될수 있다. 이는 제어입력의 크기를 고려하지 않는 다는뜻

implicit assumption)
action-value function) $Q_k(x_k,u_k)=\frac{1}{2}x_k^TQ_kx_k+\frac{1}{2}u_k^TR_ku_k+V_{k+1}$
value function) $V_k(x_k)=\min_{u_k}\left(\frac{1}{2}x_k^TQ_kx_k+\frac{1}{2}u_k^TR_ku_k+V_{k+1}\right)$
for final step, there is no u_k.) $Q_N(x_N)=\frac{1}{2}x_N^TQ_Nx_N=V_N(x_N)$
start from $k=1$ to $k=N$
⊕ 선형지식 (행렬·벡터곱 미분): $\frac{\partial}{\partial x}(x^TA)=A^T$, $\frac{\partial}{\partial x}(Ax)=A$
ex) $\frac{\partial}{\partial x}(x^TAx)=Ax+A^Tx$

알고리즘 전개 : $(x_k,u_k)_{nominal}=(0,0)\mid\forall k\in t$)

$$V_N(x_N)=\frac{1}{2}x_N^TQ_Nx_N=\frac{1}{2}x_N^TS_Nx_N\Rightarrow S_N=Q_N\cdots①$$

$$Q_{N-1}(x_{N-1},u_{N-1})=\frac{1}{2}x_{N-1}^TQ_{N-1}x_{N-1}+\frac{1}{2}u_{N-1}^TR_{N-1}u_{N-1}+\underline{V_N(x_N)}$$

↳ from ①: $V_N(x_N)=\frac{1}{2}x_N^TQ_Nx_N$
from linear dynamics: $\frac{1}{2}x_N^TQ_Nx_N=\frac{1}{2}(A_{N-1}x_{N-1}+B_{N-1}u_{N-1})^TQ_N(A_{N-1}x_{N-1}+B_{N-1}u_{N-1})$

$$=\frac{1}{2}x_{N-1}^TQ_{N-1}x_{N-1}+\frac{1}{2}u_{N-1}^TR_{N-1}u_{N-1}+\frac{1}{2}\left(x_{N-1}^TA_{N-1}^TQ_NA_{N-1}x_{N-1}+u_{N-1}^TB_{N-1}^TQ_NA_{N-1}x_{N-1}+x_{N-1}^TA_{N-1}^TQ_NB_{N-1}u_{N-1}+u_{N-1}^TB_{N-1}^TQ_NB_{N-1}u_{N-1}\right)$$

for find gain(k_k), have to find V_{k-1}.
for this, $\min_{u_{N-1}}Q_{N-1}$ needed.)

$$\frac{\partial Q_{N-1}}{\partial u_{N-1}}=R_{N-1}u_{N-1}+\frac{1}{2}B_{N-1}^TQ_NA_{N-1}x_{N-1}+\frac{1}{2}B_{N-1}^TQ_N^TA_{N-1}x_{N-1}+B_{N-1}^TQ_NB_{N-1}u_{N-1}$$

이때 R,Q의 positive definite조건 ⊕ symmetric으로 깔끔히 정리됨

$$=R_{N-1}u_{N-1}+B_{N-1}^TQ_NA_{N-1}x_{N-1}+B_{N-1}^TQ_NB_{N-1}u_{N-1}=0$$

변형으로
$u_k=k_kx_k+d_k$ 와 같은 형태가 될때
$u_k=k_kx_k$ 와 비교하여 안좋은점이 뭘까?
왜 연산을 항상 하나의 행렬곱으로 하고싶어할까?

$$\Rightarrow R_{N-1}u_{N-1}+B_{N-1}^TQ_NB_{N-1}u_{N-1}=-B_{N-1}^TQ_NA_{N-1}x_{N-1}$$

$$u_{N-1}=-(R_{N-1}+B_{N-1}^TQ_NB_{N-1})^{-1}B_{N-1}^TQ_NA_{N-1}x_{N-1}$$

$$\therefore u_{N-1}^*=k_{N-1}x_{N-1}\quad\left(k_{N-1}=-(R_{N-1}+B_{N-1}^TQ_NB_{N-1})^{-1}B_{N-1}^TQ_NA_{N-1}\right)$$

만약 $(R_{N-1}+B_{N-1}^TQ_NB_{N-1})$이 역행렬이 안된다면?
일반화하면 $(R_{k-1}+B_{k-1}^TS_kB_{k-1})$인데 얘가 역행렬이 무조건되나? 안될수도 있지않나?
역행렬이 되는조건은 뭐이지? 안된다면 어떻게 해야하지? 여기에 아마 positive-definite가 쓰일듯한데..

for N-2 step's computation,
we have to find S_{N-1})

$$V_{N-1}=\frac{1}{2}x_{N-1}^TS_{N-1}x_{N-1}=Q_{N-1}(x_{N-1},k_{N-1}x_{N-1})$$

$$=\frac{1}{2}x_{N-1}^TQ_{N-1}x_{N-1}+\frac{1}{2}(k_{N-1}x_{N-1})^TR_{N-1}(k_{N-1}x_{N-1})+\frac{1}{2}\left(x_{N-1}^TA_{N-1}^TQ_NA_{N-1}x_{N-1}+(k_{N-1}x_{N-1})^TB_{N-1}^TQ_NA_{N-1}x_{N-1}\right.$$
$$\left.+x_{N-1}^TA_{N-1}^TQ_NB_{N-1}(k_{N-1}x_{N-1})+(k_{N-1}x_{N-1})^TB_{N-1}^TQ_NB_{N-1}(k_{N-1}x_{N-1})\right)$$

$$=\frac{1}{2}x_{N-1}^T\left(Q_{N-1}+k_{N-1}^TR_{N-1}k_{N-1}+A_{N-1}^TQ_NA_{N-1}+k_{N-1}^TB_{N-1}^TQ_NA_{N-1}+A_{N-1}^TQ_NB_{N-1}k_{N-1}+k_{N-1}^TB_{N-1}^TQ_NB_{N-1}k_{N-1}\right)x_{N-1}$$

$$\therefore S_{N-1}=Q_{N-1}+k_{N-1}^TR_{N-1}k_{N-1}+\underline{A_{N-1}^TQ_NA_{N-1}+k_{N-1}^TB_{N-1}^TQ_NA_{N-1}+A_{N-1}^TQ_NB_{N-1}k_{N-1}+k_{N-1}^TB_{N-1}^TQ_NB_{N-1}k_{N-1}}$$

$$=(A_{N-1}+B_{N-1}k_{N-1})^T\underline{Q_N}(A_{N-1}+B_{N-1}k_{N-1})$$

↳ for generally, it is S_N.

Now, we have S_{N-1}, we can compute for k_{N-2} by doing same process with N-1 step's

for generalization) back ward : from $k=N$ to $k=1$

if it is first computation from $t=N$. $S_N=Q_N$

$$k_{k-1}=-(R_{k-1}+B_{k-1}^TS_kB_{k-1})^{-1}B_{k-1}^TS_kA_{k-1}$$
$$S_{k-1}=Q_{k-1}+k_{k-1}^TR_{k-1}k_{k-1}+(A_{k-1}+B_{k-1}k_{k-1})^TS_k(A_{k-1}+B_{k-1}k_{k-1})$$

if you done, you will get a list of k contains $[k_1,\cdots,k_{N-1}]$
: have to compute N-1 times

limitation of this method)

① can't handle constraints
ex) $u_k>2$

② Only for linear system. : this can handle with iLQR

forward : from $k=1$ to $k=N$

$$u_k^*=k_kx_k$$
$$x_{k+1}=Ax_k+Bu_k^*$$

if you done, you will get two lists of u,x
$[u_1,\cdots,u_{N-1}]$
$[x_2,\cdots,x_N]$
and x_1 is already known.

쓰면서 배운것)

① 밑의 개념에서 guidance를 처음에 nominal 이라고 썼었는데 좀 부적절한듯.
iLQR에선 nominal이 이전 iter에서 나온. 다음 iter의 시작점이라는 개념이니까.
guidance 라고 쓰는게 더 적절해보임.

② 처음엔 guidance를 x와 u 모두 원하는대로 줄수 있을 것이라 생각했는데 아님.
둘중하나만 guide로 줄수있음. 왜냐하면 $\delta x_{k+1}=A\delta x_k+B\delta u_k$로 쓰기위해서는
$x_{k+1_gui}=A_kx_{k_gui}+B_ku_{k_gui}$를 만족시켜야하기때문.

③ $Q_t(x_t,u_t)$ 항을 상수항, x_t가 양옆에 곱해진 이차항 등으로 분류할때, (ex. $H_{xx,t}$는 $Q_t(x_t,u_t)$에서 $x_t^T(\sim)x_t$의 $(\sim)$
$H_{u,t}$는 $Q_t(x_t,u_t)$에서 $(\sim)u_t$의 $(\sim)$)
gain k는 $k_t=-H_{uu}^{-1}H_{ux}=-H_{uu}^{-1}H_{xu}^T$로 나타낼수 있다. 변형하더라도
gain은 $Q_t(x_t,u_t)$의 계수(?)들로 구성된다.

* using guidance : follow trajectory

위의 objective function을 보면 x_k,u_k의 모든 요소가 작으면 작을수록 좋은 것으로 보인다. 즉, x를 생각하면 모든 state 값이 0에 가깝게 결과가 나올수록 좋다고 보고있는 것이다.
만약 이를 원하는게 아니라, 모든 시간대에 대해 x가 특정값을 추종하길 바란다고 해보자. 그러면 다음과 같이 계산하면 된다.

$$\begin{bmatrix}\delta x\triangleq x-x_{guidance}\\ \delta u\triangleq u-u_{guidance}\end{bmatrix},\ x_{k+1}-x_{k+1\_gui}=A_k(x_k-x_{k\_gui})+B_k(u_k-\underline{u_{k\_gui}})\Rightarrow\delta x_{k+1}=A_k\delta x_k+B_k\delta u_k$$

↳ 이는 $x_{k+1_gui}=A_kx_{k_gui}+B_ku_{k_gui}$으로부터 구해지는 값이다.

이후 objective function 역시 $\frac{1}{2}\delta x_N^TQ_N\delta x_N+\frac{1}{2}\sum_{k=1}^{N-1}\left(\delta x_k^TQ_k\delta x_k+\delta u_k^TR_k\delta u_k\right)$로 $\delta x,\delta u$에 대해서 세운다.
남은 전개과정을 모두 $\delta x,\delta u$에 대해 수행한후, $\delta u=u-u_{guidance}$이므로, 실제 u를 $u=\delta u+u_{guidance}$로 사용하면 된다. 물론 실제 x 역시 $x=\delta x+x_{guidance}$값이 나온다.
이러면 objective function 자체에서 $\delta x,\delta u$를 작게 만들려하므로, $x_{guidance}$를 추종하는 경로가 만들어진다.