과제번호	4.26649.01

2023년도 학부생연구프로그램(UGRP)

최종보고서

과 제 명	유체 상태 측정 없이(sensor-less) mobile Manipulator를 이용해 가속상황에서의 안정된 유체 수송				
연구기간	2023. 5. 15. ~ 2024. 1. 9.				
연 구 비	**1,953,240원**				
지도교수	**성명**	한수희		**학과**	IT융합공학과
참여자 인적사항	**구분**	**학과**	**성명**	**학번**	**연락처**
	연구 책임자	IT융합공학과	이도현	20200988	010-8531-0750
	공동 연구원	물리학과	김종열	20190875	010-9474-9715
	공동 연구원	기계공학과	신민경	20210553	010-8981-2871
	공동 연구원				

학부생연구프로그램 최종보고서를 아래와 같이 제출합니다.

2024년 1월 9일

연구책임자(팀장): 이도현 ( )

연구지도교수: 한수희 (인)

포항공과대학교 교육혁신센터장 귀하

※ 반드시 연구책임자(팀장)과 지도교수님의 서명 후 제출해 주세요. (자필 또는 전자서명)

2023학년도 학부생연구프로그램(UGRP)

연구비 집행 결산보고

연구책임자(팀장)	이도현	소속/학번	IT융합공학과/20200988
과제명	유체 상태 측정 없이(sensor-less) mobile Manipulator를 이용해 가속상황에서의 안정된 유체 수송		

■ 연구비 집행표

세부 비목	당초 예산(원)	최종 집행 예산(원)
연구기자재	0	0
시설비 및 임차료	0	0
시험분석료	0	0
시제품 제작비	1,964,612	1,953,240
수용비 및 수수료	0	0
기술정보활동비	0	0
합계	1,964,612	1,953,240

☞ 세부 비목 추가 가능

☞ 연구비 예산에는 연구비로 배정받은 금액을, 집행한 연구비는 중 최종 집행한 연구비를 기입함

2023학년도 학부생 연구프로그램(UGRP)

최종보고서

연구책임자(학생)	이도현	소속/학번	IT융합공학과/20200988
과제명	유체 상태 측정 없이(sensor-less) mobile Manipulator를 이용해 가속상황에서의 안정된 유체 수송		
연구계획 대비 완성 정도: 90 % / 100 %			

1. 연구 목적

Liquid container transfer system 에 관한 연구는 과거부터 현재까지 굉장히 많은 연구가 진행되어 왔다. (YanoK, 1996) (HAMAGUCHIM,, 2023). 대부분의 sloshing 관련 연구는 물컵과 같이 닫혀 있지 않은 용기 속 액체를 목표지점까지 이동시킬 때, 액체를 쏟거나 흘리지 않으면서도 빠르게 목표지점까지 이동시키는 것을 주 목적으로 한다. 최근 서빙 로봇이 상용화됨에 따라, 안정적인 유체 수송에 대한 관심도 역시 높아졌고, 보다 실용적인 방향으로 연구가 많이 진행되고 있다.

현재까지 진행된 sloshing 연구들은 용기 속 유체를 pendulum, 혹은 간단한 dynamics로 근사하여, 해당 dynamics를 안정화하는 제어기를 설계해 적용하는 방식이다. (YanoK, 1996) 이를 위해서, 액체 표면의 기울기를 실시간으로 측정해 그 값을 feedback 제어기에 반영해야 한다. 따라서 이전 연구에서는 용기 끝의 액체 표면을 거리센서로 측정하거나 (HAMAGUCHIM,, 2023), level 센서를 용기 벽면에 부착 (YanoK, 1996), 혹은 카메라로 액체 표면 영상을 찍어 영상분석을 통해 표면이 얼마나 기울어져 있는지를 파악한다. 이렇게 액체 표면의 상태를 측정하여 제어기의 feedback loop에 값을 반영시키면 비교적 쉽게 이동 중에도 액체 표면을 안정화시킬 수 있다. 그러나 이러한 시스템을 실생활에 적용시킨다 가정했을 때, 옮겨야 할 유체 내에 항상 센서를 집어넣어야 하거나, 카메라를 컵 앞에 설치해야 한다. 단적인 예시로 서빙 로봇에 연구를 적용시킨다고 가정했을 때, 손님이 마실 컵 속에 센서를 집어넣기는 어려울 것이다.

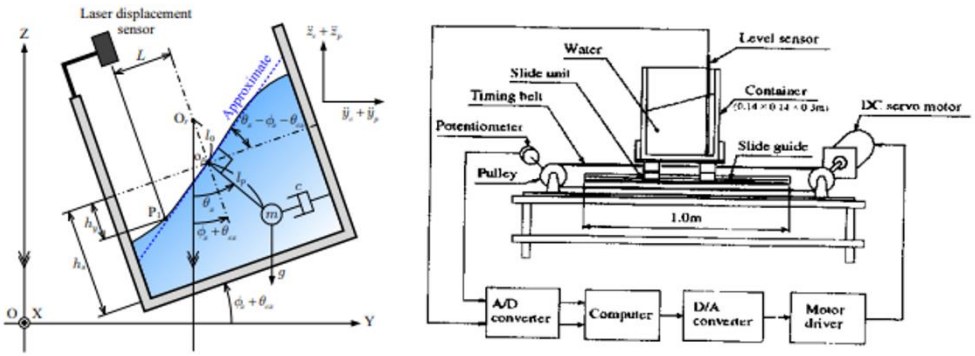

그림 1. laser, level sensor 등을 활용해 물의 상태를 측정한 이전 연구들

또한, 기존 연구들 대부분이 이동 시스템의 가속도를 input으로 두어 sloshing 안정화 제어를 설계한다. 간단하

게 예시를 들자면, 컵 속의 물이 앞으로 쏠리게 되면 그 물컵을 싣고 있는 카트가 빠르게 앞으로 이동해 반대방향으로 가속도를 만들어내어 물이 쏟아지지 않게 제어하는 식이다. 이러한 방식은 이동해야 할 목표 지점이 설정되면, 그 지점까지 어떻게 이동해야 액체가 적게 흔들릴지를 제어기가 결정한다. 이러한 제어 방식의 문제점은 이동 속도, 가속도, 경로에 대해 자유도가 떨어진다는 점이다. 목표 지점을 설정하는 순간 제어기가 로봇의 이동 경로, 속도, 가속도를 결정해 버리기 때문에, 원하는 속도나 경로로 이동할 수 없다는 점이다. 사람이 직접 카트를 끌고 가려고 한다면 이러한 제어기는 카트 위 유체를 안정화시킬 수 없다. 또 이러한 방식은 외부에서 카트에 충격이 전해져 물컵 속 물이 흔들리는 경우에 대해서 대응하기 힘들다는 것이 단점이다. Cart가 이동할 때, 이동에 의해서 액체가 흔들리는 것에 대해서만 안정화가 가능하다.

본 연구에서는 앞서 언급한 단점들의 해결책을 제시한다. 첫 번째로, 본 연구에서는 유체의 상태를 직접 측정하지 않는, sensor-less한 방식으로 유체를 제어한다. 이를 위해 우리는 유체의 움직임을 neural network통해 학습한다. network가 유체에 가해지는 가속도를 Input으로 받아, 유체 표면의 기울기, 기울기의 각속도 $\theta, \dot{\theta}$를 output으로 내놓는다. 실시간으로 neural network를 통해 얻어지는 $\theta, \dot{\theta}$를 제어기의 feedback loop에 반영시킨다. 유체의 움직임을 estimate하려는 연구는 이전에도 있어왔지만, 대부분 Input 가속도를 완전히 아는 상태에서 ODE를 풀어 결과를 얻는 방식이었다. (GuagliumiL„ 2021) 이와 같은 방법론의 경우 예정된 가속도가 아닌 모르는 가속도가 물컵에 가해졌을 때에는 유체의 움직임을 예측할 수 없다.

일반적인 neural network로는 유체의 dynamics를 정확하게 예측할 수 없기에, 우리는 PIRNN(Physics-Informed recurrent neural network) 이란 방법론을 변형하여 적용하였다. (Zheng& Wu, ZY„ 2023), (ZhengY„ 2023). RNN을 활용하여 과거 데이터를 학습에 반영할 수 있도록 하였고, 학습 loss를 계산할 때 label과 얼마나 일치하는지 뿐 만 아니라, 우리가 알고 있는 유체의 dynamics와 얼마나 유사하게 network의 output이 나오는지 역시 계산하여 loss에 포함시켰다. 일반 RNN으로 학습시킨 것 보다 PIRNN을 적용했을 때 그 결과물이 train data 내에 없는 data에 대해 더 높은 예측도를 보임을 확인하였다.

Cart의 이동 자율성 및 velocity, acceleration에 대한 자유도를 부과하기 위해, 우리는 6축 manipulator를 이용해 sloshing control를 진행한다. Cart 위에 로봇팔이 물컵을 잡고 있는 형태이며, 덕분에 카트가 어떻게 움직이더라도 로봇팔이 물이 쏟아지지 않도록 제어가 가능하다. PIRNN을 통해 얻어진 $\theta, \dot{\theta}$ 과 로봇팔의 end effector의 position, velocity를 state로 하는 LQR제어기를 통해, θ와 end effector의 position을 최소화하도록 한다. 이렇게 설계된 제어기 덕분에 카트에 랜덤한 외란이 가해지거나 계획되지 않은 움직임을 보인다고 해도 그렇게 발생한 가속도에 PIRNN network가 반응해 유체의 상태를 estimate하며, 이를 이용해 로봇팔이 자신의 가동범위 내에서 액체를 쏟지 않도록 제어해 준다.

본 연구의 **main contribution**은 다음과 같다.

- **i.** Physics-informed 된 RNN을 설계하여, 유체의 움직임을 실시간으로 예측할 수 있는 네트워크를 개발, 기존 방법론들에 비해 우수한 성능
- **ii.** Sloshing control에서 최초로 Sensor 없이 network output만으로 LQR control이 가능함을 simulation뿐 아니라 실제 세계에서 Real demo로 보임. 물의 상태를 측정해야 하는 이전 연구들에 비해 실용성 측면에서 우수함. 또한, neural network를 state observer 역할로 하는 이러한 방식은 sloshing 뿐 아니라 다른 시스템에도 적용될 수 있음.
- **iii.** 바퀴형 모바일 로봇 혹은 벨트형 이동장치로 물통을 제어하던 이전 연구들과 달리, 본 연구에서는 mobile manipulator로 물통을 제어함. 이동의 trajectory (velocity, acceleration, position)에 대해서 자유로우며 외란에 유연하게 대응이 가능함. 또한 본 연구에서는 로봇팔의 endpoint를 점으로 지정해두고

그 점을 기준으로 control을 진행했지만, endpoint가 따라가야할 trajectory를 주고 그 trajectory를 따라가면서 sloshing control을 진행하는 등의 확장 역시 가능함. (ex. 물통을 옮기면서 외란에 대응해 물 흘리지 않도록 제어)

2. 연구내용 및 진행

센서 없이 유체의 움직임에 해당하는 값인 $\theta, \dot{\theta}$ 을 얻기 위해서, $\theta, \dot{\theta}$를 만들어내는 Neural network를 만들어야 한다. 해당 Neural Network는 유체에 가해지는 가속도를 Input으로 받으며, RNN모델을 기반으로 만들어진다. Loss의 경우 일반적으로 쓰이는 label과의 MSE(mean square error) 를 사용함과 동시에 Physics Informed 한 정보를 고려할 수 있는 Loss 역시 만들어 추가한다(PIRNN). 이렇게 만들어진 Neural Network를 통해 유체의 상태를 실시간으로 얻어낼 수 있게 되면, LQR (linear quadratic regulator) 제어를 통해 유체가 빠르게 0도로 수렴할 수 있도록 제어한다. 이때 LQR 제어의 state feedback 에 사용되는 state는 $\theta, \dot{\theta}, x, \dot{x}$이며, 안정화시키고자 하는 state는 θ, $x, \dot{x}$ 이다. 유체가 빠르게 안정화되어야 하므로 θ를 regulate해야 하며, 로봇팔의 가동범위가 제한되어 있기 때문에 x 역시 최소화하는 방향으로 제어한다. 우리가 사용한 로봇팔이 실질적으로 velocity tracking을 진행하는데, 모터가 낼 수 있는 최대속도가 낮기 때문에 가능한 저속으로 제어할 수 있도록 하기 위해 $\dot{x}$도 최소화 하는 방향으로 제어한다. 전체적인 control flow는 다음과 같다.

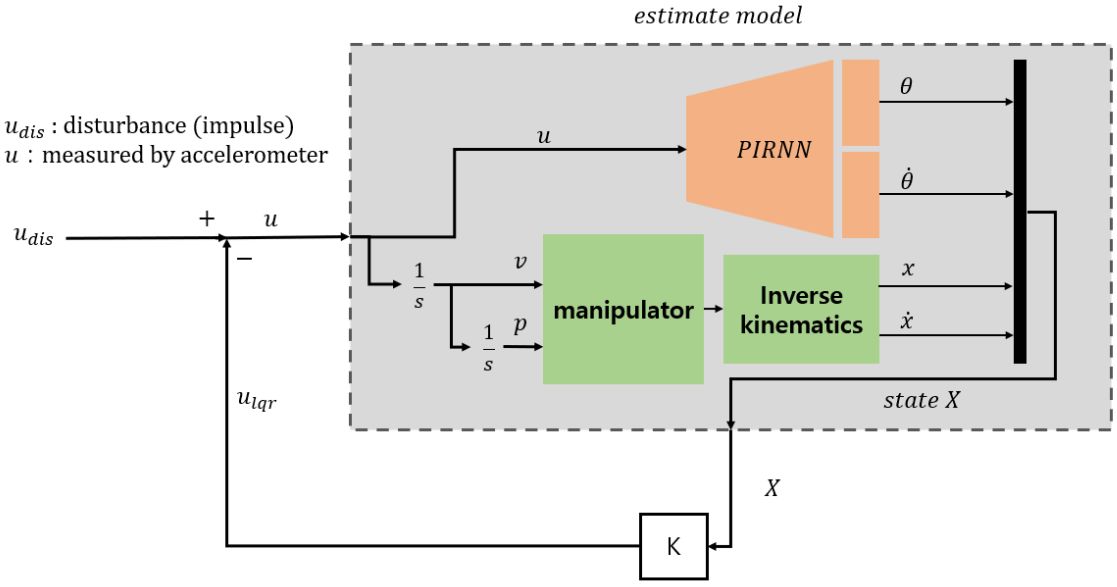

그림 2. Control flow

A에서 로봇 하드웨어, B에서 유체의 모델링, C에서 PIRNN 모델의 구조 및 training과정, D에서 LQR control 및 hardware deploy를 다루고자 한다.

A. Hardware

ROBOTIS 사의 XM540-W270 모터와 3D printing을 이용해 연구에 사용될 manipulator를 직접 제작하였다.

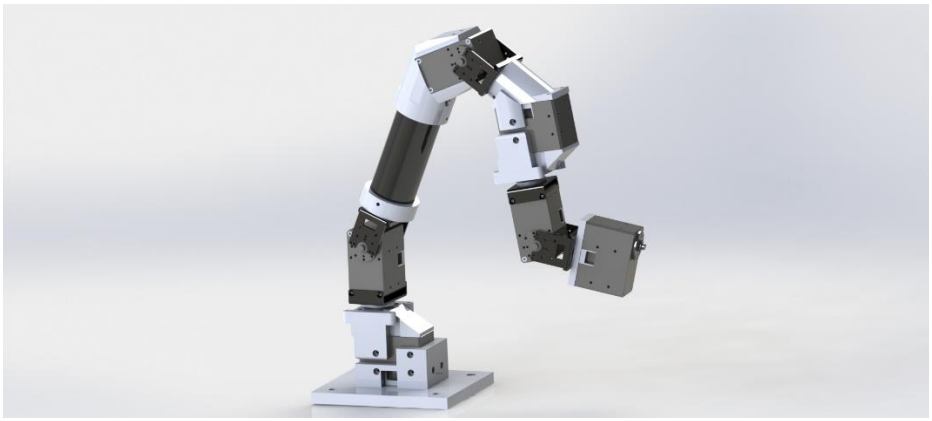

그림 3. 로봇팔 렌더링 이미지

이후 로봇팔이 들고 있을 유체에 외부 가속도 u_{dis}를 가해줄 수 있도록 모바일 로봇과 연결하였다. 전체적인 형태는 다음과 같다.

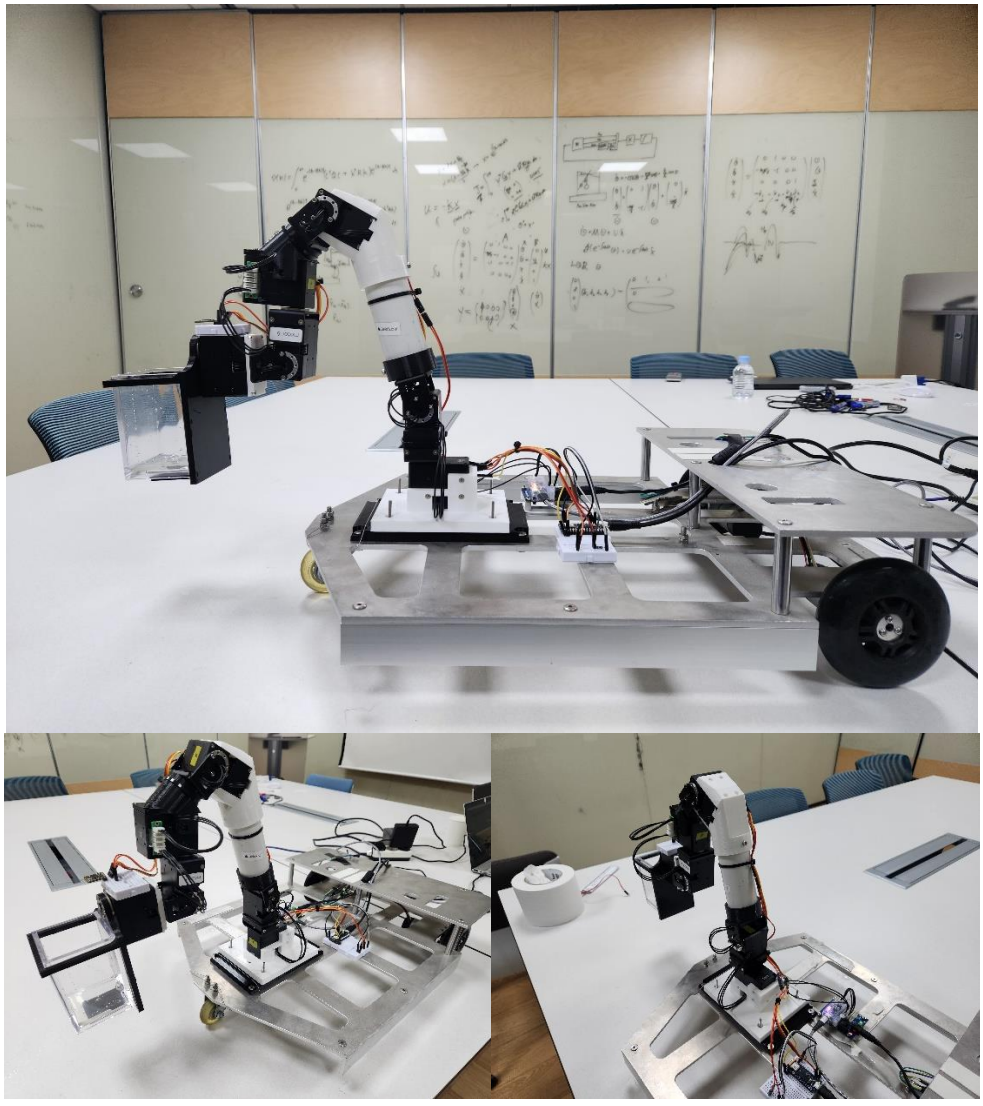

그림 4. Experiment setting (manipulator + mobile robot)

로봇팔의 end effector에는 가속도를 측정하기 위한 가속도센서(EBIMU)가 부착되어 있다. 컵 속 유체에 가해지는 가속도를 측정해 로봇팔 및 PIRNN에 정보를 제공한다.

Control loop내에는 포함되어 있지 않지만, 실험 결과를 얻기 위해 level sensor(수위 측정 센서)도 사용하였다.

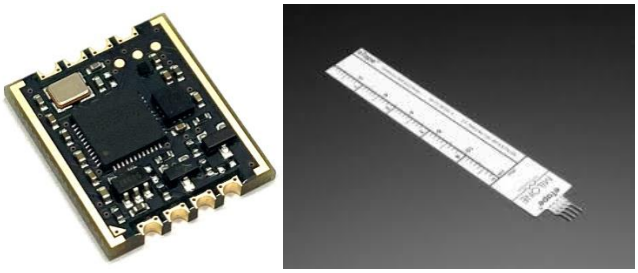

그림 5. 가속도센서(EBIMU) 와 level sensor (etape 5inch)

로봇팔에 사용된 모터는 position control, velocity control 까지 가능하다. Torque control은 가능은 하나 그 정확도가 떨어져 manipulator에서 사용하기에는 부적합하다. 따라서, impedence cotrol과 같이 torque input을 필요로 하는 control은 불가능 하며, 본 연구에서는 velocity trajectory 및 position trajectory만을 활용한다.

모든 센서 및 로봇 시스템은 ubuntu 22.04 LTS를 OS로 두고 있는 노트북 위에서 돌아가며, 센서와 로봇 간 통신은 ROS2 humble을 사용하였다. Hardware 작동 코드는 C++로 작성되었으며, PIRNN training 은 python으로 작동한다. libtorch라이브러리를 통해 python 코드로 만든 network model을 C++에서도 작동할 수 있게 해준다.

B. Manipulator-Fluid Modeling

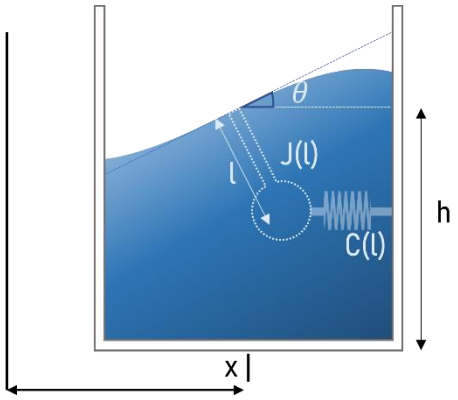

그림 6. 용기와 그 속에 담긴 유체가 외부 힘에 의해 진동하는 순간을 표현한 도식.
J는 진자로 간주하였을 시의 관성모멘트를 의미한다.

유체의 측면에서 외부에서 가해지는 요인은 유체를 담는 용기가 받는 힘이다. 그 결과로써 용기가 받는 가속도가 유체에 미치는 영향을 방정식의 형태로 계산해내는 것이 제어하고자 하는 대상을 모델링하는 것으로 이해하였다. 구체적으로 유체역학을 적용하는데 어려움이 있었으나, 유체 표면의 진동을 가상의 진자가 감쇠 진동하는 것으로 간주하여도 효과적으로 기술할 수 있음이 알려져 있음을 확인하였다. (YanoK., 1996), (GuagliumiL., 2021) 따라서, 본 연구에서는 유체를 진자로 취급하고, 용기의 가속도에 대한 유체의 상태 (유체 표면의 역학)를 다음과 같이 기술할 수 있었다.

$$\ddot{\theta} = -c(l)\dot{\theta} - \frac{mg}{l}sin\theta - \frac{\ddot{x}}{l}cos\theta \quad \text{(EQ1)}$$

위 식에서, $c(l)$은 진자 높이에 따라 변화하는 감쇠계수, m은 용기 내 유체의 질량, l은 유체를 진자에 대응시켰을 때의 길이이다. (YanoK, 1996)를 참고하여, 우리 실험 세팅에 맞는 계수로 결정하였다.

C. PIRNN

i. Training dataset

유체의 dynamics를 학습하기 위해, 우선 유체 시뮬레이터를 제작하였다. 유체의 움직임 구현은 B에서의 식을 사용하였다.

가능한 모든 상황을 dataset이 포함하고 있는 것이 좋다. 시뮬레이션 내 유체에는 크게 두가지 종류의 힘이 가해져, 그 힘과 그때의 $\theta, \dot{\theta}$를 csv파일로 만들어낸다. 첫번째 종류의 힘(force design 1)은 impulse형태의 힘이다. 일정 범위의 크기 안에서 랜덤한 크기의 impulse 힘을 유체에 가하도록 하였으며, 가해지는 timing, 횟수, 힘의 지속시간 역시 모두 랜덤이다. 두번째 종류의 힘(force design 2)은 LQR output이다. 앞서 impulse 힘에 의해 유체의 $\theta, \dot{\theta}$이 0에서 벗어나 진동하고 있을 때, LQR제어기가 적용되면 빠르게 그 두 값이 regulate된다. 제어기가 가하는 힘 역시 실제세계에서 경험할 것이므로 해당 data도 학습할 수 있도록 dataset에 포함시켰다. 첫 번째 종류의 힘은 랜덤하게 계속 가해지고, 랜덤한 타이밍에 랜덤한 시간동안 LQR output이 더해진다.

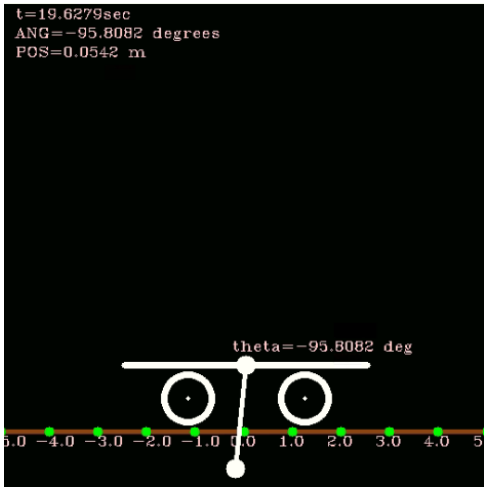

그림 7. 시뮬레이션 시각화 영상. 유체를 damper-pendulum model로 근사함

```python
def u(t):
    global global_h, jump_duration, stop_time, stop_count

    if jump_duration > t: # maintain amplitude during duration & reduce jump_duration
        a = global_h + 0.5 * np.random.randn()
    else:
        if t >= stop_time: # 2.5 # probability related parameter, (The larger the value, the lower the pro
            stop_time = t + 1 + 5*np.random.rand()
            global_h = 10 * np.random.rand() - 5
            if global_h > 0 and global_h < 0.5:
                global_h = 0.5
            if global_h < 0 and global_h > -0.5:
                global_h = -0.5
            jump_duration = t + 0.001 + 0.1*np.random.rand() # 900,1000 # duration related parameter, (The
            a = global_h + 0.5 * np.random.randn()
        else: # otherwise, just 0
            global_h = 0
            a = 0
```

Force design 1

```python
# Y : [ x, x_dot, theta, theta_dot]
def func3( t, y):
    global duration, choice,state,zero_dist
    g = 9.8
    L = 1./21.
    k = 0.58 # coefficients c/m
    LQR_u = 0
    if(t >= duration):
        duration = 4 + t
        zero_dist = y[0]
        choice = np.random.randint(0,10) # 0,1,2,3
    if(choice >= 1):
        state = 0
        LQR_k = [10.0000, 11.3488, -4.3096, -0.2765]
        LQR_state = np.array(y).reshape([4,1])
        LQR_state[0,0] = y[0] - zero_dist
        LQR_state[2,0] = y[2] + np.pi/2

        LQR_k = np.array(LQR_k).reshape([1,4])
        LQR_U = -LQR_k.dot(LQR_state)
        LQR_u = LQR_U.squeeze()
    else:
        state = 1
    he = 1
    if(state == 0):

        x_ddot = u(t) + LQR_u
        if he == 1 and (x_ddot > 10 or x_ddot < -10):
            he = 0
            x_ddot = u(t)
        else:
            x_ddot = u(t) + LQR_u


    elif(state == 1):
        x_ddot = u(t)

    theta_ddot = -k*y[3]*np.cos(y[2]+np.pi/2)-(g/L)*np.sin(y[2]+np.pi/2)-(x_ddot/L)*np.cos(y[2]+np.pi/

    # xdot, xddot, thetadot, theta_ddot
    return [ y[1], x_ddot, y[3], theta_ddot ]
```

Force design 2 & water dynamics

이렇게 힘을 디자인 해 모델에 가해주고 그 결과를 기록하여 dataset으로 만든다. 미분방정식을 푸는 데는 파이썬의 solve_ivp를 사용하였다. Dataset의 총 길이는 10분이다.

ii. Network structure

유체의 dynamics를 학습하는 것은 일반적인 Neural network로는 힘들다. 우리가 필요한 모델은 input으로 u를 받아 output으로 $\theta, \dot{\theta}$을 얻어내는 형태이다. 그러나 유체의 dynamics는 u와 $\theta, \dot{\theta}$ 이 일대일 대응이 되

지 않는다. Input u가 갑자기 0으로 된다고 해서 $\theta, \dot{\theta}$이 0으로 바로 되지 않는다. 또한 같은 힘을 가한다고 해도 이전 $\theta, \dot{\theta}$이 어떤 것이었는지에 따라 도출되는 $\theta, \dot{\theta}$값이 바뀐다. 즉 이전 $\theta, \dot{\theta}$값들 역시 network에 영향을 주는 요소들 중 하나이다.

일반적인 Neural network의 input에 이전 $\theta, \dot{\theta}$들을 쌓아서 input으로 넣어주는 방식도 있겠지만, 비효율적이고 쌓는 것에 한계가 있다. 따라서 이런 시계열 데이터를 학습하는 데 유리한 RNN(Recurrent Neural Network)구조를 이용한다. RNN의 구조는 다음과 같다.

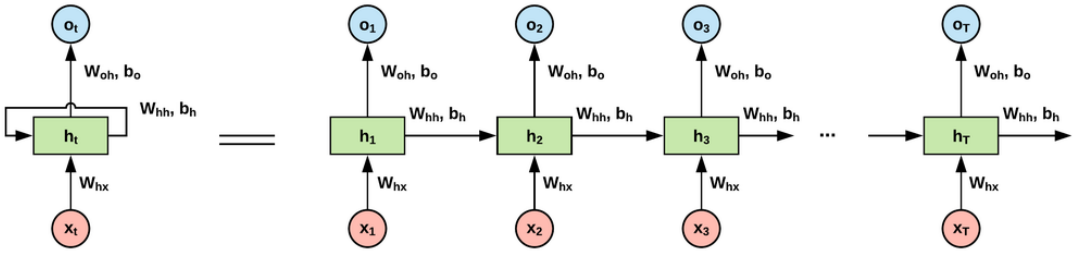

그림 8. RNN의 구조

Input 외에도 이전 hidden state 역시 전달되어 학습이 된다. 과거 정보까지 hidden state에 포함되어 넘겨주므로 시계열 데이터 학습에 유리하다. 본 연구에서는 여러 실험을 통해 layer를 2개, hidden state 크기는 8로 두었으며, 한번 학습하는데 보는 데이터의 길이인 sequence length는 1000이 가장 적합하다 결론내렸다. Control loop가 50Hz로 돌아가므로, 한번 학습할 때 고려하는 이전 데이터의 최대 길이는 1000/50 = 20(초) 이다. 과거 20초동안의 정보를 고려한다는 의미이다. 그 외 학습 관련된 다른 파라미터들은 Learning rate : 1e-3, batch size : 20, epoch : 200 로 정하였다.

일반적인 RNN만으로는 유체의 dynamics가 잘 학습되지 않는다. 우리는 유체의 dynamics를 알고 있으므로, 해당 정보를 이용해보기로 하였다.

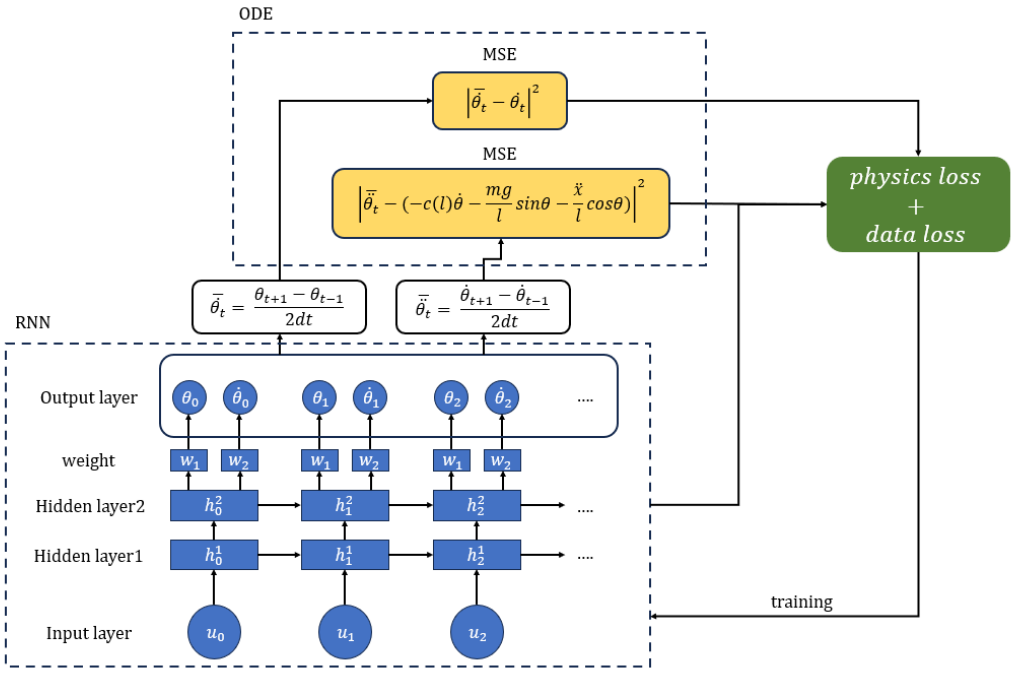

그림 9. Our PIRNN structure

일반적인 data loss 외에 추가적으로 physics loss라는 것을 추가하였다. Sequence length만큼 학습 후, 거기까지 학습이 진행된 네트워크에 dataset의 input u를 넣어 $\theta, \dot{\theta}$을 다시 뽑아낸다. 얻어낸 $\theta, \dot{\theta}$, 그리고 input u 정보를 통해 dynamics 식에 대입하면 $\ddot{\theta}$ 역시 구할 수 있다. 위 그림에서 $\bar{\theta}, \bar{\ddot{\theta}}$ 를 구한 방식으로 각각을 구해 계산으로 구한 $\ddot{\theta}$, 그리고 network의 output으로 나온 $\dot{\theta}$를 MSE로 계산해 loss로 만든다.

D. LQR Control and hardware deploy

i. LQR control

Pendulum model을 기반으로 PIRNN을 학습했기에, 이를 제어하는 LQR 컨트롤러 역시 같은 pendulum model을 기반으로 설계하면 된다. PIRNN network 대신 센서를 통해 state를 측정해 쓸 경우에도 같은 LQR 제어기가 쓰인다. 즉 PIRNN이 sensor역할을 대체할 수 있다는 가정 하에 LQR 제어기를 설계한다고 볼 수 있다.

EQ1을 선형화하여 다음과 같이 state equation을 세웠다.

$$\dot{X} = AX + Bu$$
$$Y = CX$$

$$X = \begin{pmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{pmatrix}, A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\frac{g}{l} & -\frac{c}{m} \end{pmatrix}, B = \begin{pmatrix} 0 \\ 1 \\ 0 \\ -\frac{1}{l} \end{pmatrix}, C = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Pair (A, B)가 stabilizable하고 pair (C, A) 가 observable하므로 LQR 제어가 가능하다. Regulate할 cost J는 다음과 같이 나타난다.

$$J = \int_0^\infty (y^T Q y + u^T R u)\, dt$$

위 cost function과 Lyapunov stability를 통해 얻어진 Riccati equation 및 K, u는 아래와 같다.

$$A^T P + PA - PBR^{-1}B^T P + C^T QC = 0,$$
$$K = R^{-1}B^T P, \;\; u = -KX$$

본 연구에서는 Q, R 을 matlab simulation 및 hardware test를 통해 결정하였다. hardware에 최종적으로 deploy된 Q, R은 아래와 같다.

$$Q = \begin{pmatrix} 0.005 & 0 & 0 \\ 0 & 0.005 & 0 \\ 0 & 0 & 0.02 \end{pmatrix}, \quad R = 0.0005$$

ii. Hardware deploy

상술하였듯, 본 연구에서 사용하는 manipulator를 제어하는 방식은 하드웨어의 한계로 인하여 velocity, position 형태의 trajectory input을 로봇팔에 제공해야 한다. 그러나, 식 (EQ1)에서 얻어진 결과가 가속도이므로 이를 구현하기 위해서 적분의 과정이 수행되었다.(그림 2 참조) 그 결과로, 모터에는 매 단위시간마다 정해진 속도 ($\dot{p}_d(t)$)와 위치 ($p_d(t)$)가 입력됨으로써 end effector가 LQR의 예측에 따른 가속도를 구현해 최종적인 유체 시스템의 안정을 유도한다.

(x, y, z) 좌표계에서 요구되는 경로를 모터가 올바르게 추적하기위한 제어는 다음과 같이 기술된다.

$$\begin{bmatrix} \omega_b(t) \\ \dot{p}(t) \end{bmatrix} = \begin{bmatrix} R^T(t)R_d(t) & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \omega_d(t) \\ \dot{p}_d(t) \end{bmatrix} + K_p X_e(t) + K_i \int_0^t X_e(t')dt' \qquad \text{(EQ2)}$$

$$where,\ X_e(t) = \begin{bmatrix} \omega_e(t) \\ p_d(t) - p(t) \end{bmatrix}, [\omega_e] = \log(R^T R_d)$$

이때 구해지는 ω_b, $\dot{p}$ 은 end effector가 주어진 trajectory를 따라가기 위해 형성되는 속도를 translation velocity와 angular velocity로 분리한 것이다. $X_e(t)$는 특정 시간 t 에서 요구되는 $\omega_b(t), p_d(t)$와 실제 end effector에서 구현되고 있는 $\omega(t), p$ 의 오차를 의미한다. 즉, 식 (EQ2)는 요구되는 velocity input에 대하여 형성되는 오차를 기반으로 얻어지는 PI Control이며, 실제 제어에서 사용한 K_p, K_i의 수치는 $0.8, 0.01$로 설정해 두었다.

```
VectorXf w_d = so3ToVec((R_desired.transpose())*R_desired_dot);
VectorXf w_b = ((R_sb.transpose())*R_desired)*w_d;
VectorXf P_desired_dot = (P_desired - before_P_desired)/Duration;
VectorXf Xe(6);
Xe << so3ToVec(MatrixLog3(((R_sb.transpose())*R_desired))), P_desired-P_sb;
Iterm += Xe*Duration;
VectorXf twist_decoupled(6);
twist_decoupled << w_b, P_desired_dot;
twist_decoupled += Kp*Xe + Ki*Iterm;
MatrixXf J_decoupled(6,6);
J_decoupled << Jb.block<3,6>(0,0), R_sb*Jb.block<3,6>(3,0);
MatrixXf J_decoupled_pinv = J_decoupled.completeOrthogonalDecomposition().pseudoInverse();
thetalist_dot = J_decoupled_pinv*twist_decoupled;
```

그림 1. 코드로 구현한 식 (EQ2). 아래 첨자 b는 end-effector의 성분, d는 목표로 하는 속도 성분을 가리킨다.

식 (EQ2)을 구성하는 변수 목록을 정리하면, (1) 현재 모터의 속도와, (2) 위치정보, (3) 목표하는 모터의 속도와, (4) 목표하는 위치정보로 구성된다. 특정 시간 t에서, 유체가 받고 있는 가속도를 통해 (3), (4)가 계산되고, (1)과 (2)의 수치는 모터 내 내장된 encoder를 통해 읽어오게 된다.

마지막으로, velocity를 구현하기 위해 manipulator의 각 actuator가 출력해야 하는 joint space상에서의 angular velocity는 다음과 같은 변환을 거쳐 계산된다.

$$\dot{\theta} = J_b^{\dagger}(\theta) \begin{bmatrix} \omega_b(t) \\ \dot{p}(t) \end{bmatrix} \qquad \text{(EQ3)}$$

식 (EQ3)와 같이 각속도로 변환하고, 모터를 회전시킴으로써 목표하는 움직임을 구현할 수 있었다.

```
// actuate velocity
actuate_motor.motor[0].vel = thetalist_dot(0);
actuate_motor.motor[1].vel = thetalist_dot(1);
actuate_motor.motor[2].vel = thetalist_dot(2);
actuate_motor.motor[3].vel = thetalist_dot(3);
actuate_motor.motor[4].vel = thetalist_dot(4);
actuate_motor.motor[5].vel = thetalist_dot(5);
actuate_motor.setVelocity(groupSyncWrite);
```

그림 3. 식 (EQ3)을 통해 계산된 속도가 모터에 반영되는 코드

최종적으로, LQR을 통해 얻어진 u 를 실시간으로 적분해 velocity trajectory와 position trajectory를 생성한 후 그것을 로봇팔의 end point(물통)가 따라갈 수 있도록 하였다.

3. 연구 결과

A. PIRNN vs RNN

Physics-inform을 사용했을 때와 그렇지 않았을 때 Network의 estimate 성능을 비교한다. 아래 이미지는 model의 test에 사용된 dataset이다. LQR input과 impulse input이 적절히 섞여 있다.

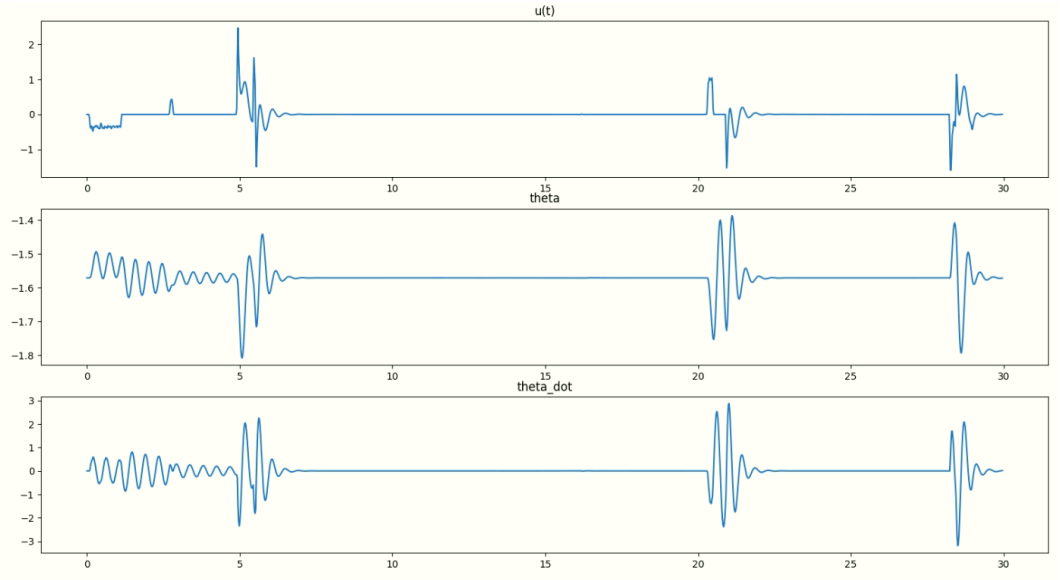

그림 10. Test data

PIRNN과 RNN은 loss 구조 외에 모든 것을 같게 하여 training 하였다. 위 샘플에 대한 test 결과이다. 점선이 groundtruth, 파랑,빨강 실선이 model의 output, 초록색 실선은 model의 output에 차단주파수가 3Hz인 LPF filter가 통과된 값이다. 실제 하드웨어에 적용될 때에는 LPF을 거쳐 입력된다.

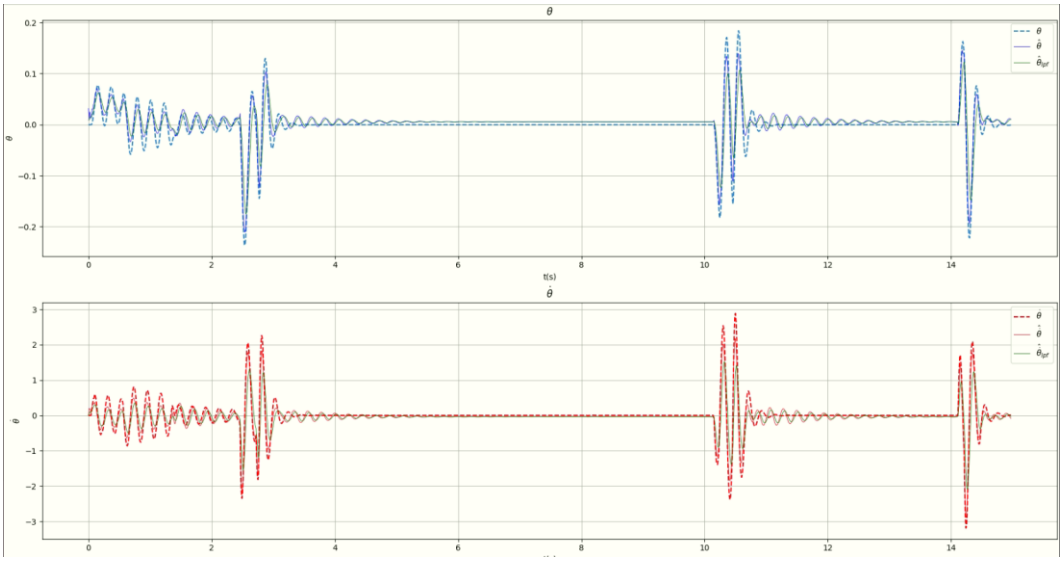

그림 11. PIRNN

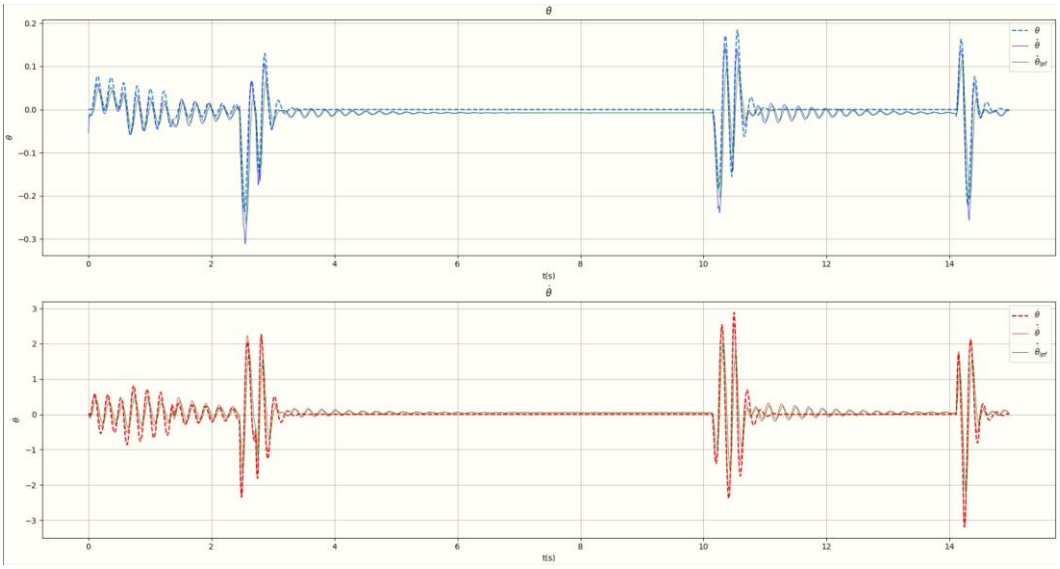

그림 12. RNN

위 두 그래프에서 확인할 수 있듯, train data의 발생원리와 같은 방식 (Impulse+LQR) 으로 만들어진 test data에서는 PIRNN과 RNN의 성능에 큰 차이가 없다. 다음은 train data를 만든 방법과 다르게 만들어진 test data에 대한 PIRNN과 RNN의 반응이다. 이번 test data는 아래와 같다.

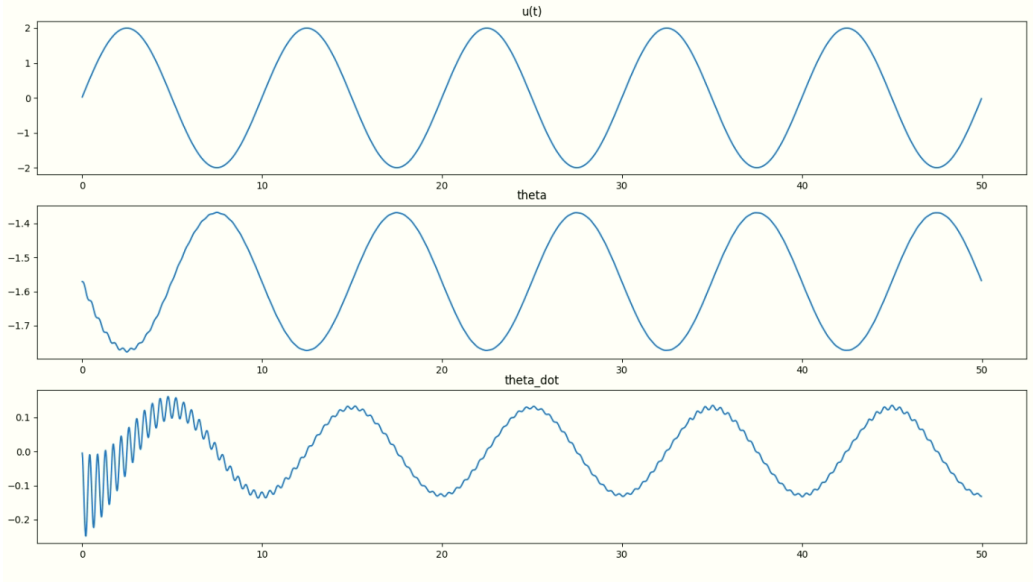

그림 13. test data

유체에 힘을 impulse형태로 가하지 않고, 진동수가 0.1Hz인 sin파 형태로 힘을 가하도록 하였다. 실제 세계에서 로봇에 임펄스 형태의 힘만 가해진다는 보장은 없기 때문에, 다른 형태의 힘에 대한 estimation의 정확도 역시 중요하다.

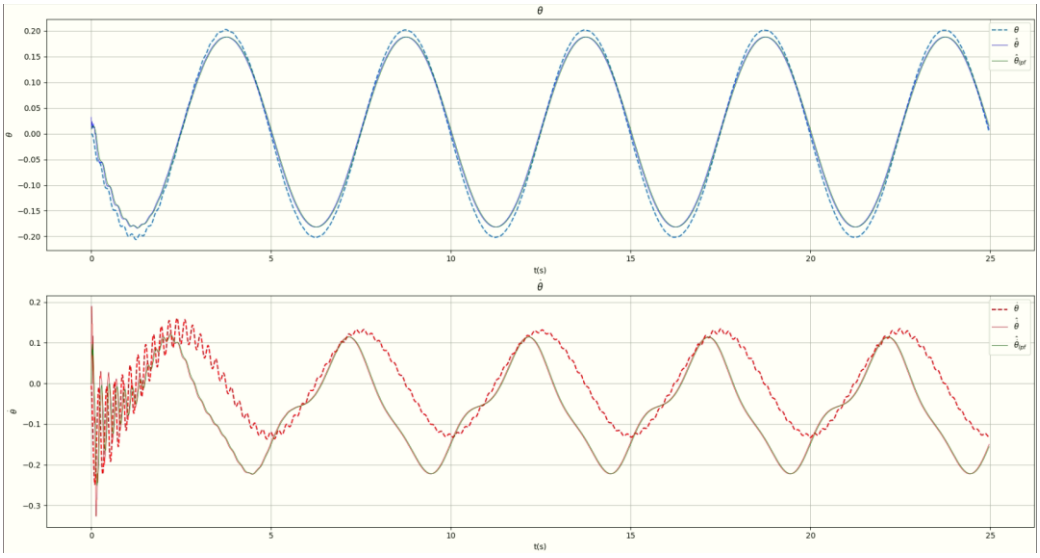

그림 14. PIRNN

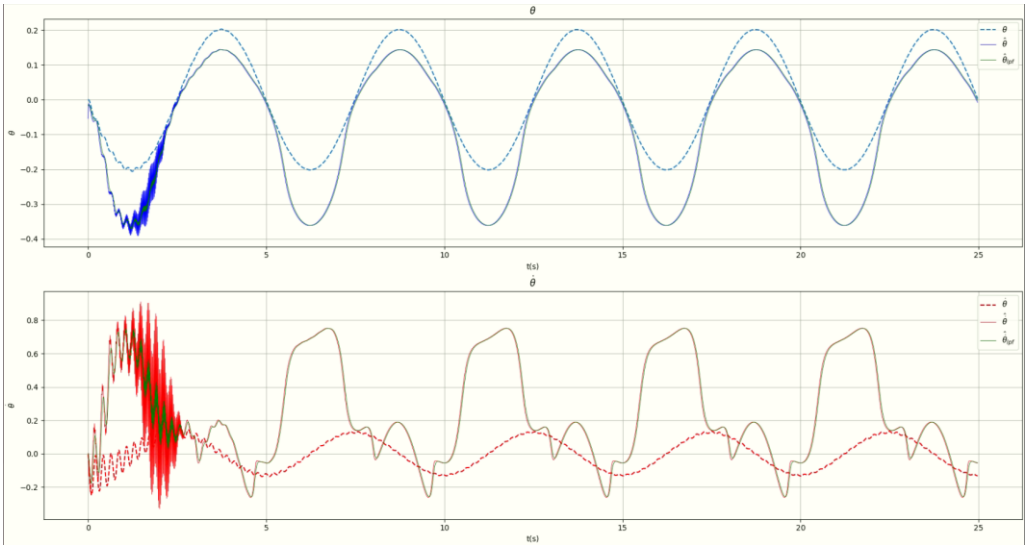

그림 15. RNN

위 그래프에서 확인할 수 있듯, sin형태의 input에 대한 정확도 차이는 굉장히 크게 나타났다. PIRNN의 경우 학습 시 θ와 $\dot{\theta}$의 관계, 그리고 모델 dynamics와의 관계까지 고려해 학습한 반면, 일반 RNN은 train dataset과 얼마나 비슷하게 따라가는지에 대해서만 고려되었다. 그러다 보니, train set에서 보지 못한 형태의 input이 들어오게 되면 PIRNN에 비해 RNN이 훨씬 낮은 성능을 보이게 되었다.

본 연구에서 설계한 PIRNN의 성능이 훨씬 우수하다고 할 수 있다.

실제 하드웨어에 적용할 때에는 train dataset이 만들어진 방식인 impulse u + LQR u 에서 LQR u의 비중을 위 모델보다 좀 더 높인 PIRNN 모델을 사용하였다. 하드웨어의 낮은 응답속도, 부정확성 등에 의해 큰 진동을 조금 못 따라가더라도 수렴성을 잘 더 잘 따라가는 모델을 사용하였다. 첫번째 test data에 대한 응답은 다음과 같다.

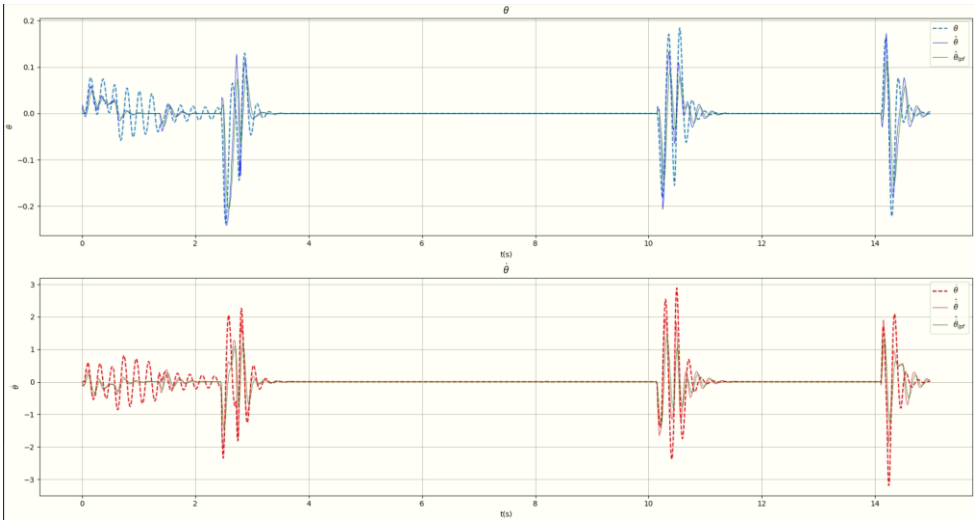

그림 16. Hardware에 적용된 model의 test결과

B. LQR

Matlab을 통해 LQR모델을 실험하였다. 아래 그래프는 실제 하드웨어에 적용할 때 사용한 계수로 시뮬레이션상에서 LQR을 진행했을 때의 결과이다. 초기 θ를 30도로 두었을 때의 제어 결과이다.

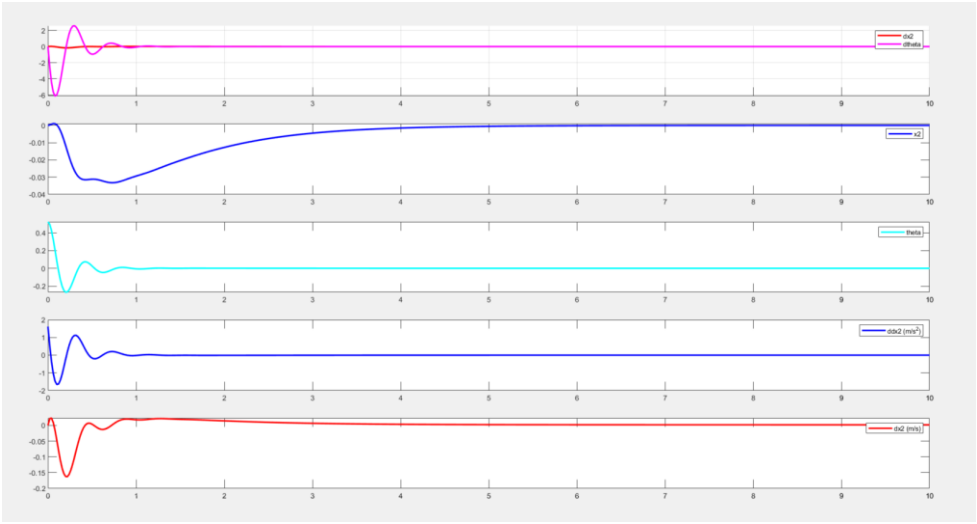

그림 17. LQR result (hardware apply)

위에서부터 $\dot{\theta}(pink)$, $x, \theta, \ddot{x}, \dot{x}$이다. 가장 위 그래프에는 $\dot{\theta}$말고도 $\dot{x}$이 그려져 있는데, 축척 차이 때문에 잘 안 보여서 마지막에 한번 더 넣었다. LQR계수가 아래와 같았는데,

$$Q = \begin{pmatrix} 0.005 & 0 & 0 \\ 0 & 0.005 & 0 \\ 0 & 0 & 0.02 \end{pmatrix}, \quad R = 0.0005$$

Q11에 대응되는 값이 x, Q22에 대응되는 값이 $\dot{x}$, Q33은 θ, R은 $\ddot{x}$에 대응된다. 값이 클수록 수렴이 빠르게 되는 경향을 보인다. 만약 계수를 아래와 같이 바꾸게 되면,

$$Q = \begin{pmatrix} 0.05 & 0 & 0 \\ 0 & 0.05 & 0 \\ 0 & 0 & 0.2 \end{pmatrix}, \quad R = 0.0005$$

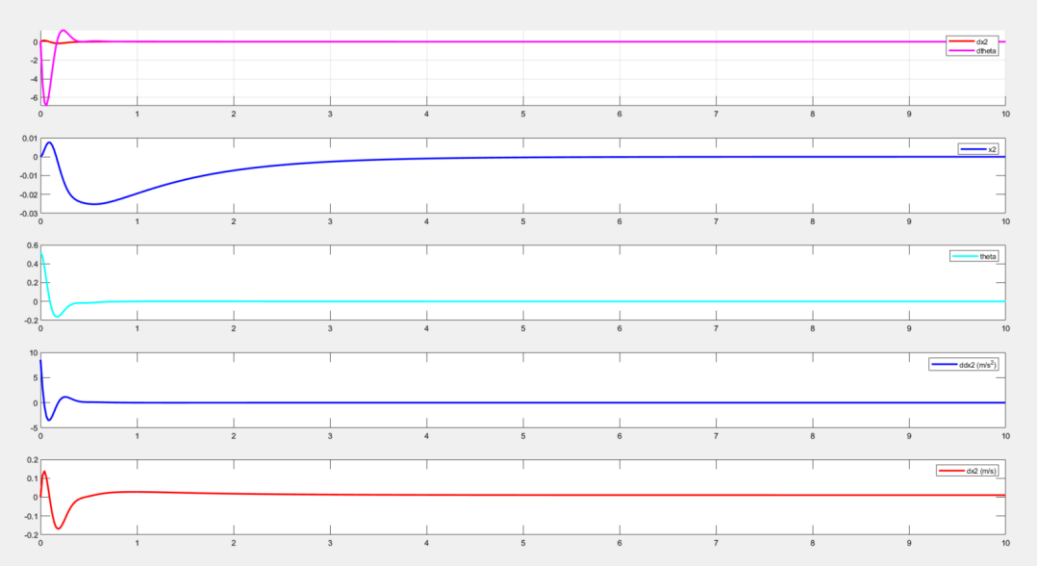

그림 18. LQR result 2

이렇게 더 빠른 수렴속도를 보임을 확인할 수 있다. 아래 링크는 LQR에서 얻어낸 velocity trajectory를 로봇팔이 그대로 따라가도록 한 영상이다.

https://youtu.be/iFXcP1niz1c?si=wchmqjdsmRDMK9jh

https://youtu.be/3L_tE4xatJM?si=PNG8lcJSo90ZFYpZ

C. Real world test (PIRNN + LQR)

실제 하드웨어에 LQR 및 PIRNN network를 적용하기 전에, 몇 가지 고려해야 할 사항이 존재한다. 첫번째로, 본 연구에 사용된 로봇팔의 제어 주기가 50Hz밖에 되지 않는다는 점이다. 그보다 높일 시 통신이 불안정해져 로봇팔이 동작중 정지하거나 크게 흔들리는 현상이 발생한다. LQR제어는 continuous time input이 가능하다 가정하고 푼 식이기 때문에, Hz가 낮아질수록 그 간극이 커진다. 계수를 결정함에 있어서 보수적으로 진행해야 한다. PIRNN 역시 이를 의식하여, 수렴성을 좀 더 보장한 모델을 사용하였고, LPF 역시 약하게 걸어주었다. 두 번째로, 로봇팔이 torque control이 되지 않는다는 점이다. 모터 자체 성능의 한계 때문에, velocity 및 position input만 가능하다. 이런 상황에서 가능한 velocity tracking 방식을 사용했기 때문에, LQR이 내놓는 output을 한번 적분(v), 두번 적분(p) 하여 그 둘을 합쳐 사용한다. 이 과정에서 data 손실이 있을 수 있으며 특히 가속도에 대한 즉각적인 반응성이 조금 떨어질 수 있다. 마지막으로, 로봇팔을 구성하는 모터 각각이 낼수 있는 최대 속도가 제한되어 있다. 이를 고려해서 너무 빠른 input trajectory를 만들어내지 않도록 제어기를 설계해야한다. 이러한 사항들을 고려하였기에, B에서 더 빠른 수렴성을 가지는 계수가 있음에도 낮은 계수를 쓰게 되었다.

Libtorch를 이용해 파이썬에서 training한 model을 c++로 옮겨 로봇 코드에 적용하였고, LQR 계수 역시 적용하였다. 실험 방식은 로봇을 앞 뒤로 두번씩 흔들었을 때 물 높이가 어떻게 변화하는지를 측정하도록 하였다. 로봇팔이 가만히 있을 때, PIRNN + LQR제어기가 켜져 있을 때로 나누어 실험을 진행하였다.

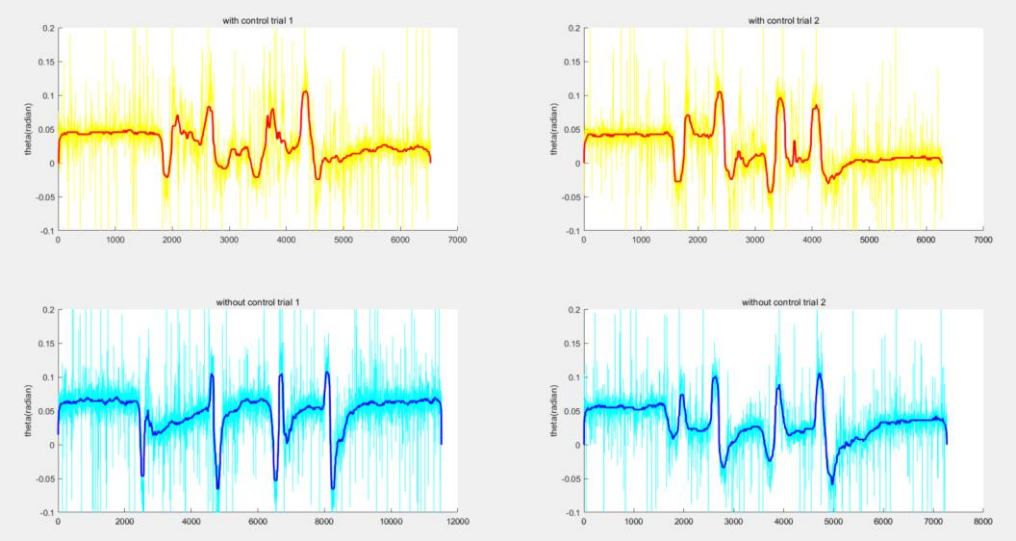

그림 19. Real test data. 위쪽 2개가 with control, 아래쪽 2개가 without control
노랑색, 하늘색이 sensor의 raw data, 빨간색과 파란색이 raw data에 중앙값 필터 적용한 값

크게 차이가 나진 않지만, 눈에 띌 정도의 차이점이 존재한다. 전반적으로 control을 적용했을 때 진동폭이 적어지는 것을 확인할 수 있으며, 카트를 밀었을 때 발생하는 가속도와 멈췄을 때 반대로 발생하는 가속도에 의한 유체의 움직임을 효과적으로 감소시켰다.

아래는 control이 적용되었을 때 로봇팔의 움직임을 담은 영상이다.

https://youtu.be/WXFvpuvKKA8?si=HQklh5dtNkfUqKTe

4. Future work 및 활용방안

다른 연구들과 달리, 본 연구에서는 유체의 sloshing을 센서 없이 제어하기 때문에, 서빙로봇과 같은 실제상황에 바로 적용이 가능하다. 또한, 카트의 이동으로 제어를 하는 다른 연구들과 다르게 로봇팔로 제어가 되므로 서빙로봇과 같은 카트와 별도의 연결 없이 그냥 위에 올려두기만 해도 유체 안정화 제어가 바로 가능하다.

로봇팔로 sloshing control을 했을 때 또다른 장점은 x, y, z 방향 그리고 rotation 방향으로 가해지는 가속도에게까지 빠르게 대응이 가능하다는 점이다. 본 연구에서는 x방향에 대해서만 제어를 진행했지만 확장이 가능하다. 또한, 로봇팔 이기에 sloshing control을 진행하면서 로봇팔의 끝점을 원하는 위치로 이동시키는 것 또한 가능하다. 예시로 서빙 로봇 위 로봇팔이 물을 따르는 trajectory, 컵을 옮기는 동작을 수행하면서도 동시에 sloshing control을 수행하는 것이 있겠다.

따라서, future work로는 우선 x축 뿐 아니라 다른 축 방향까지 동시에 고려해 제어하는 것을 구현해보려 한다. 또한 LQI control과 같은 기법을 이용해 로봇팔의 끝점이 trajectory tracking을 하면서도 sloshing control까지 진행할 수 있도록 발전시킬 계획이다.

또한, 현재는 sloshing control을 할 때 앞뒤로 움직여 추가적인 가속도를 만들어내 물이 넘치지 않도록 하는데, 로봇팔의 경우에는 컵을 기울이면서 넘치지 않게 하는 것 역시 가능하다. 이렇게 rotation움직임까지 사용해 제어를 한다면 훨씬 빠르고 안정적으로 물이 넘치지 않을 수 있을 것이다. 이러한 점들을 고려하여 후속연구를 진행해보고자 한다.

이번에 수행한 연구에서 아쉬운 점은, 실험 시 힘을 일정하게 가해주지 못했다는 점이다. 사람이 일정하게 흔들었다고는 하지만, 그 힘의 크기를 완전히 똑같게 유지하긴 힘들다. 따라서 후속 연구에서는 일정하게 외력을

줄 수 있는 장치도 개발할 계획이다.

참고 문헌

GuagliumiBerti, A., Monti, E., & Carricato, ML.,. (2021). A simple model-based method for sloshing estimation in liquid transfer in automatic machines. " IEEE Access."

HAMAGUCHI& YAJIMA, T.M.,. (2023). Vibration control for sloshing in liquid container in cart with active vibration reducer (Transfer on an uneven road). " Mechanical Engineering Journal, 10(4)."

YanoI., Yoshida, T., Hamaguchi, M., & Terashima, KK. (1996). Liquid container transfer considering the suppression of sloshing for the change of liquid level. " IFAC Proceedings Volumes", (페이지: 29(1), 701-706.).

Zheng& Wu, ZY.,. (2023). Physics-informed online machine learning and predictive control of nonlinear processes with parameter uncertainty. "Industrial & Engineering Chemistry Research", (페이지: 2804-2818.).

ZhengHu, C., Wang, X., & Wu, Z.Y.,. (2023). Physics-informed recurrent neural network modeling for predictive control of nonlinear processes. "Journal of Process Control."

학부생연구프로그램(UGRP)
개인정보 및 결과물 활용 동의서

개인정보 수집·이용 동의

본인은 학부생연구프로그램(UGRP) 결과물에 포함된 개인정보를 교육혁신센터가 '학부생연구프로그램(UGRP)'의 교육적 운영을 위해 수집 및 이용하는 것에 동의합니다.

1. 개인정보 수집·이용 목적: 프로그램 운영, 관련 보고문서 작성, 비영리 교육홍보자료 이용, 성과 공유 등
2. 개인정보 수집 항목: 성명, 학년, 학과, 학번, 연락처, 메일주소
3. 개인정보 보유 및 이용기간: 개인정보 수집 및 이용 목적의 종료 시까지

▣ 동의함 □ 동의하지 않음

결과물 활용 동의

본인은 학부생연구프로그램(UGRP)에 제출한 결과물을 교육혁신센터가 교육적 목적을 위해 관련한 업무·행사에서 활용할 권리는 갖는 것에 동의합니다.

* 결과물 활용 예시: 교·내외 홍보 및 성과 공유, 프로그램 개선을 위한 활용, 관련 보고·교육자료 자료 작성, 프로그램 운영

▣동의함 □ 동의하지 않음

본인과 우리팀은 위의 모든 내용을 충분히 확인하였으며, 이에 서명합니다.

2024년 1월 9일

팀장: 이도현 (이도현)

팀원: 김종열 (김종열)

팀원: 신민경 (신민경)

포 항 공 과 대 학 교 귀하