

1(1) 다음과 같은 unstable system은 짧자

$$x_{k+1} = \begin{bmatrix} 1.1 & -0.2 \\ -1 & 1 \end{bmatrix} x_k + \begin{bmatrix} 0.3 \\ 0.2 \end{bmatrix} u_k$$

이때 $\begin{bmatrix} 1.1 & -0.2 \\ -1 & 1 \end{bmatrix}$ 의 Eigenvalue는 각각 1.5, 0.6이다. Eigenvalue가 1보다 큰 것이 충족된다.

1(2) $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ 와 하면 $x \in [-1, 1]^2$, $u \in [-0.7, 0.7]$ 이라 짧자

1(3) 해당 문제에서 만족 가능한 recursively feasible 초기 x_f 와 u_f 를 찾다면 ARE에서 계산된 controller

바탕의 \mathcal{O}_{∞} 와 weight Matrix P_{∞} 를 이용하면 된다. (\mathcal{O}_{∞} : maximal positive invariant set).

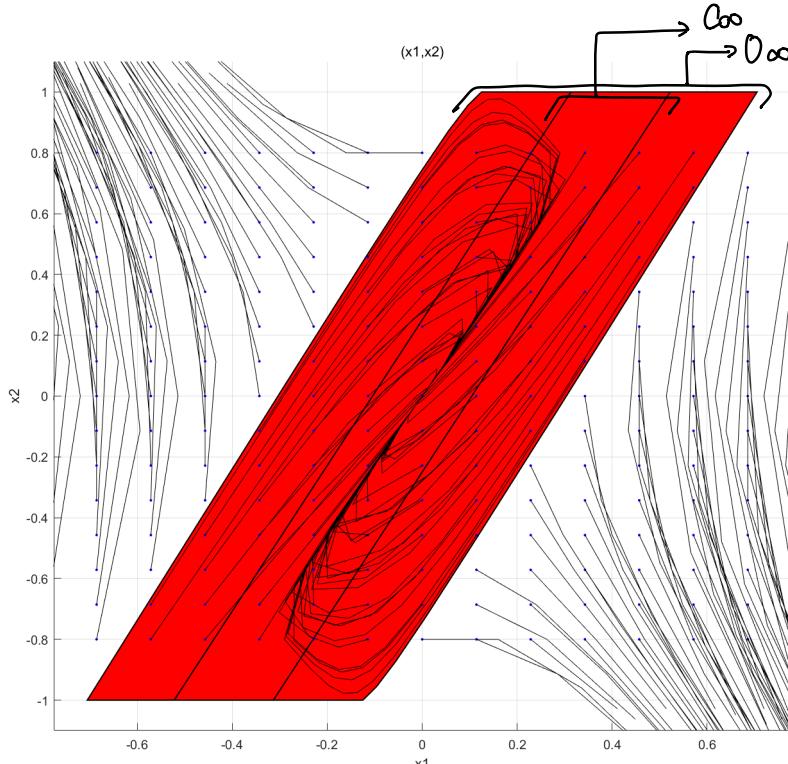
즉 $x_N = 0_{\infty}$, $p = p_{\infty}$ 로 짧으려면 보장. 해당 set을 구하는 code는 section 1에 있다.

1(4) ① Implicit MPC using different initial state

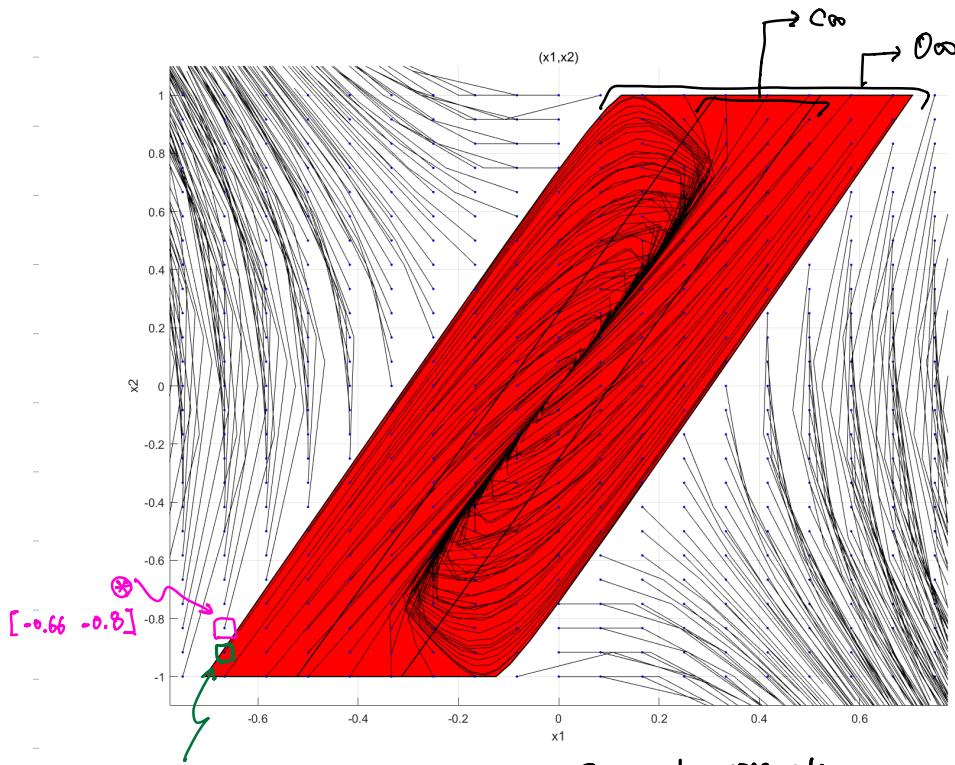
해당 문제에서 다음과 같이 cost Matrix를 setting하고 향했다.

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, R = 0.05, N = 10$$

그리고 initial state는 $x \in [-1, 1]^2$ 에 포함되는 서로 다른 25개의 점에 대해서 조사를 해보면 다음과 같다.



[Figure 1. x 에 속한 시도 다른 100개의 경]



[Figure 2. initial point은 더 많이 있는 경우]
[[-0.66 -0.8], [-0.66 -0.91]]

이때 가장 외부에 있는 polyhedron of Maximal Control invariant set (C_∞)가 I 부분에 있는 polyhedron of

C_∞ 이다. 이때 대응의 절을 보면 확연히 가로하였다.

- ① C_∞ 내부에 있는 initial point는 D로 표기한다 }
- ② C_∞ 외부에 있는 initial point는 B로 표기한다 } 각각 initial points들의 차이점/공통점
- ③ C_∞ 외부에 있는 initial point는 Diverge set

@ implicit MPC using different horizon length.

즉 2가지 종류의 initial point와 대체 설정을 전개해보았다.

첫번째 initial point는 C_∞ 에 내부에 들어오는 위의 ① 포인트라고 두번째 point는 ② 포인트다.

첫번째 initial point인 $\begin{bmatrix} -0.66 \\ -0.91 \end{bmatrix}$ 에 대해 (third point는 Figure 2에 ③인 초록색 점으로 표시)

$N = [3 5 7 8 9 10 11 13 15 30 45]$ 일때 closed loop trajectory를 그려면 다음과 같다.

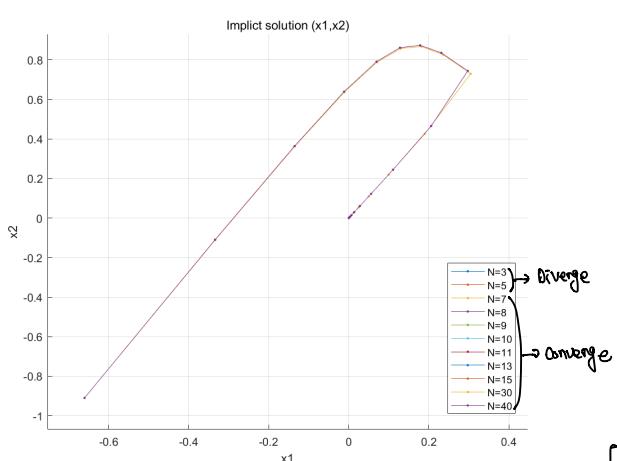


Figure 3. closed trajectory ab...t N I.

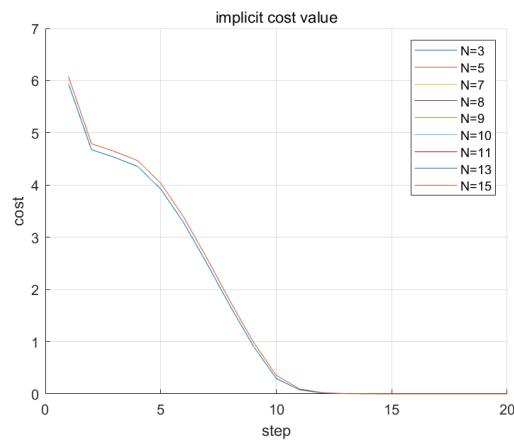


Figure 3-2. cost value of N that makes Converge trajectory 1

I 경우 N=3, 5 인 경우 해당 implicit MPC 의 Solution 이 존재하지 않는다.

그리고 N=7 ~ 40 의 경우 모든 N에 대해 비슷한 trajectory 을 따라서 원점으로 수렴함을 알수 있다.

만약 N에 따라 원점으로 수렴하는 trajectory 가 2개로 나뉘는 것을 알수 있다. (실제로 수렴하는 N은 7, 8, 9, 10, 11, 13, 15, 30, 40 으로 총 9개이지만 trajectory 가 서로 경에서 2개로 브인다.)

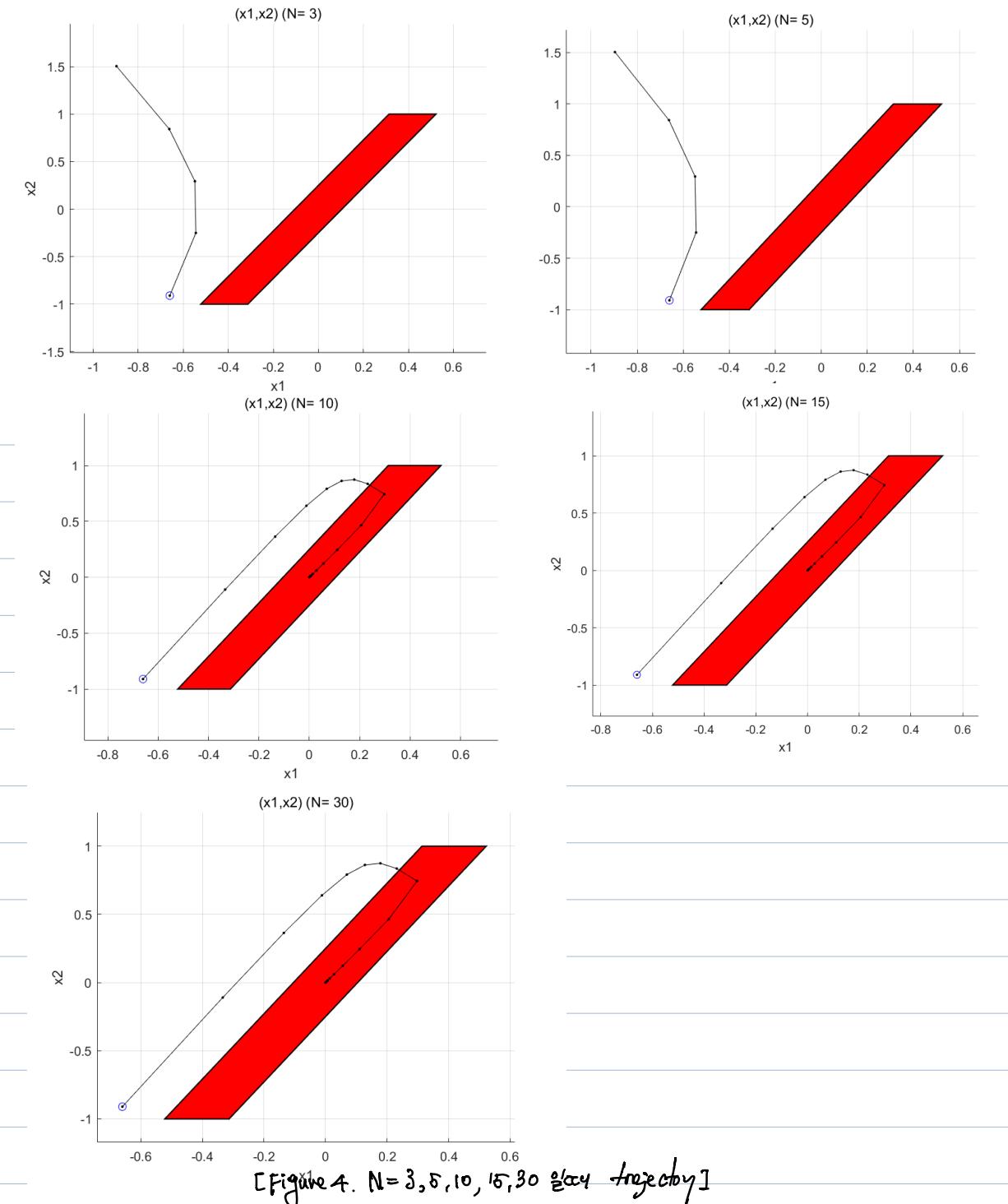
따라서 위 결과를 통해 적당한 N에 대해 원점으로 수렴하는 initial state에 대해 그 N보다 작거나

N을 잡는다면 (해당 문제에서는 N=3, 5로 잡는 경우) feasibility 가 성립하지 않을 수 있다.

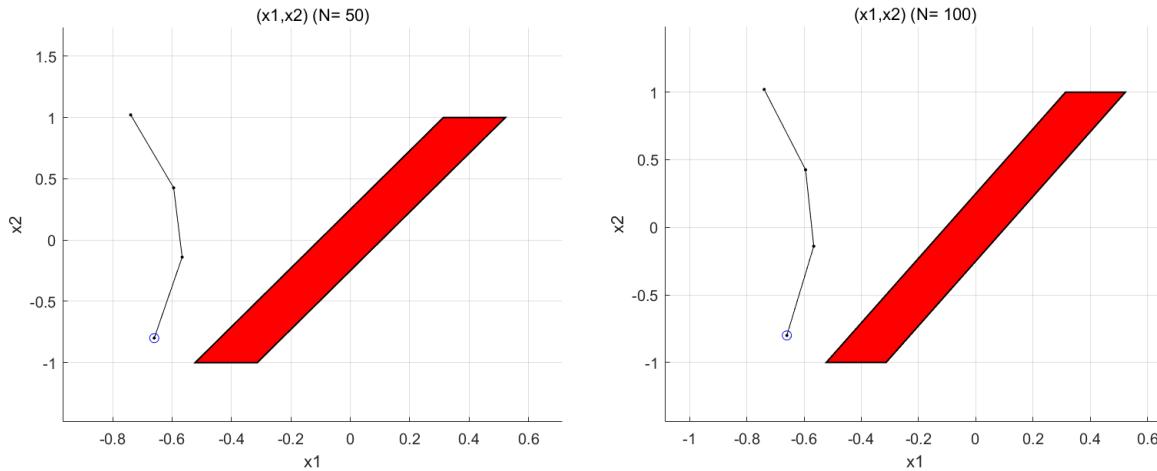
그리고 특정 N을 증가시키면 feasibility 가 보장이 되고 (=즉 trajectory 가 origin 으로 수렴하고)

N에 따라서 Converge 하는 trajectory 가 약간씩 차이가 있음을 확인할수 있다.

같 [Figure 4]에서는 동일한 initial state에서 N=3, 5, 10, 15, 30 인 경우 trajectory 를
보았다.



이제 2원짜리 point 인 경우는 $N = [3 5 10 50 100]$ 으로 흐름을 하였다. $N=50, 100$ 일때 (x_1, x_2) trajectory 을 그려면 다음과 같다.



[Figure 5. $N=50 / N=100$]

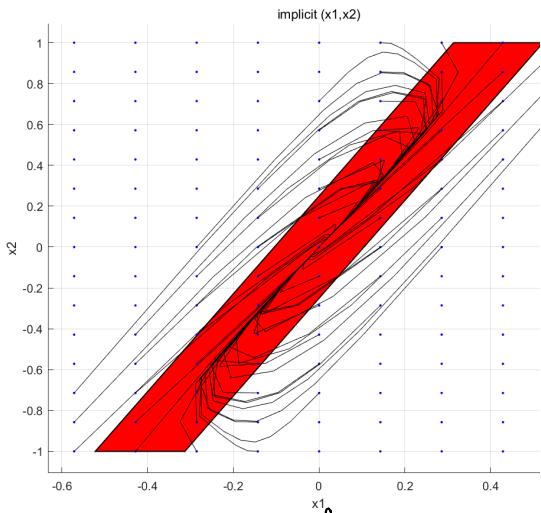
결과적으로 $N=50, N=100$ 인 경우에도 initial point가 C_∞ 외부에 있다면 원점으로 수렴하지 않을수록 흘러갈 수 있다. 이에 대한 해설은 간단하다. C_∞ 의 정의가 $x \in C_\infty \rightarrow \text{Next } x \in C_\infty$ 인 것의 모든 값을 포함하는 것이다. C_∞ 가 아닌 set에서 limit을 고를 경우 $x_k \notin C_\infty$ 인 $k \rightarrow \infty$ 존재하여 ($x_k = x_{k+n}$) 원점으로 x_k 가 수렴하지 않을수 있다.

1(5). 해당 위 문제에서 initial point가 $\begin{bmatrix} -0.66 \\ -0.91 \end{bmatrix}$ 인 경우에 대해서 해당 문제를 explicit MPC로 풀어보았다. Explicit MPC의 경우에도 1(a)에서 진행한 2가지 방법으로 동일하게 진행하고 computation time은 40, 15s 를 넘김해보았다.

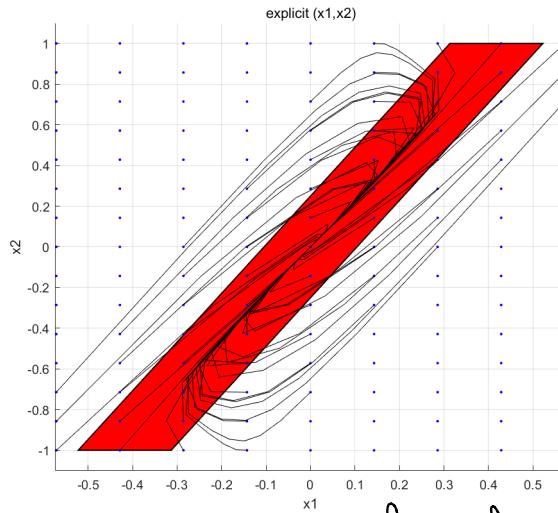
① explicit MPC using different initial state

해당 ①에서도 1(c)의 ①에서 진행한 동일한 initial state 100개에 대해 분석을 진행하였다 (figure 6)

그 결과 implicit MPC로 푼 경우와 explicit MPC로 푼 경우 feasible point의 여부가 같은지를 확인할 수 있다. (figure 6 와 figure 7 비교). 따라서 해당 MPC를 implicit form으로 풀든, explicit form으로 풀든 해당 문제의 feasibility는 동일하다고 볼 수 있다.



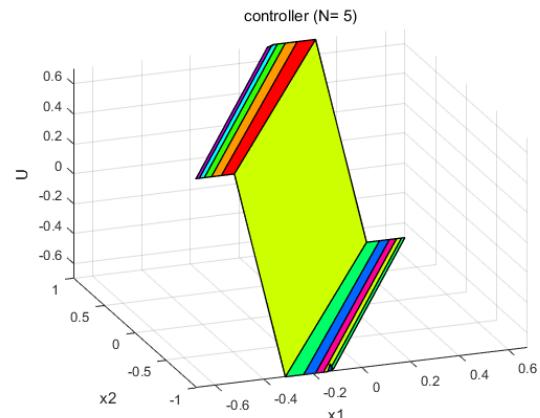
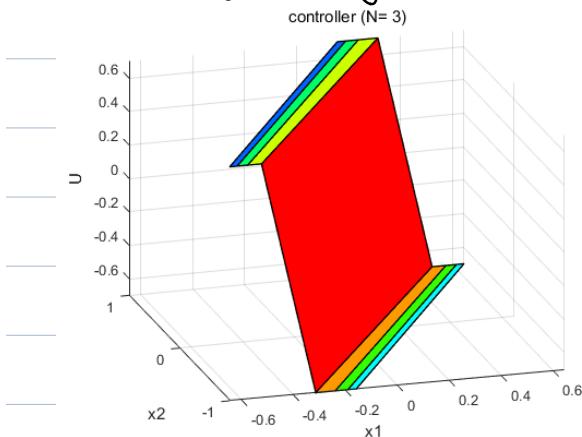
[Figure 6. Implicit MPC feasible initial points]



[Figure 7. Explicit MPC feasible Points]

② explicit MPC using different horizon length.

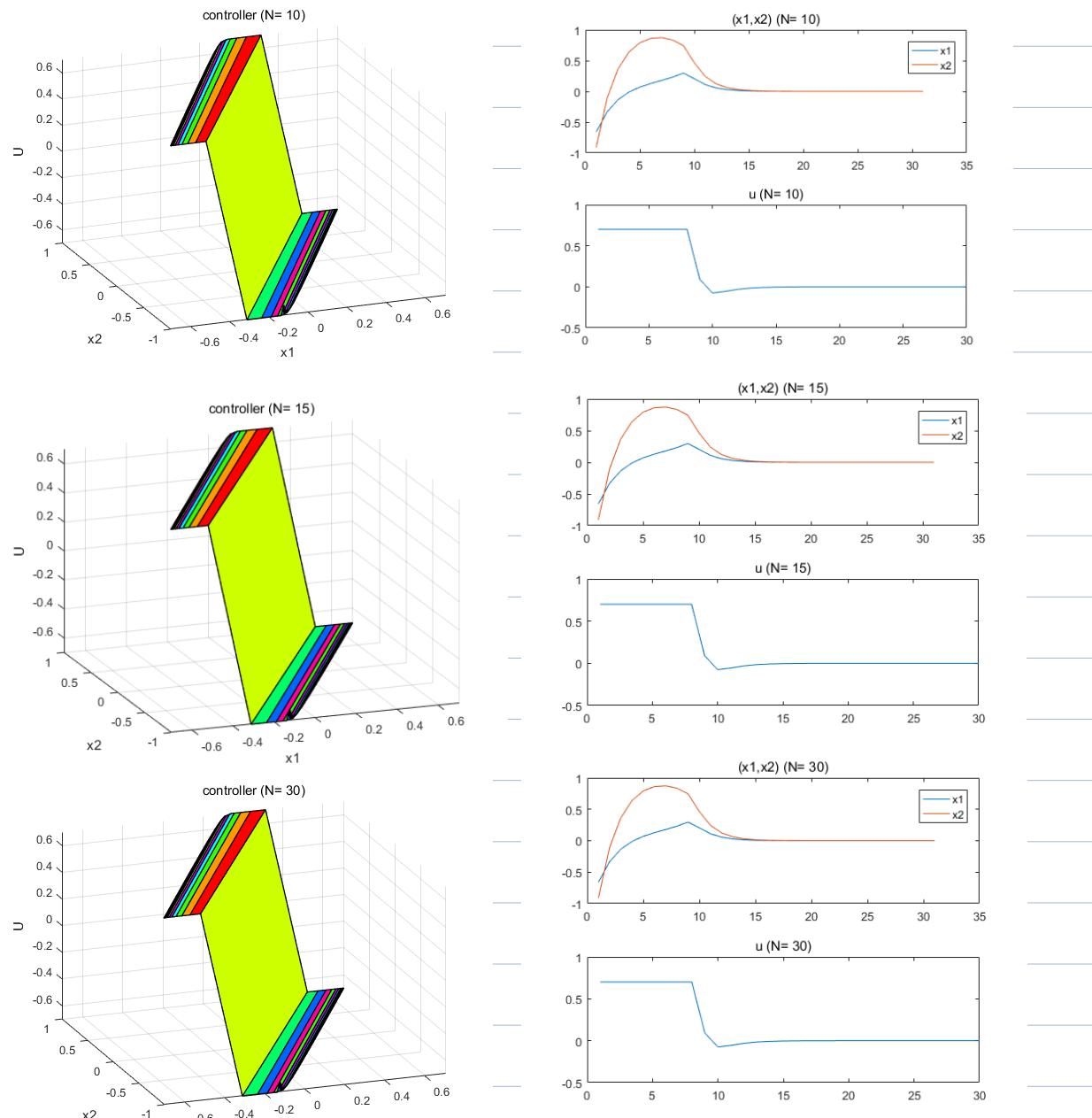
이전 explicit MPC 와 implicit MPC 를 비교하기 전에 explicitMPC 의 controller 을 $N=3, 5, 10, 15, 30$ 일때 살펴보자. initial state 은 implicit MPC controller 사용할 때 $\begin{bmatrix} -0.66 \\ -0.91 \end{bmatrix}$ 로 잡았다. 그 결과를 figure 8, figure 9 를 통해서 확인이 가능하다.



[Figure 8, 9. Explicit MPC when N=3, 5]

그 결과 $N=3, 5$ 일때 closed loop solution 이 존재하지 않았고 $N=10, 50, 100$ 일때 closed loop solution 이 존재했다. 각 N에 대해 explicit mpc controller 와 closed loop simulation 결과는 다음과 같다.

for $N=3, 5 \rightarrow$ no closed loop simulation.

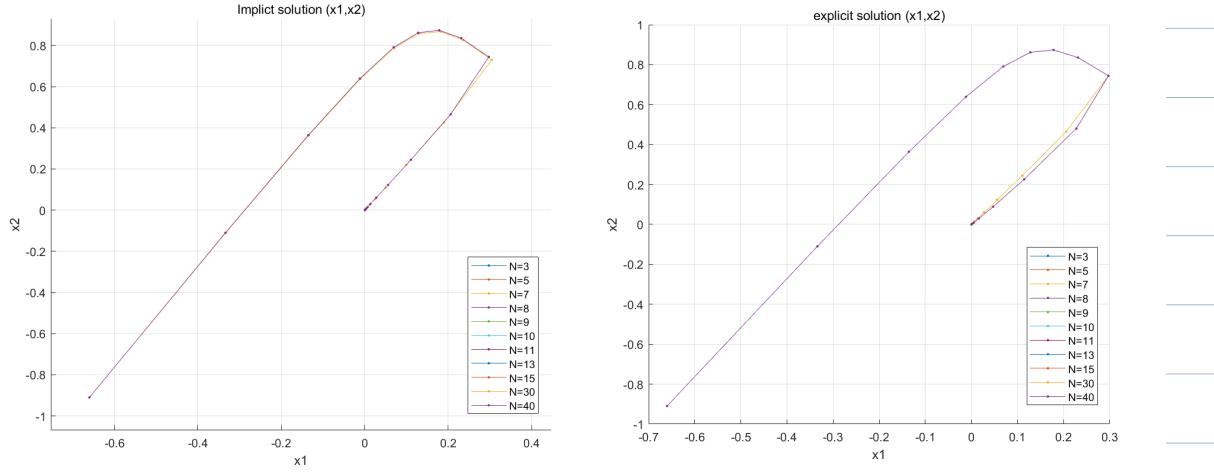


[Figure 9. Explicit MPC controller & closed loop simulation]

각 N 에 따른 critical region의 개수는 다음과 같다.

	$N=3$	$N=5$	$N=10$	$N=15$	$N=30$
CR regions	7 regions	19 regions	119 regions	257 regions	293 regions

이제 implicit MPC와 explicit MPC의 비교를 위해 explicit MPC로 $N = [3, 5, 7, 8, 9, 10, 11, 13, 15, 30, 40]$ 인 경우 (x_1, x_2) 의 trajectory를 살펴보았다.



[figure 10. implicit MPC와 Explicit MPC의 (x_1, x_2) trajectory].

fig 10 & 10-2 을 통해서 implicit MPC와 Explicit MPC의 (x_1, x_2) trajectory가 매우 유사성을 알 수 있다. 따라서 N이 정해졌다면 Solution의 stability는 implicit MPC와 Explicit MPC에 우위하다는 것을 알 수 있다.

따라서 figure 6, 7, 10 을 통해 분석되었던 앞선 결론을 다시 정리하자면 결국 implicit solution과 explicit solution은 해당 MPC optimization 문제를 어떻게 접근하는지에 따라 분류되는 Solution으로 결국 2가지를 통해 나온 Solution은 같아야 한다. 즉 implicit/explicit MPC solution의 stability / recursive feasibility / performance은 같아야 하고 실제 시뮬레이션으로 이를 확인하였다.

③ implicit / explicit MPC의 computation time 비교.

implicit MPC와 explicit MPC의 computation time은 다음 2가지 시간을 계산해서 성능을 비교하였다.

[time①]: mpc controller $U(x)$ 을 계산하는 시간 (type 1 time 이라고 함)

[time②]: ①을 끝낸 뒤 수행될 때까지 (x_1, x_2) points를 계산하는 시간 (type 2 time 이라고 함)

위 ①, ② 시나리오를 계산하는 코드는 아래와 같다.

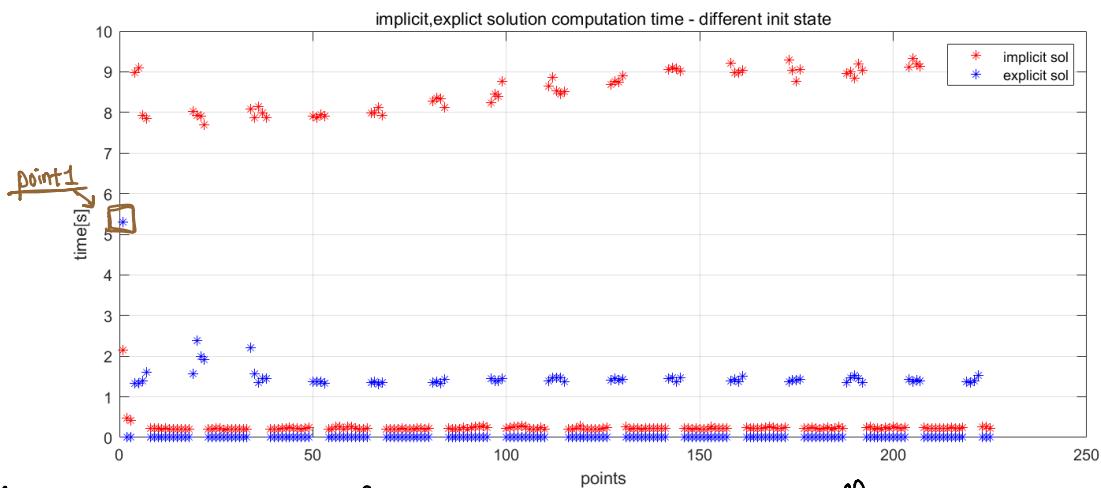
③-1. initial state 이 달라지는 경우 implicit / explicit Solution의 Computation time

```
time1 = clock;
if (string == "implicit")
    mpc = MPCController(sys, N);
    loop = ClosedLoop(mpc, sys);
elseif (string == "explicit")
    if (count == 1)
        mpc = MPCController(sys, N);
        exmpc = mpc.toExplicit();
        loop = ClosedLoop(exmpc, sys);
    end
else
    fprintf("ERROR \n");
end

Nsim = 1000;
data = loop.simulate(x0, Nsim); : type 2 time
time2 = clock;
timeElapsed = etime(time2, time1); → 결국 이 경우 type1 time + type2 time 을 계산.
```

[figure 11. computation time of implicit/explicit solution when different initial state]

위의 결과는 아래와 같다.



[figure 12. computation time of implicit/explicit solution when different initial state]

위 figure 12를 해석하면 다음과 같다.

- ① implicit Solution의 경우 매번 initial state 이 달라짐에 따라 동일한 MPC 문제를 풀더라도 계속 반복적으로 풀어야 한다. 이는 매번 모든 225개의 initial point들 중 feasible point의

$(x_0, u_0) \sim (x_N, u_N)$ 을 구하는 계산시간이 약 B초~우초 정도 동안 확인할 수 있다.

② Explicit solution 의 경우 동일한 MPC 문제를 풀고 매번 initial state 이 달라지므로 가

figure 12에서 □로 표기된 점의 경우 Explicit MPC 을 함수로 표현하고 (약 5.5초)

그리고 그 계산된 explicit mpc function 으로 두의 initial state 들의 $(x_0, u_0) \sim (x_N, u_N)$

을 계산한다. 따라서 이 경우 implicit Sol 보다 훨씬 적은 시간이 소요된다 (약 1~2초)

③-2. Horizon length가 달라지는 경우 Implicit / explicit Solution 의 Computation time

```
if (string == "implicit")
    time1 = clock;
    mpc = MPCController(sys, N);
    loop = ClosedLoop(mpc, sys);
    time2 = clock;
    timeElapsed1 = etime(time2, time1);
else if (string == "explicit")
    time1 = clock;
    mpc = MPCController(sys, N);
    expmpc = mpc.toExplicit();
    loop = ClosedLoop(expmpc, sys);
    time2 = clock;
    timeElapsed1 = etime(time2, time1);
else
    fprintf("ERROR %n");
end

Nsim = 1000;

time1 = clock;
data = loop.simulate(x0, Nsim);
time2 = clock;
timeElapsed2 = etime(time2, time1);
```

implicit Sol의 type1 time 짧은
(= MPC controller 소요시간)

explicit Sol의 type1 time 짧은
(= MPC controller 소요시간)

type2 time 짧은
(즉 $(x_0, u_0) \sim (x_N, u_N)$ 을 구하는데 걸리는 시간)

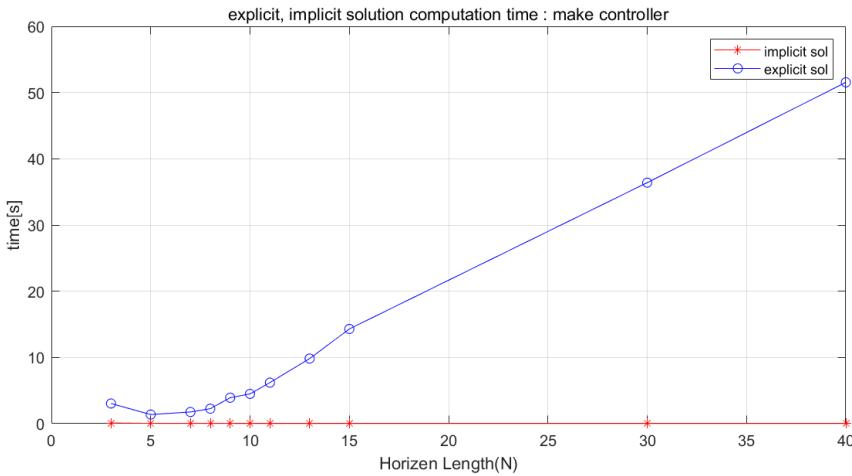
[figure 12. computation time of implicit/explicit
solution when different horizon length]

위의 결과는 아래와 같다. Figure 13 은 type1 time of MPC controller 를 계산하는데 걸리는 시간을

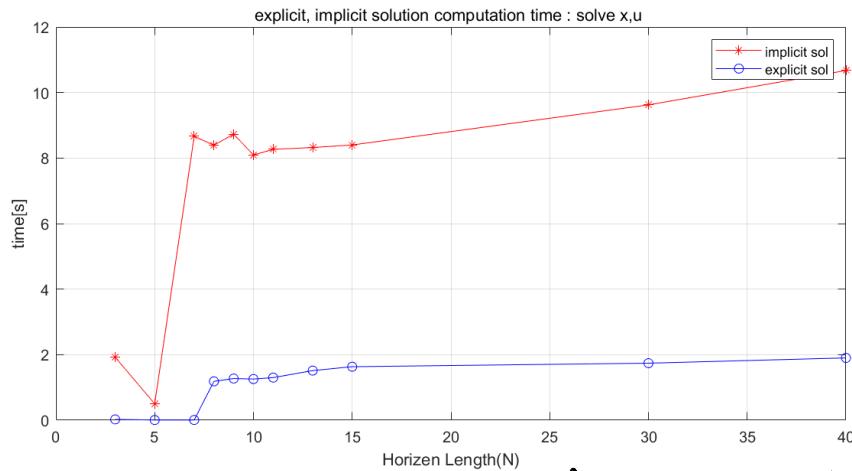
implicit Solution 와 explicit Solution 을 구분해서 표시해보았고, Figure 14 은 type 2 time 인 계산된

Controller 를 바탕으로 $(x_0, u_0) \sim (x_{N-1}, u_{N-1})$ 와 x_N 을 계산할때까지 걸리는 시간을 implicit solution

와 explicit solution 을 구분해보았다.



[Figure 13. computation time (type1 time) of exp, imp solution]

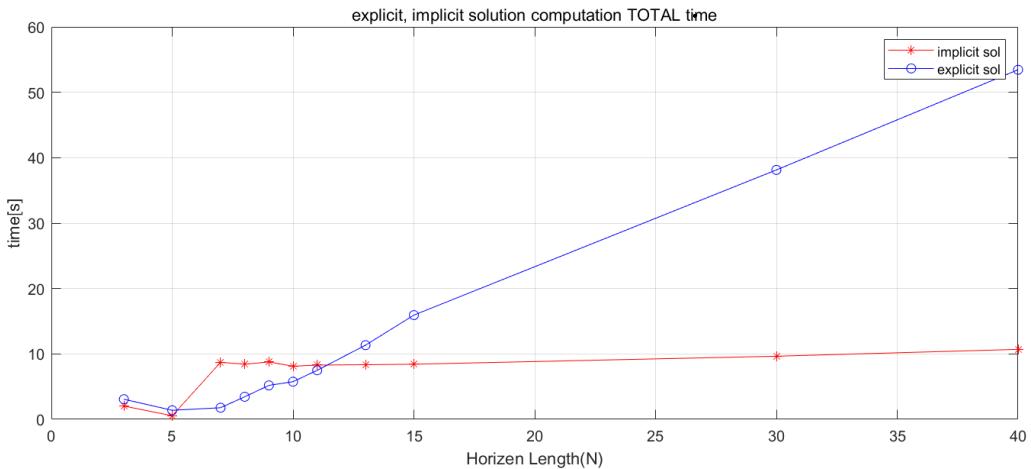


[Figure 14. computation time (type2 time) of exp, imp solution]

이 경우 해석은 아래와 같다.

- ① Figure 13을 보면 동일한 initial state인 $\begin{bmatrix} -0.66 \\ -0.91 \end{bmatrix}$ 에 대해서 $N=3 \sim 40$ 까지 MPC controller를 계산하는 시간을 graph로 표시하였다. 이 경우 explicit MPC는 N이 커질수록 Computation time이 점점 증가하지만 implicit MPC의 경우 1 시간은 N이 무관하게 거의 일정하고 매우 짧음(0.5초)임을 알 수 있다. 이는 다른 이유는 explicit MPC는 offline으로 $u(x)$ 를 함수로서 계산하는데 N이 커지면 계산수를 계산하기 어려워지며 한 차원의 커짐은 Computation time이 점점 증가한다. 이에 비해 implicit MPC는 다음 Step의 x를 계산할 Controller $u(x)$ 를 계산하면 되므로 explicit MPC 계산보다 훨씬 빠른다.

② Figure 14을 보면 동일한 initial state인 $\begin{bmatrix} -0.66 \\ -0.91 \end{bmatrix}$ 에 대해 N=3 ~ 40 까지 계산한 MPC controller 을 바탕으로 $(x_0, u_0) \sim (x_{N-1}, u_{N-1})$, x_N 을 계산하는 시간을 표시해보았다. 이 경우 explicit MPC는 N이 커질수록 일정하게, 모든 1~2초 사이의 시간이 걸리는 것을 확인할 수 있었고, implicit MPC는 N이 커질수록 조금씩 그 시간이 8초~12초 사이로 커짐을 확인할 수 있다. Figure 14에서 주목 해야할 점은 (x_i, u_i) 을 계산하는 과정에서 explicit MPC solution time이 implicit MPC solution time보다 훨씬 늦음을 확인할 수 있다. 이에 대한 이유는 explicit MPC의 경우 이미 알려진 controller는 $u(x)$ 형태를 이용해서 $u_0 \sim u_{N-1}$ 을 direct하게 함수 output으로 계산하기란 implicit controller는 BH time step ofct이 과정에 반복된다.



[Figure 15. TOTAL computation time (type 2 time) of exp, imp solution]

추가적으로 Figure 15의 결과 위 2개의 time을 합한 total computation time을 표현하였다.
따라서 explicit MPC와 implicit MPC의 computation time을 위와 같이 비교를 해본 결과 각 방법은 장단점, 특징은 아래와 같다.

① explicit MPC / implicit MPC 둘중 어느방식으로 풀면 Solution의 stability, feasibility, performance 은 통일된다.

② implicit MPC Solution은 N이 증가할 때 explicit controller를 계산이 자주적으로 증가하기에 큰 N에 대해 계산의 어려움으로 유용하다.

③ Explicit MPC Solution은 그와 다르게 N이 증가할 때, 혹은 online 계산에서 high computing power 을 소비 못할때 유용하다.