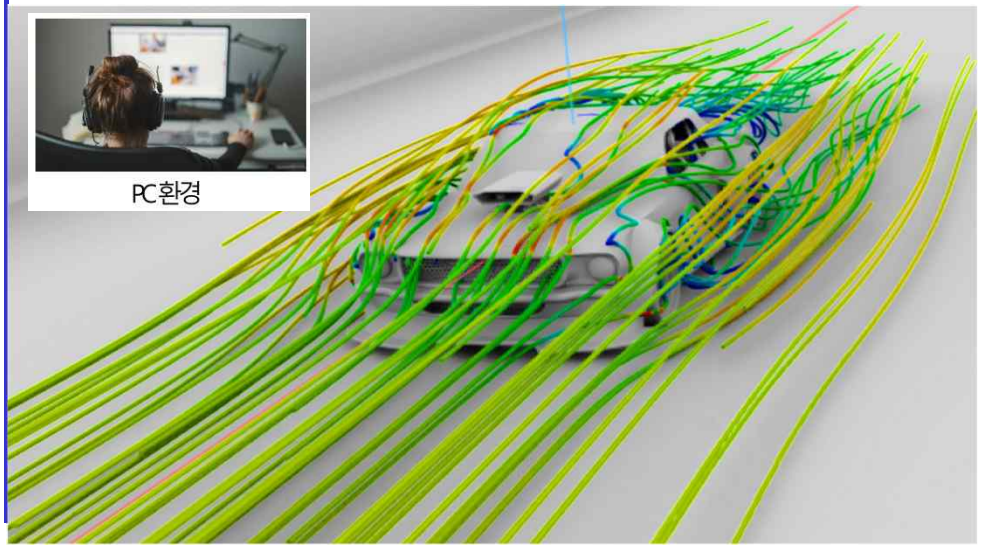
CFD를 위한 Data·AI 공개 라이브러리 활용
: Airfoil 공기역학 Case

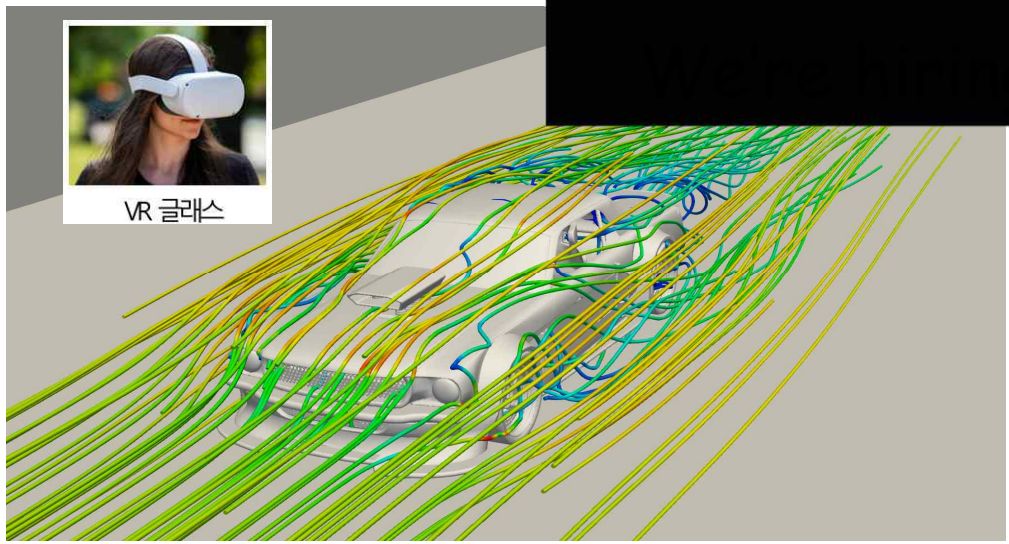
신 정 훈

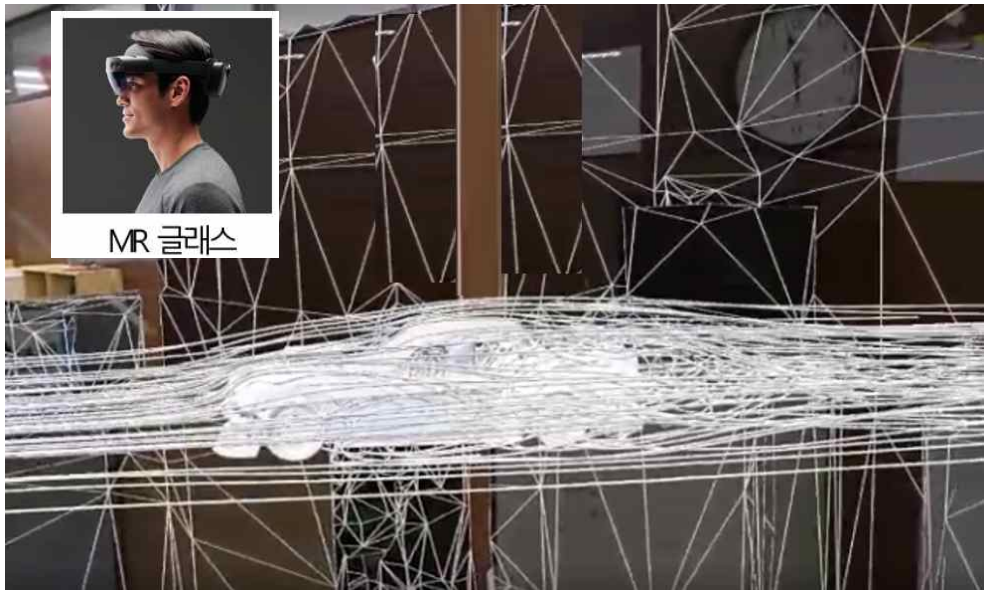
KISTI, Open XR 플랫폼 융합연구단

참고 Code Set Repository -
https://github.com/Jameshin/AI_Airfoil_Aerodynamics
참고 Data Set Repository -
https://dataon.kisti.re.kr/search/view.do?mode=view&svcId=f3face48acf03e666ba0d44394438ed8

부서업무 소개 – CFD for Extended Reality (XR)



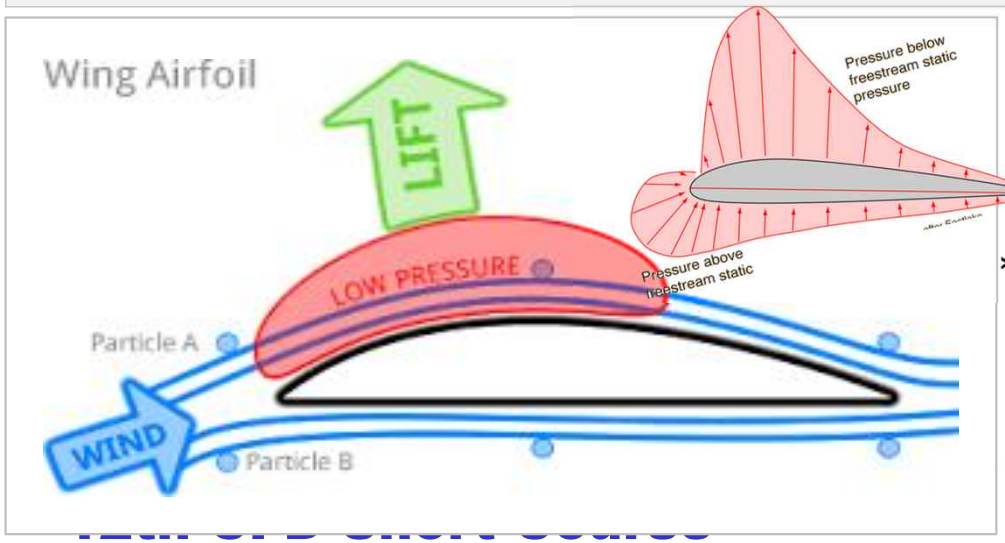


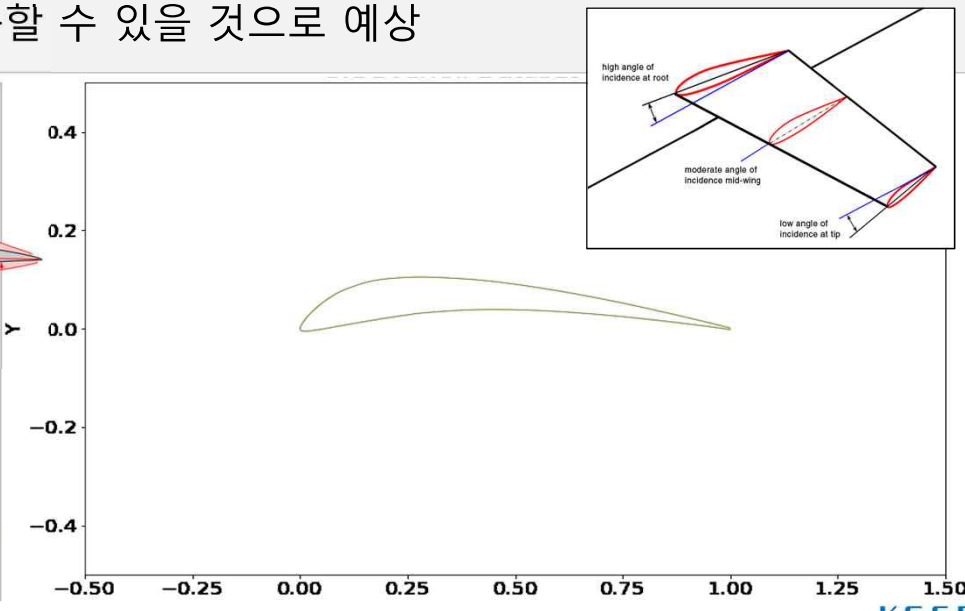


목차

문제 : **Airfoil** 공기역학

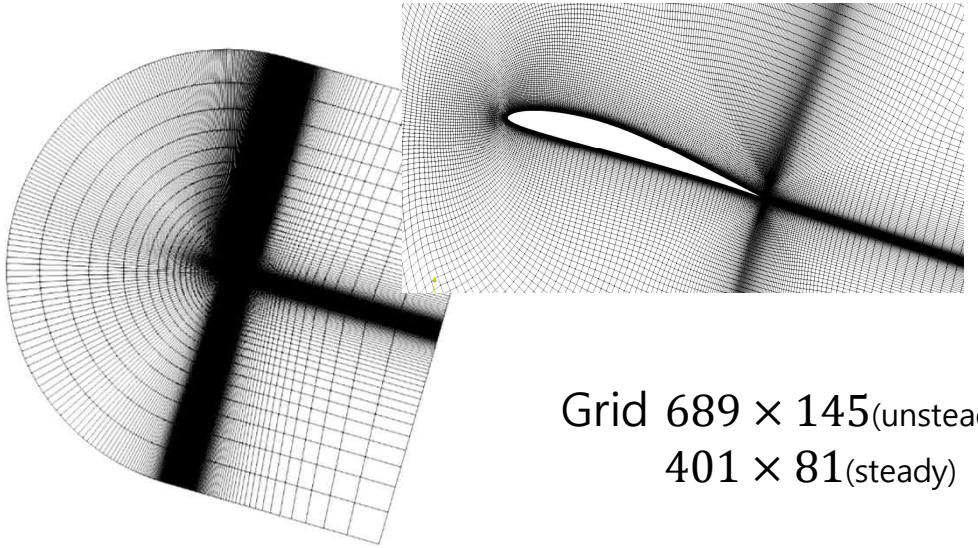
- **[Simulatable]** Data 및 시뮬레이션 SW 확보 수월 (예. UIUC airfoil dataset, In-house SW)
- **[Validatable]** 현재 아키텍처/규모/데이터 수준에서 예측모형 검증 Study 즉시 수행 가능
- **[Challenging]** 경계층, 박리, 층류, 난류, 천이, 충격파 등의 다양한 물리현상 포함하고, 미소한 형상 변화에도 큰 성능(Lift/Drag)의 차이를 나타냄 → 실시간 예측모형 엄밀 검증
- **[Widely-Applied]** 다양한 산업군에 날개 제품 적용 → 2차원 날개는 풍동실험으로 수행이 어렵고, 제품 초기설계 단계의 성능예측이 필요하므로 익형 시뮬레이션 수행이 필수적
- **[Understandable]** 문제 Set-up과 이해 용이함 → 주어진 조건(형상/유동)에서의 성능 확인
- **[Extensible]** 2차원 해석결과가 3차원 날개의 성능에 지배적 영향력을 가짐 → 향후 3차원 문제로 확장할 때 2차원 실시간 모델 및 데이터를 활용할 수 있을 것으로 예상



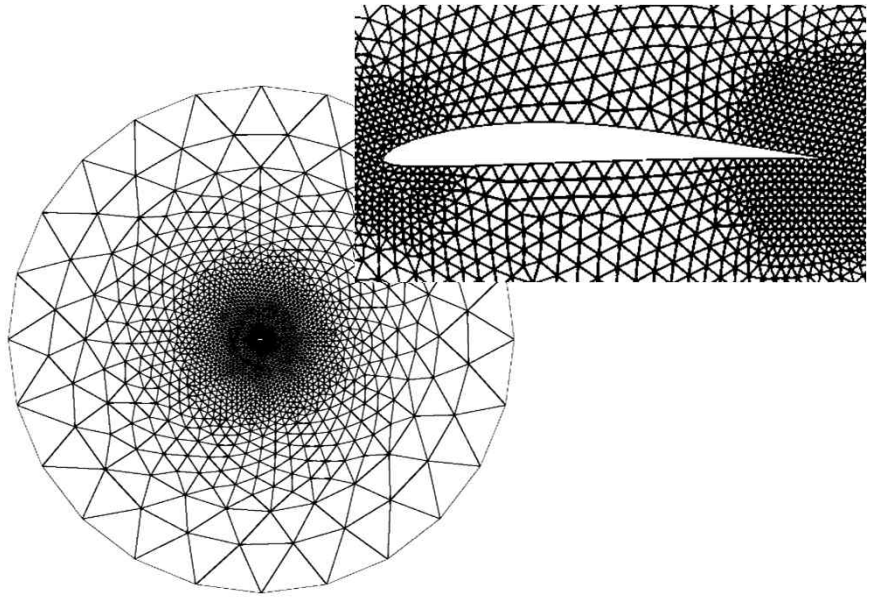


KSCFE

Sim. Data & 학습/추론 기법



Grid 689×145(unsteady)
401×81(steady)



항목	Case 1	Case 2	Case 3
해석 대상 및 조건	Eppler387,정렬격자, Unsteady, 익형 받음각 16 deg, Mach 0.6, Re $1x10^4$, RANS/SA, Roe flux	NACA0012, 비정렬격자, Unsteady, Laminar, Reynolds 계수 $1x10^4$, Mach 0.4, 받음각 15 deg	UIUC airfoil 1550종, Reynolds 계수 $1x10^5$, Mach 0.5, 받음각 1 deg, RANS/SA, AUSM+
Dataset	Tecplot format 학습 – 1, 5, 9, 13, …, 77 (21 Time steps with interval 4) 예측 – 3, 7, 11, …, 75 (20 Time steps with interval 4)		Tecplot format 학습 – 1530개 형상 (10% validation) 예측 – 20개 형상
학습/추론 기법	POD(Local linear), POD(NN), FCDNN	POD(Local linear), POD(NN), FCDNN	CNN (U-net)

Data Format - Tecplot

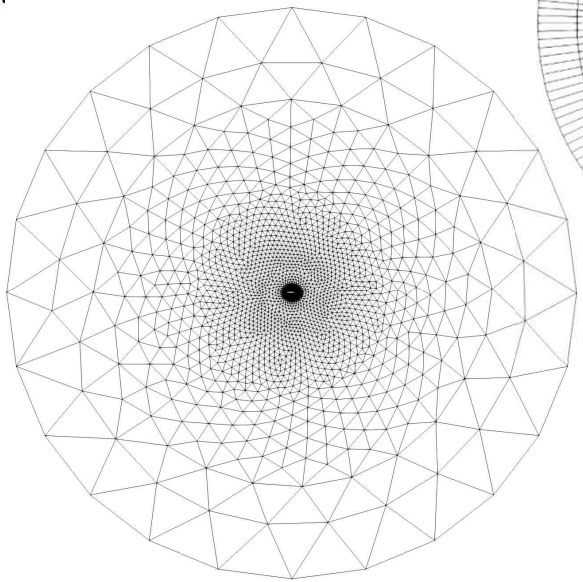
flo001.OOOuns
mid_result_OOO.rlt
flo001.OOO.dat

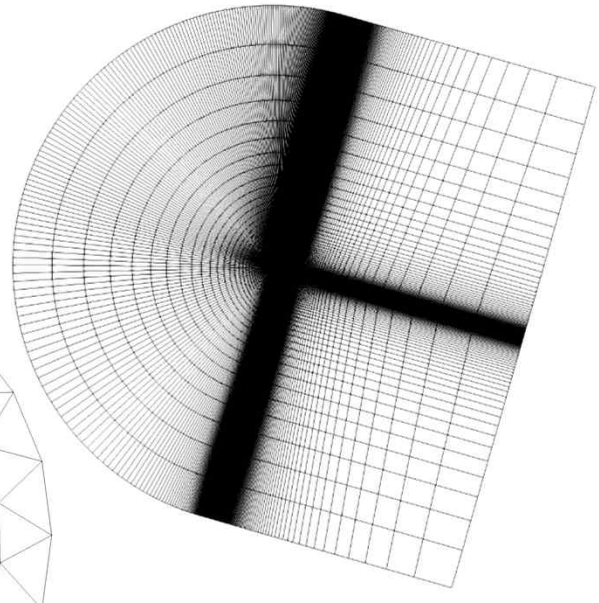
Structured Grid

```
variables = "X","Y","Z","rh","u","v","w","p","T","Vor","Q_cri"
 zone i=         401  j=          81
 0.50500000E+02  0.00000000E+00  0.10000000E+01  0.10001481E+01  0.10017640E+01  0.15663552E-01  0.00000000E+00 ...
 0.45277153E+02  0.12948362E-04  0.10000000E+01  0.99997457E+00  0.10026291E+01  0.15541468E-01  0.00000000E+00
 0.40605200E+02  0.24530963E-04  0.10000000E+01  0.10000083E+01  0.10030670E+01  0.15363839E-01  0.00000000E+00
                                                 ⋮
```

Unstructured Grid

```
variables ="x","y","rho","u","v","p","m"
zone t=" 0.358614E-01",n=        5143 ,e=       10134
,varlocation=([3,4,5,6,7]=cellcentered),zonetype=fetriangle
datapacking=block
  1.00899934800000
  1.00654161000000
  1.00383460500000
  1.00085294200000
 0.997568726500000
 0.993951082200000
                  ⋮

     1          2          185
     185        2          186
     2          3          186
     3          4          186
     186        4          187
     4          5          187
```





분석/개발 환경 구축

Anaconda - https://www.anaconda.com

ANACONDA

jupyter

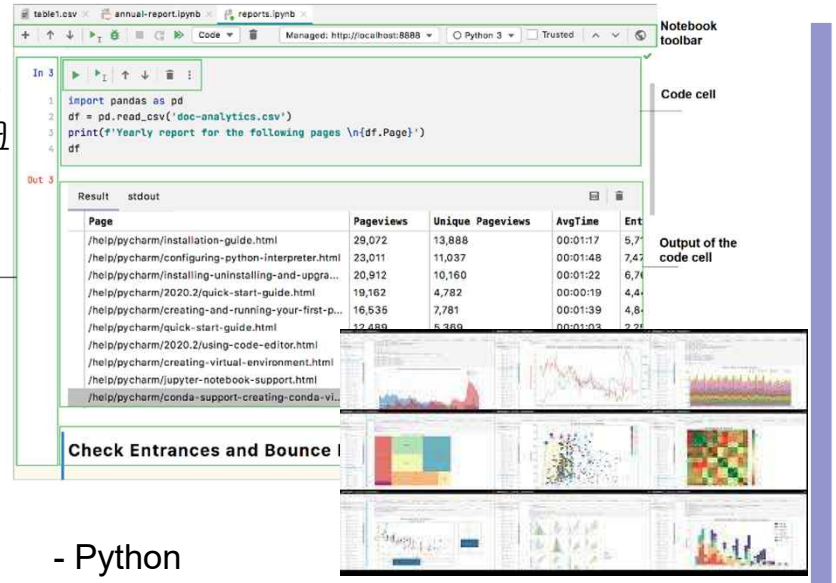
추가 패키지/프레임워크

pip install OOO 혹은

conda install OOO

환경 Preset 제공
: Python + 수치/물리 패키지
(Numpy/Scipy/Pandas)

Interactive 코딩 UI 제공
: 웹브라우저, Visualization



Tensorflow - https://www.tensorflow.com

In Conda Prompt,
pip install tensorflow

NVIDIA CUDA cuDNN

최대 머신러닝/데이터사이언스 프레임워크
: 참고 코드 多, Tensorflow 1.x & Tensorflow 2.x 구분 필요,
GPU 구동을 위해 NVIDIA CUDA, cuDNN 설치(버전 확인)

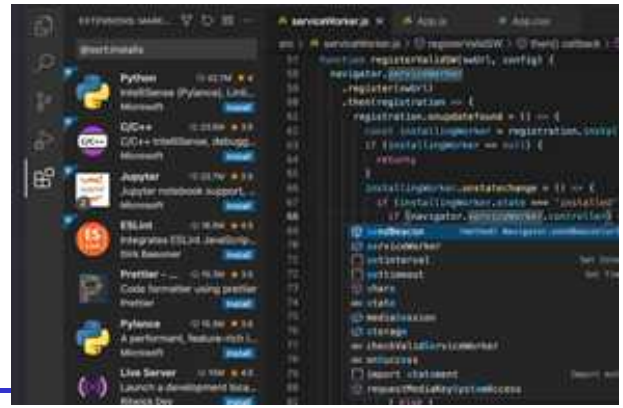
- Python
  Import tensorflow

- C, C++
  #include <tensorflow/c/c_api.h>

Extension / Interpreter

Visual Studio Code - https://www.anaconda.com

가벼운 Visual Studio, 데이터 과학 기능 제공
: 윈도우/맥OS/리눅스, 다양한 Extension/Interpreter



datacamp

Python For Data Science
NumPy Cheat Sheet

Learn NumPy online at www.DataCamp.com

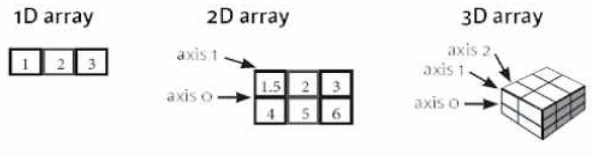
Numpy

The NumPy library is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays

Use the following import convention:

```
>>> import numpy as np
```

NumPy Arrays



Creating Arrays

```
>>> a = np.array([1,2,3])
>>> b = np.array([(1.5,2,3), (4,5,6)], dtype = float)
>>> c = np.array([[(1.5,2,3), (4,5,6)],[(3,2,1), (4,5,6)]], dtype = float)
```

Initial Placeholders

```
>>> np.zeros((3,4)) #Create an array of zeros
>>> np.ones((2,3,4),dtype=np.int16) #Create an array of ones
>>> d = np.arange(10,25,5) #Create an array of evenly spaced values (step value)
>>> np.linspace(0,2,9) #Create an array of evenly spaced values (number of samples)
>>> e = np.full((2,2),7) #Create a constant array
>>> f = np.eye(2) #Create a 2X2 identity matrix
>>> np.random.random((2,2)) #Create an array with random values
>>> np.empty((3,2)) #Create an empty array
```

I/O

Saving & Loading On Disk

```
>>> np.save('my_array', a)
>>> np.savez('array.npz', a, b)
>>> np.load('my_array.npy')
```

Saving & Loading Text Files

```
>>> np.loadtxt("myfile.txt")
>>> np.genfromtxt("my_file.csv", delimiter=',')
>>> np.savetxt("myarray.txt", a, delimiter=" ")
```

Asking For Help

```
>>> np.info(np.ndarray.dtype)
```

Inspecting Your Array

```
>>> a.shape #Array dimensions
>>> len(a) #Length of array
>>> b.ndim #Number of array dimensions
>>> e.size #Number of array elements
>>> b.dtype #Data type of array elements
>>> b.dtype.name #Name of data type
>>> b.astype(int) #Convert an array to a different type
```

Data Types

```
>>> np.int64 #Signed 64-bit integer types
>>> np.float32 #Standard double-precision floating point
>>> np.complex #Complex numbers represented by 128 floats
>>> np.bool #Boolean type storing TRUE and FALSE values
>>> np.object #Python object type
>>> np.string_ #Fixed-length string type
>>> np.unicode_ #Fixed-length unicode type
```

Array Mathematics

Arithmetic Operations

```
>>> g = a - b #Subtraction
 array([[-0.5, 0. , 0. ],
        [-3. , -3. , -3. ]])
>>> np.subtract(a,b) #Subtraction
>>> b + a #Addition
 array([[ 2.5, 4. , 6. ],
        [ 5. , 7. , 9. ]])
>>> np.add(b,a) #Addition
>>> a / b #Division
 array([[ 0.66666667, 1. , 1. ],
        [ 0.25 , 0.4 , 0.5 ]])
>>> np.divide(a,b) #Division
>>> a * b #Multiplication
 array([[ 1.5, 4. , 9. ],
        [ 4. , 10. , 18. ]])
>>> np.multiply(a,b) #Multiplication
>>> np.exp(b) #Exponentiation
>>> np.sqrt(b) #Square root
>>> np.sin(a) #Print sines of an array
>>> np.cos(b) #Element-wise cosine
>>> np.log(a) #Element-wise natural logarithm
>>> e.dot(f) #Dot product
 array([[ 7., 7.],
        [ 7., 7.]])
```

Numpy is an efficient linear algebra library
Conda에 포함

Comparison

```
>>> a == b #Element-wise comparison
 array([[False, True, True],
        [False, False, False]], dtype=bool)
>>> a < 2 #Element-wise comparison
 array([True, False, False], dtype=bool)
>>> np.array_equal(a, b) #Array-wise comparison
```

Aggregate Functions

```
>>> a.sum() #Array-wise sum
>>> a.min() #Array-wise minimum value
>>> b.max(axis=0) #Maximum value of an array row
>>> b.cumsum(axis=1) #Cumulative sum of the elements
>>> a.mean() #Mean
>>> np.median(b) #Median
>>> np.corrcoef(a) #Correlation coefficient
>>> np.std(b) #Standard deviation
```

Copying Arrays

```
>>> h = a.view() #Create a view of the array with the same data
>>> np.copy(a) #Create a copy of the array
>>> h = a.copy() #Create a deep copy of the array
```

Sorting Arrays

```
>>> a.sort() #Sort an array
>>> c.sort(axis=0) #Sort the elements of an array's axis
```

Subsetting, Slicing, Indexing

Subsetting

```
>>> a[2] #Select the element at the 2nd index
 3
>>> b[1,2] #Select the element at row 1 column 2 (equivalent to b[1][2])
 6.0
```

Slicing

```
>>> a[0:2] #Select items at index 0 and 1
 array([1, 2])
>>> b[0:2,1] #Select items at rows 0 and 1 in column 1
 array([ 2., 5.])
>>> b[:1] #Select all items at row 0 (equivalent to b[0:1, :])
 array([[1.5, 2., 3.]])
>>> c[1,...] #Same as [1,:,:]
 array([[[ 3., 2., 1.],
         [ 4., 5., 6.]]])
>>> a[ : :-1] #Reversed array a array([3, 2, 1])
```

Boolean Indexing

```
>>> a[a<2] #Select elements from a less than 2
 array([1])
```

Fancy Indexing

```
>>> b[[1, 0, 1, 0],[0, 1, 2, 0]] #Select elements (1,0),(0,1),(1,2) and (0,0)
 array([ 4. , 2. , 6. , 1.5])
>>> b[[1, 0, 1, 0]][:,[0,1,2,0]] #Select a subset of the matrix's rows and columns
 array([[ 4. ,5. , 6. , 4. ],
        [ 1.5, 2. , 3. , 1.5],
        [ 4. , 5. , 6. , 4. ],
        [ 1.5, 2. , 3. , 1.5]])
```

Array Manipulation

Transposing Array

```
>>> i = np.transpose(b) #Permute array dimensions
>>> i.T #Permute array dimensions
```

Changing Array Shape

```
>>> b.ravel() #Flatten the array
>>> g.reshape(3,-2) #Reshape, but don't change data
```

Adding/Removing Elements

```
>>> h.resize((2,6)) #Return a new array with shape (2,6)
>>> np.append(h,g) #Append items to an array
>>> np.insert(a, 1, 5) #Insert items in an array
>>> np.delete(a,[1]) #Delete items from an array
```

Combining Arrays

```
>>> np.concatenate((a,d),axis=0) #Concatenate arrays
 array([ 1, 2, 3, 10, 15, 20])
>>> np.vstack((a,b)) #Stack arrays vertically (row-wise)
 array([[ 1. , 2. , 3. ],
        [ 1.5, 2. , 3. ],
        [ 4. , 5. , 6. ]])
>>> np.r_[e,f] #Stack arrays vertically (row-wise)
>>> np.hstack((e,f)) #Stack arrays horizontally (column-wise)
 array([[ 7., 7., 1., 0.],
        [ 7., 7., 0., 1.]])
>>> np.column_stack((a,d)) #Create stacked column-wise arrays
 array([[ 1, 10],
        [ 2, 15],
        [ 3, 20]])
>>> np.c_[a,d] #Create stacked column-wise arrays
```

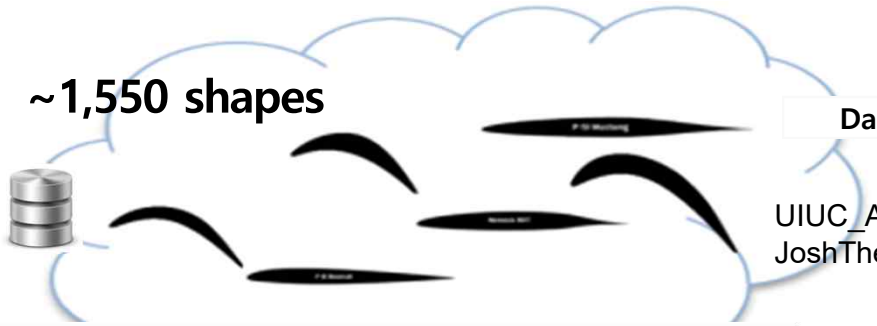
Splitting Arrays

```
>>> np.hsplit(a,3) #Split the array horizontally at the 3rd index
 [array([1]),array([2]),array([3])]
>>> np.vsplit(c,2) #Split the array vertically at the 2nd index
 [array([[[ 1.5, 2. , 1. ],
          [ 4. , 5. , 6. ]]]),
 array([[[ 3., 2., 3.],
         [ 4., 5., 6.]]])]
```

[1단계] UIUC Airfoil 형상 기반 CFD Dataset 구축

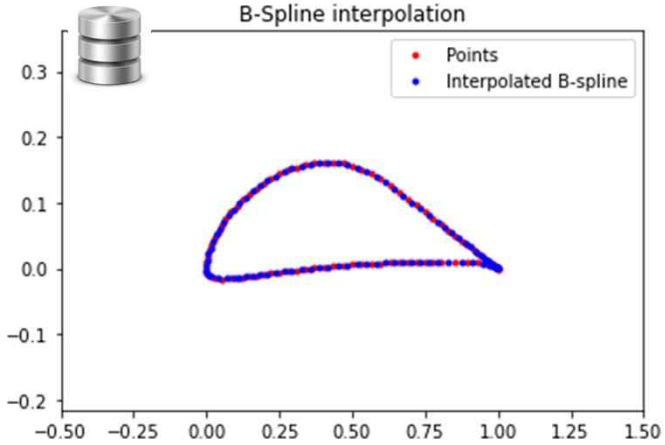
https://m-selig.ae.illinois.edu/ads/coord_database.html

Dataset preparation & preprocess

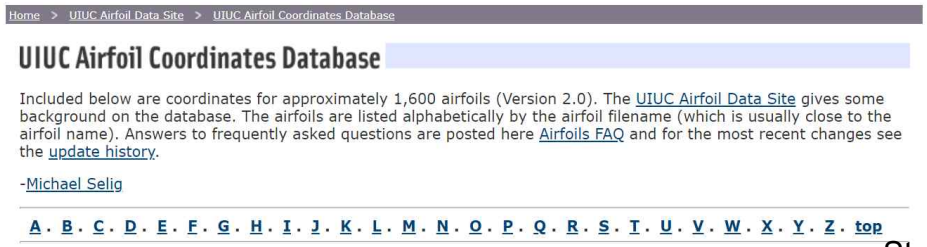


Data Crawling

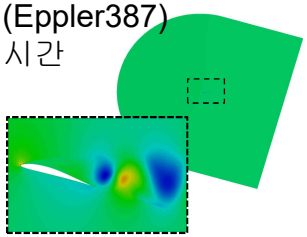
UIUC_Airfoil_Crawler_by_JoshTheEngineer.py

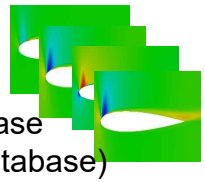


Visualization_UIUC_airfoils.ipynb



Unsteady case
(Eppler387)
시간





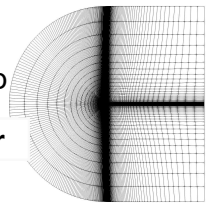
Steady case
(UIUC database)
형상 or 유동조건

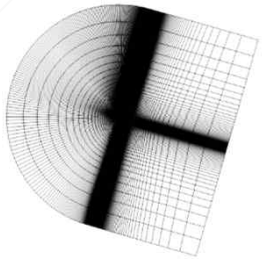
cleaning_UIUC.py
change_foldername.py

Gridding

foldering_griding.py
kgrid
preflow
in Gen_plot3d

KFLOW
2D_uComp
Flowsolver

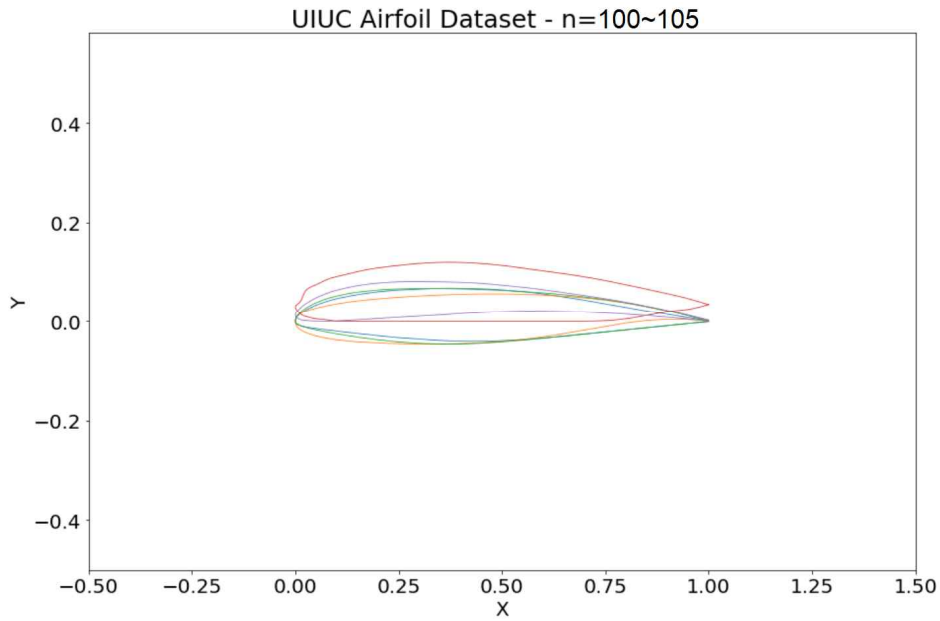




Dataset repository (현재 3개 유동조건만) –
https://dataon.kisti.re.kr/search/view.do?mode=view&svcId=f3face48acf03e666ba0d44394438ed8

visualization_UIUC_airfoils.ipynb

```
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
#from scipy import interpolate
#import time

airfoil_data_path = "d:\JupyterNBook\Airfoildata\Shapedata\\"
file_name = os.listdir(airfoil_data_path)
print(file_name[0:10])

plt.figure(figsize=(15,10))
for i in range(100,105):
    f = airfoil_data_path+ file_name[i]
    df = pd.read_csv(f, na_filter=True, dtype='float64', delimiter=' ', skipinitialspace=True, skiprows=3, header=None)
    ctr = df.values
    x = ctr[:,0]
    y = ctr[:,1]
    plt.plot(x, y, linewidth=0.8)
plt.axis([min(x)-0.5, max(x)+0.5, min(y)-0.5, max(y)+0.5])
plt.xlabel('X', fontsize=20)
plt.ylabel('Y', fontsize=20)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.title('UIUC Airfoil Dataset - n=100~105', fontsize=25)
plt.show()
```



cleaning_UIUC.py

Head 제거
- next() 함수

```
path_dir = './'
file_list = os.listdir(path_dir)
n_remove = 2
numd = len(file_list)
#print(numd, file_list[0])
for i in range(numd):
    with open(file_list[i],'r', encoding='cp949') as f:
        with open("../UIUC_DB_KISTI/"+file_list[i],'w') as f1:
            next(f)
            next(f)
            for line in f:
                f1.write(line)
```

list_airfoils.py

```
import os

path_dir = './UIUC_DB_KISTI/'
file_list = os.listdir(path_dir)
file_list.sort()

n_remove = 2
numd = len(file_list)
#print(numd, file_list[0])
f = open("list_airfoils.dat", "w")
for i in range(0,numd):
    filename = os.path.splitext(file_list[i])[0]
    f.write('AirfoilName(' + str(i+1).rjust(4,'0') + ')' + str(filename) + '\n')
    #os.makedirs("./make_p2d/"+filename) #, exist_ok=True)
f.close()
```

Airfoil명 추출
- splitext() 함수 등

change_foldername.py

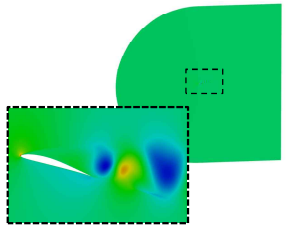
cmd 명령
- os.system() 함수

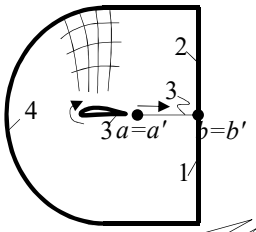
```
import os

numd = 1550
path_dir = './'
for i in range(138,numd+1):
    os.system("mv Airfoil_"+str(i)+" Airfoil_"+str(i).rjust(4,'0'))
    #os.system("mv Airfoil_"+str(i).rjust(4,'0') +" Airfoil_"+str(i-1).rjust(4,'0'))
```

[2단계] UIUC Airfoil 형상 기반 Training Dataset 구축

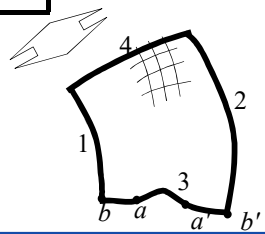
Dataset preparation & preprocess

ExtracData_OOO.py



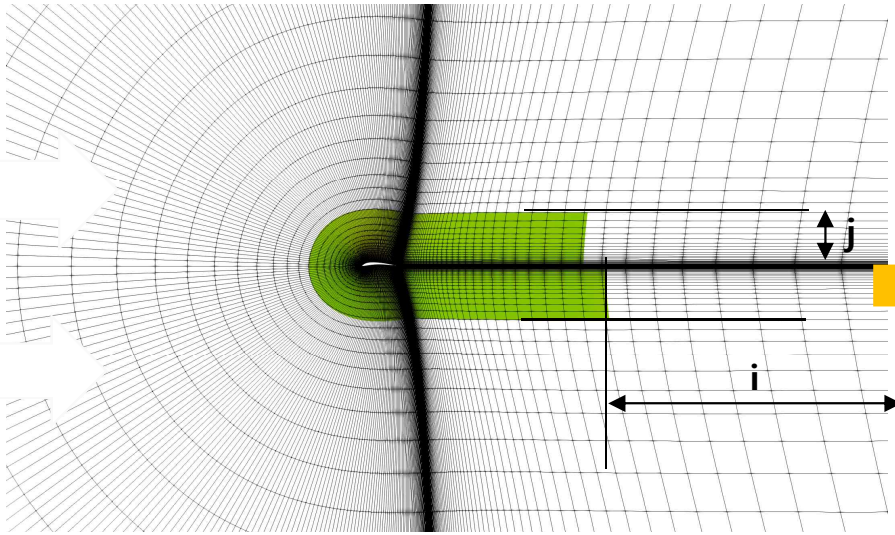


입력정의
(공간좌표)



Data Size
Reduction

Stack Arrays
into a Binary File



메모리 효율을 위한 데이터 사이즈 축소
(i, j 만큼 절삭)

cuttail, glayer
crop_i, crop_j

array_OOO.npz

Numpy based
Binary File

Tip

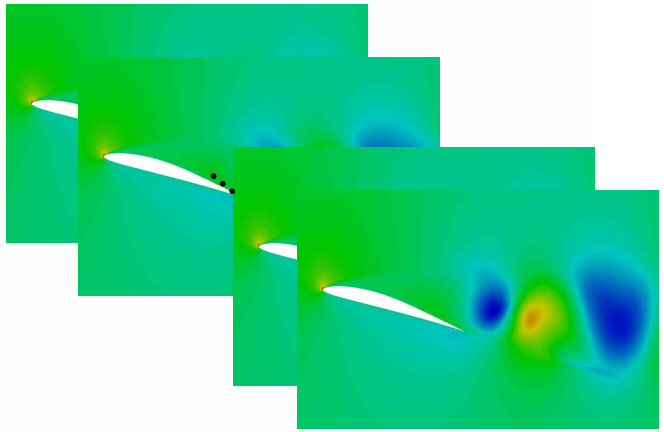
- 순서: **Pandas Data** 로딩 → **Numpy array**로 변환 → **Data** 프로세싱 **(Slice** 등) → 반출/쓰기
- 파이썬 플랫폼 **Datatype (List, Array, Pandas)** 중 **List**의 **Stack operation** 성능이 가장 좋고 **Array**의 **Stack** 성능이 가장 낮음 → **stack** 함수 사용시 **Array** 타입은 피해야 함

ExtracData_OOO.py (1/4)

```
import numpy as np
import time
import sys
import pandas as pd
import os

noCol = 11
numd = 20
initial_time = 101
inc_time = 4
zone1_i = 689
zone1_j = 145
glayer = 145
cuttail = 0
dt = 0.1
sim_data_path = "G:\\CFD\\Unsteady\\Eppler387\\sol01_RANS3\\"
Tecplot_header_in = "variables=X, Y, Z, Rho, U, V, W, P, T, Vor, Qcri"
Tecplot_header_out = "variables=X, Y, Rho, U, V, P"
```

- Stack snapshots

ExtracData_UnsteadyCase.py



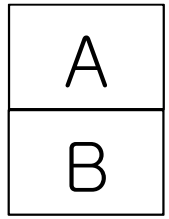
- 노드점 개수(N) = 689 x 145
- 시간스텝 개수(T) = 20

ExtracData_OOO.py (2/4)

Numpy Data stack의 일반 형태

```
# list of file names
filenames = []
merged = []
Ntime = 0
for i in range(0, numd):
        filenames.append(sim_data_path+"flo001.0000"+str(initial_time+i*inc_time).rjust(3,'0')+"uns")
        Ntime += 1
print(Ntime, filenames)
###
snapshot_data = np.array([])
POD = np.array([])
for i in range(0,numd):
        pd_data = pd.read_csv(filenames[i], dtype='float64', delimiter=' ', skipinitialspace=True, skiprows=2, header=None)
        #make it an np ndarray
        data = np.nan_to_num(pd_data.values)
        array_data = data.flatten()
        array_data = array_data.reshape(-1,noCol)
        if i==0:
                snapshot_data = array_data
                xy = snapshot_data[:,0:2]
                N = snapshot_data.shape[0]
                #POD = array_data[:,[3,4,5,7]].flatten()[:,None]
        else:
                snapshot_data = np.vstack((snapshot_data, array_data))
                snapshot_pod = array_data[:,[3,4,5,7]].flatten()[:,None]
                #POD = np.hstack((POD,snapshot_pod))
array_data1 = snapshot_data
shp = array_data1.shape
print(shp)
```

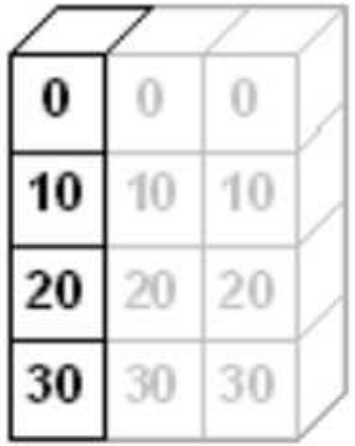
numpy.vstack(A, B)



ExtracData_OOO.py (3/4)

```
### Vectorize each variable
t_star = np.arange(Ntime)[:,None]*dt*inc_time # T(=1) x 1
T = t_star.shape[0]
#print(POD.shape)
xc_star = array_data1[:,0] # NT x 1
yc_star = array_data1[:,1] # NT x 1
NT = xc_star.shape[0]
dc_star = array_data1[:,3]
uc_star = array_data1[:,4]
vc_star = array_data1[:,5]
pc_star = array_data1[:,7]
print(xc_star.shape) #"X","Y","rh","u","v","w","p","M","vorticity"
#np.savez("./PODarray4.npz", xy=xy, snapshot=POD)
```

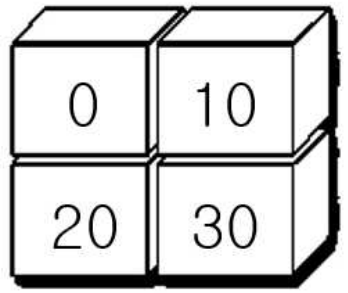
array_data1[:,0]



reshape

```
DC = np.reshape(dc_star, (T,N)).T # N x T
UC = np.reshape(uc_star, (T,N)).T # N x T
VC = np.reshape(vc_star, (T,N)).T # N x T
PC = np.reshape(pc_star, (T,N)).T # N x T
XC = np.reshape(xc_star, (T,N)).T # N x T
YC = np.reshape(yc_star, (T,N)).T # N x T
TC = np.tile(t_star, (1,N)).T # N x T
```

XC = np.reshape(xc_star, (T,N)).T

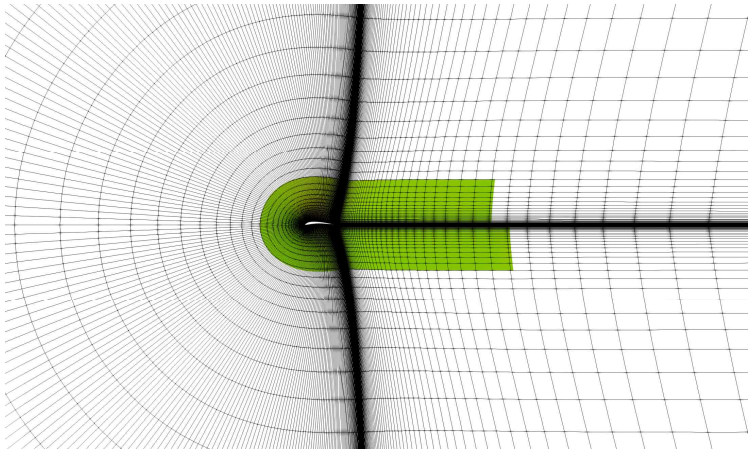


if T=2, N=2

ExtracData_OOO.py (4/4)

Numpy slice

```
#### Cut grid for memory efficiency
idx_x_slice = np.array([])
for i in range(glayer):
    idx_x_slice = np.append(idx_x_slice, np.arange(cuttail+i*zone1_i, \
                            (zone1_i-cuttail)+i*zone1_i)).astype('int32')
    print(idx_x_slice.shape[0])
DC_star = DC[idx_x_slice,:]
UC_star = UC[idx_x_slice,:]
VC_star = VC[idx_x_slice,:]
PC_star = PC[idx_x_slice,:]
XC_star = XC[idx_x_slice,:]
YC_star = YC[idx_x_slice,:]
TC_star = TC[idx_x_slice,:]

###
np.savez("array4(dataset_20timesteps_RANS_unsteady).npz", TC=TC_star, \
         XC=XC_star, YC=YC_star, DC=DC_star, UC=UC_star, VC=VC_star, PC=PC_star)
### check
#saved = np.load("./array4(dataset_20timesteps_RANS_unsteady).npz")
#print(saved['TC'])
#for i in range(Ntime):
#        p3d_result = np.hstack((x_test, y_test, D_pred[:,snap:snap+1], U_pred[:,snap:snap+1], \
#                                V_pred[:,snap:snap+1], P_pred[:,snap:snap+1]))
#        np.savetxt("../Data/airfoil_unsteady/RANS/Case_flo_(t="+str(i)+").dat", \
#                   p3d_result, delimiter=" ", header="variables = X, Y, rho, u, v, p \n zone i=" \
#                   +str(zone1_i)+" j="+str(glayer)+" ", comments=' ')
```



[Tip] Do not use “stack np.array” for Huge Dataset

```
### It is better to use "append list" than "vstack array"
snapshot_data = []
for i in range(Ncase):
    pd_data1 = pd.read_csv(foilcases[i], na_filter=True, dtype='float64', delimiter=' ', \
                           skipinitialspace=True, skiprows=2, header=None)
    data = np.nan_to_num(pd_data1.values)
    array_data = data.flatten()
    array_data = array_data.reshape(-1,noCol)
    snapshot_data.append(array_data)
    if i==0:
        N = array_data.shape[0]
    '''
    if i==0:
        snapshot_data = array_data
        N = snapshot_data.shape[0]
        #POD = array_data[:,[3,4,5,7]].flatten()[:,None]
    else:
        snapshot_data = np.vstack((snapshot_data, array_data))
        #snapshot_pod = array_data[:,[3,4,5,7]].flatten()[:,None]
        #POD = np.hstack((POD,snapshot_pod))
    '''
snapshot_dataset = np.vstack((snapshot_data))
```

ExtracData_AirDB_cuttail_cond_single.py

Stack이 많을 때는 List type 사용!

[Tip] Parallel Computing for Huge Dataset (1/3)

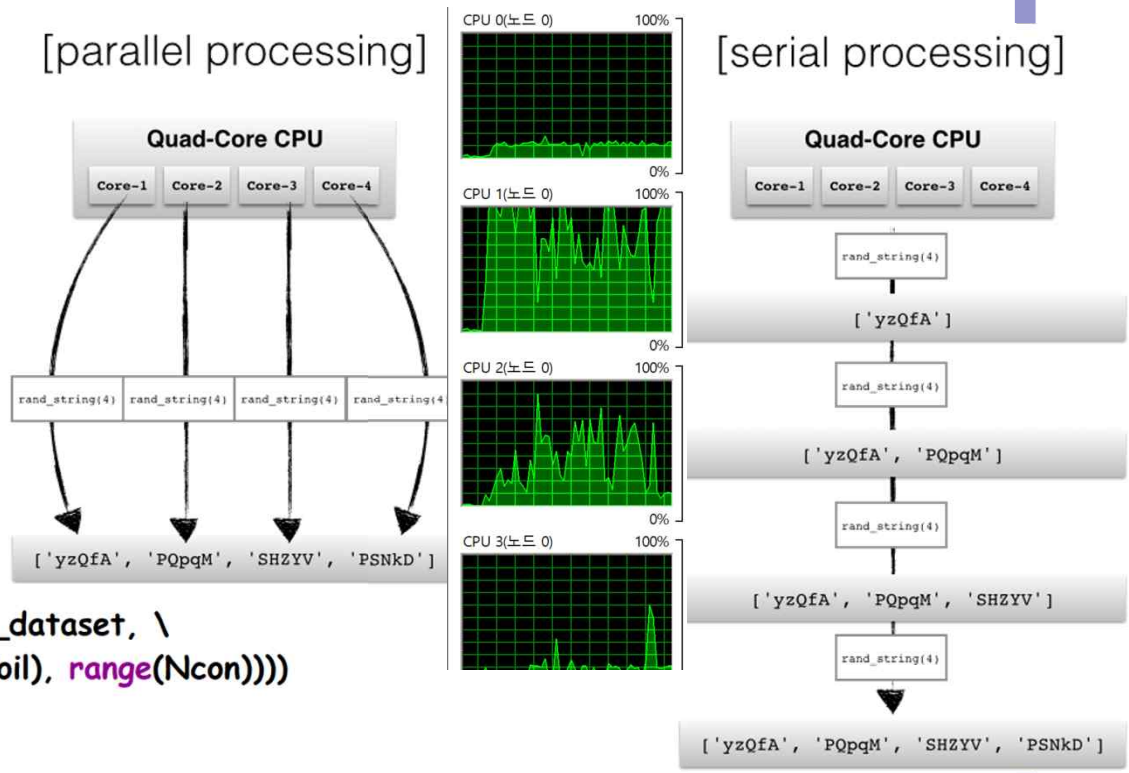
- Many Data Case : 1550종 x 多유동조건(Re, Mach, AOA)

```
# list of file names
Ncon = Re.shape[0] * Mach.shape[0] * AOA.shape[0]
Ncase = 0
foilcases = []
for j in range(Nfoil):
    sim_data_path = data_path + 'Airfoil_' + str(j+1).rjust(4,'0') + '/'
    for k in range(1,Re.shape[0]+1):
        for l in range(1,Mach.shape[0]+1):
            for m in range(1,AOA.shape[0]+1):
                foilcases.append(sim_data_path+"result_"+str(k)+"_"+ \
                                 str(l)+"_"+str(m).rjust(2, '0')+"/flo001.dat")
            Ncase += 1
print(Nfoil, Ncase)
```

[Tip] Parallel Computing for Huge Dataset (2/3)

ExtracData_AirDB_cuttail_cond_parallel.py

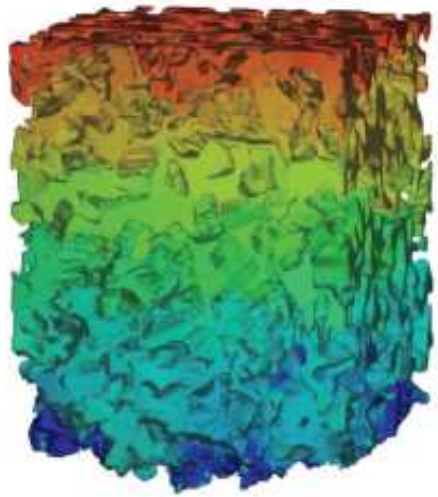
```
from multiprocessing import Pool, cpu_count
def snapshot_dataset(i,j):
        pd_data1 = pd.read_csv(foilcases[i*Ncon+j], na_filter=True, \
                               dtype='float64', delimiter=' ', skipinitialspace=True, \
                               skiprows=2, header=None)
        data = np.nan_to_num(pd_data1.values)
        array_data = data.flatten()
        array_data = array_data.reshape(-1,noCol)
        snapshot_data = array_data

        return snapshot_data

### Start Time
start = int(time.time())

### Distributed Session
num_cores = cpu_count() #4
pool = Pool(num_cores)
snapshot_dataset = np.concatenate(pool.starmap(snapshot_dataset, \
                                               product(range(Nfoil), range(Ncon))))
shp = snapshot_dataset.shape
print(shp)
pool.close()
pool.join()

### check end time
print("***run time(sec) :", int(time.time()) - start)
```



[Tip] Parallel Computing for Huge Dataset (3/3)

- Single Big Data over Memory size

```
# loading file with pandas
import pandas as pd
%time data_1 = pd.read_csv('/CFD/bigdata_turbulence.dat')

- Output:
CPU times: user 601 s, sys: 58.2 s, total 659.2 s
Wall time: 676 s

# loading the file using dask
import dask.dataframe as dd
%time data_2 = dd.read_csv("/CFD/bigdata_turbulence.dat")

- Output:
CPU times: user 12.4 s, sys: 7.8 s, total 20.2 s
Wall time: 24.6 s
```



[3단계] 학습에 활용된 AI 모형들

POD 기반 축소모형

스냅샷 & 모드 기반 차원 축소

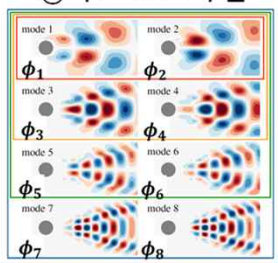
POD+SVD+OOO.py
POD+Evaluator+OOO.py

① Snapshot Data 수집 (해석,실험)

② Data 행렬 수립

$$[X] = [\vec{x}_1\ \vec{x}_2\ \vec{x}_3]$$

③ 주요 모드 추출



④ POD 축소모형 수립

Singular value decomposition of X

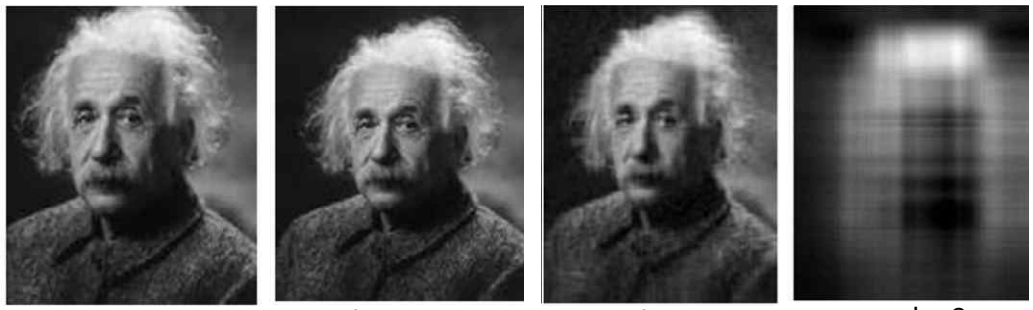
ϕ : Proper orthogonal mode

$$r_i^{(j)} = \langle X^{(j)} \phi_i \rangle$$

Proper orthogonal decomposition of u

$$u_{POD} = \sum_{i=1}^{m} \phi_i r_i$$

이미지 압축 기법 채용 : Singular Value Decomposition (SVD)



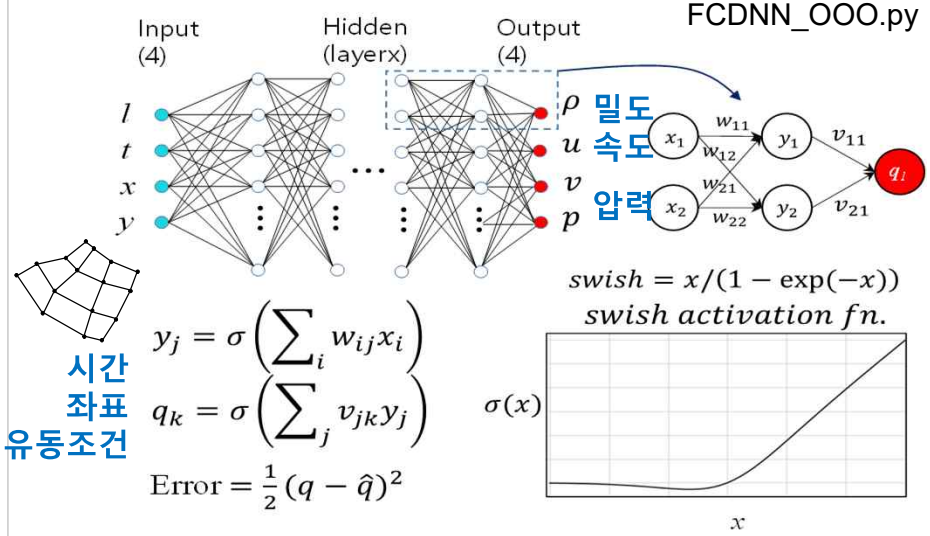
k=264 k=112 k=12 k=2

Image compression with decreasing selected no. of SVD basis

딥러닝 기반 축소모형

범용 Approximator 채용 : Fully-Connected Neural Network

FCDNN_OOO.py



시간
좌표
유동조건

$$y_j = \sigma\left(\sum_i w_{ij} x_i\right)$$

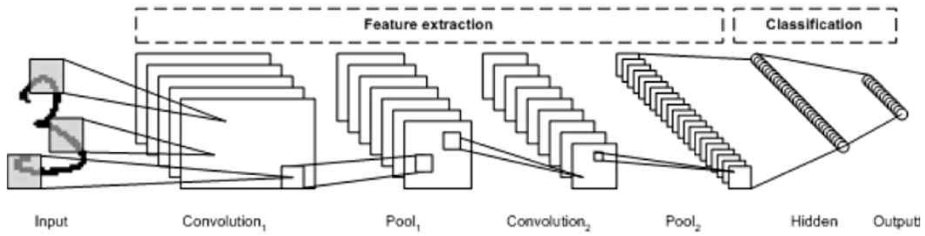
$$q_k = \sigma\left(\sum_j v_{jk} y_j\right)$$

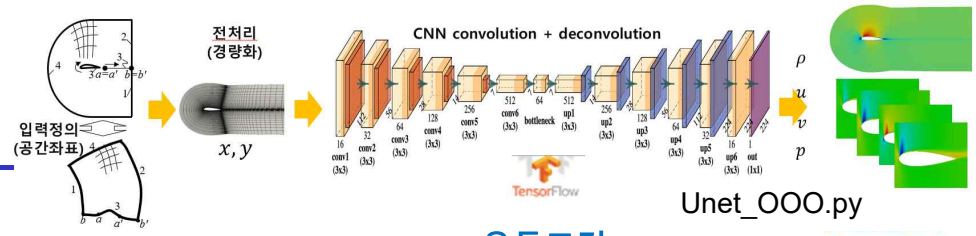
$$\text{Error} = \frac{1}{2}(q - \hat{q})^2$$

$$swish = x/(1 - \exp(-x))$$

swish activation fn.

형상인식 알고리즘 채용 : Convolutional Neural Network (CNN)





Unet_OOO.py

날개형상

유동조건

KSCFE

12th CFD Short Course

POD = PCA (Principle Component Anaysis)

[1] 적합직교분해 (POD) 기반 축소모형

ϕ : POD basis, a : expansion coefficient

POD description: $X = q(\xi, s) - q_{mean}(\xi) = \sum_j a_j(s)\phi_j(\xi)$

q is snapshot data, n-DOF spatial(ξ), m-DOF temporal(s)

Method of snapshot: $X = [x(s_1) \;\; x(s_2) \;\cdots\; x(s_n)] \in \mathbb{R}^{m\times n}$

침습(Intrusive) 기법: $\dot{u} = N(u) - \nabla p, \quad \dot{a}_j = \langle N(u), \phi_j \rangle - \langle \nabla p, \phi_j \rangle$ ex) Galerkin Projection

비침습(Non-intrusive) 기법: $a_j(t) = \langle q(\xi,t) - \overline{q}(\xi), \phi_j(\xi) \rangle = \langle x(t), \phi_j \rangle$ ex) Local linear regression

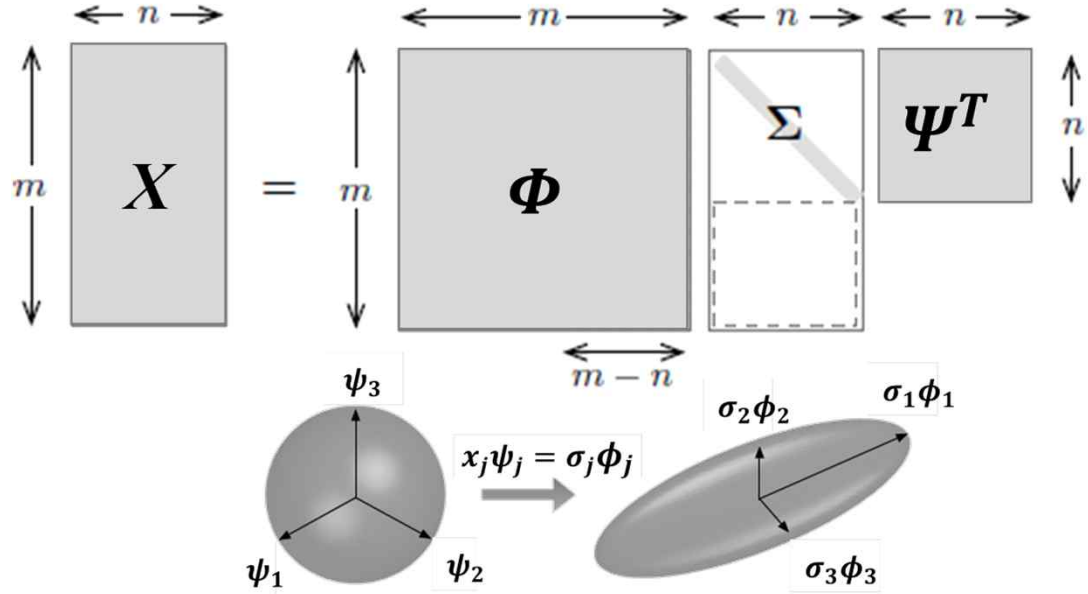
SVD decomposition:

$$X = \Phi \Sigma \Psi^T$$

Orthogonal basis 추출

where $\Phi = [\vec{\phi}_1 \; \vec{\phi}_2 \cdots \vec{\phi}_m] \in \mathbb{R}^{m\times m}$, $\Psi = [\vec{\psi}_1 \; \vec{\psi}_2 \cdots \vec{\psi}_n] \in \mathbb{R}^{n\times n}$, $\Sigma \in \mathbb{R}^{m\times n}$

$$X\Psi = \Phi\Sigma,$$



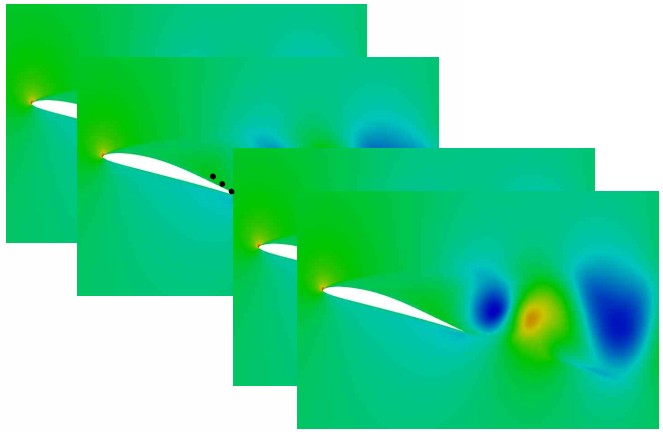
(2018, 2020) Taira et al. Modal Analysis of Fluid Flows, *AIAA J.*

POD+SVD-OOO.py

```
import numpy as np
import csv
import pandas as pd
import pickle
import RomObject

#initiate variables
noCol = 11
noConcernVar = 4
numd = 20
initial_time = 203
inc_time = 4
zone1_i = 689
zone1_j = 145
dt = 0.1
sim_data_path = "~/Data/airfoil_unsteady/"
res_data_path = "../Data/airfoil_unsteady/results/"
Tecplot_header_in = "variables=X, Y, Z, Rho, U, V, W, P, T, Vor, Qcri"
Tecplot_header_out = "variables=X, Y, Rho, U, V, P"
#create time_configurations
_train_times = [list(range(201, 280, 4))]
```

- Stack snapshots

POD+SVD-Unsteady.py



- 노드점 개수(N) = 689 x 145
- 시간스텝 개수(T) = 20

POD+SVD-OOO.py

SVD operation
→ u, s, v 추출 → Basis

```
#pickle the times
with open('times4.pkl', 'wb') as output:
    pickle.dump(times, output, pickle.HIGHEST_PROTOCOL)

#read extracted array
saved_npz = np.load("./PODarray4.npz")
snapshot_data = saved_npz['snapshot']
shp = snapshot_data.shape
xy = saved_npz['xy']
print(snapshot_data.shape)
np.savetxt("./xy.csv", xy)

#mean center the data
mean_array = None
mean_data_tensor = np.mean(snapshot_data, axis=1)
print(mean_data_tensor)
mean_centered_data = np.subtract(snapshot_data, np.tile(mean_data_tensor, (numd,1)).T)

#Singular Value Decomposition
u, s, v = np.linalg.svd(mean_centered_data, compute_uv=True, full_matrices=False)
print(mean_centered_data.shape)
print(u.shape)
print(v.shape)
print(s.shape)
```

POD+SVD-OOO.py

Coefficients, s-energy 계산
ROM parameters 저장

```
# POD coefficients
compute_coeffs = np.matmul(np.transpose(mean_centered_data),u)
e = np.sum(s)
s_energy = np.divide(s,e)*100
coeffs = compute_coeffs
mean_array = mean_data_tensor

print('Raw s-matrix')
print(s)
print('Cumulative Energy')
print(e)
print('Normalized Energy')
print(s_energy)

#some assertions for correctness
#UU^T = I
uut = np.matmul(np.transpose(u),u)
np.testing.assert_almost_equal(uut, np.eye(u.shape[1],u.shape[1]))
#VV^T = I
vvt = np.matmul(np.transpose(v),v)
np.testing.assert_almost_equal(uut, np.eye(v.shape[1],v.shape[1]))

print(coeffs.shape)
print(coeffs.dtype)
print(compute_coeffs)

#Save coefficients (coeffs) , left-singularnvectors (u) and singular values (s)
rom_object = RomObject.romobject(u, s_energy, coeffs, mean_array)
with open('rom-object4.pkl', 'wb') as output:
    pickle.dump(rom_object, output, pickle.HIGHEST_PROTOCOL)
```

POD+Evaluator-OOO.py

추론 지점의
Coefficients 계산

$$X = q(\xi, s) - q_{mean}(\xi) = \sum_j a_j(s)\phi_j(\xi)$$

```
import tensorflow.compat.v1 as tf
import numpy as np
import pickle
import pandas as pd

#perform coefficient interpolation here
total_steps =19
infer_times = np.arange(202, 240, 2)
noConcernVar = 4
zone1_i = 689
zone1_j = 145

#READ TRAIN TIMES
with open('times4.pkl', 'rb') as input:
    train_times = pickle.load(input)[0]

#read in saved rom object
with open('rom-object4.pkl', 'rb') as input:
    read_rom = pickle.load(input)
```

POD+Evaluator-Unsteady.py

POD+Evaluator-OOO.py

ROM parameters 반입

```
#READ TRAIN TIMES
with open('times4.pkl', 'rb') as input:
    train_times = pickle.load(input)[0]

#read in saved rom object
with open('rom-object4.pkl', 'rb') as input:
    read_rom = pickle.load(input)

#read xy-coordinates
pd_data = pd.read_csv('xy.csv', dtype='float64', delimiter=' ', header=None, skipinitialspace=True)
xydata = pd_data.values
print(xydata.shape)
print(xydata)

#read SVD matrix
coeffs = read_rom.coeffsmat
u = read_rom.umat
mean_data = read_rom.mean_data
mean_tensor = tf.constant(mean_data, name="mean_data_tensor")
```

POD+Evaluator-OOO.py

Linear interpolation

$$\vec{a}(t^*) = \vec{a}(t_k) + (\vec{a}(t_{k+1}) - \vec{a}(t_k)) \frac{(t^* - t_k)}{(t_{k+1} - t_k)}$$

```
# Use Tensorflow 1.x
for i, x in zip(range(total_steps), infer_times):
    print(i)
    hi_idx = [idx for idx,v in enumerate(train_times) if v > x][0]
    lo_idx = hi_idx - 1
    #interpolate coefficients
    interp_coeffs = coeffs[lo_idx] + \
          (coeffs[hi_idx]-coeffs[lo_idx])*(x-train_times[lo_idx])/(train_times[hi_idx]-train_times[lo_idx])
    int_coeff_tensor = tf.Variable(interp_coeffs)
    #add a dim to make it a 2-D tensor
    int_coeff_tensor = tf.expand_dims(int_coeff_tensor, 0)
    print(int_coeff_tensor)
    #compute the POD approximation
#    init_op = tf.global_variables_initializer()
    with tf.Session() as sess:
        sess.run(tf.global_variables_initializer())
        result_op = tf.matmul(int_coeff_tensor, tf.transpose(u) )
        modal_result = sess.run(tf.transpose(result_op))
        modal_result.flatten()
        mean_tensor.eval()
        result_op = tf.add( modal_result, mean_tensor)
        result = sess.run(result_op)
    result = result.reshape((int(modal_result.shape[0]/noConcernVar), noConcernVar))
    height, width = result.shape
    #Create p3d field data
    p3d_result = np.hstack((xydata, result))
    #Save tecplot field data
    np.savetxt("./Case_flo_POD_t="+str(x)+".dat", p3d_result,     delimiter=" ", \
              header="variables = X, Y, rho, u, v, p \n zone i= "+str(zone1_i)+", j=    " \
              +str(zone1_j), comments=' ')
```


KSCFE

[확장] 유동조건에 따른 Regression

POD+SVD-Steady.py

```
for i, x in zip(range(total_steps), input_para):
    #print(i)
    hi_para = np.zeros(Npara)
    lo_para = np.zeros(Npara)
    for k in range(Npara):
        print(k)
        hi_para[k] = min([v for idx_h, v in enumerate(read_paras[:,k]) if v >= x[k]])
        lo_para[k] = max([w for idx_l, w in enumerate(read_paras[:,k]) if w <= x[k]])
    hi_idx = np.where((read_paras == np.array(hi_para)).all(axis=1))[0]
    lo_idx = np.where((read_paras == np.array(lo_para)).all(axis=1))[0]
    print(hi_para, lo_para, hi_idx, lo_idx, '==========')
    #interpolate coefficients
    if hi_idx != lo_idx:
        interp_coeffs = coeffs[lo_idx] + \
            (coeffs[hi_idx]-coeffs[lo_idx])*np.sqrt(np.sum((x-read_paras[lo_idx])**2)) \
            /np.sqrt(np.sum((read_paras[hi_idx]-read_paras[lo_idx])**2))
    else:
        interp_coeffs = coeffs[lo_idx]
    int_coeff_tensor = tf.Variable(interp_coeffs)
```

POD 결과

t

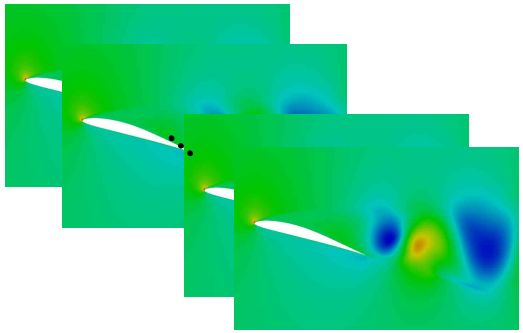
시간스텝 1) 10 s

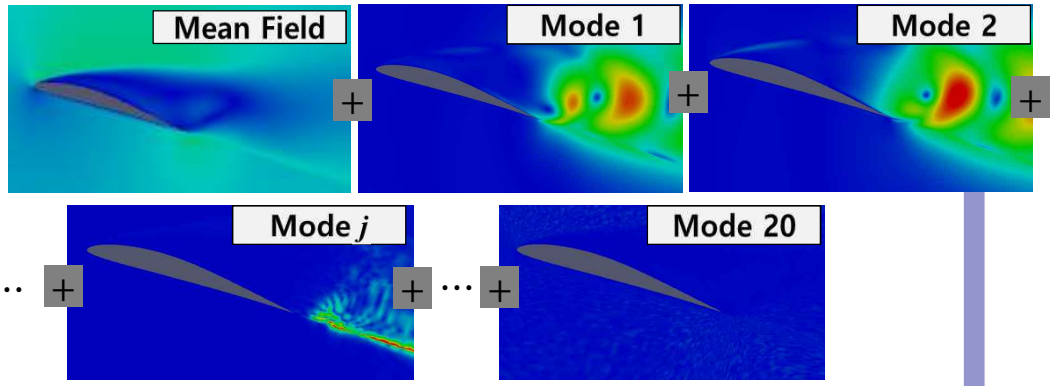
시간스텝 2) 10.1 s

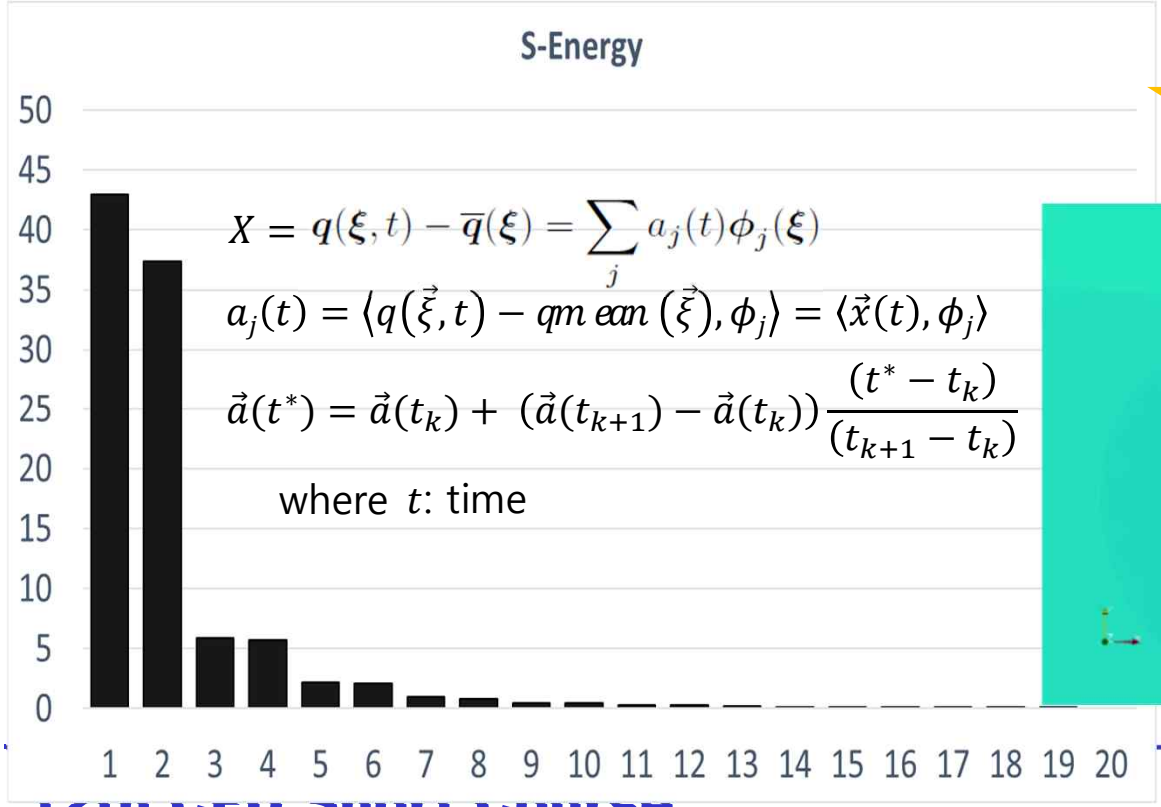
⋮ ⋮

시간스텝 19) 11.8 s

시간스텝 20) 11.9 s







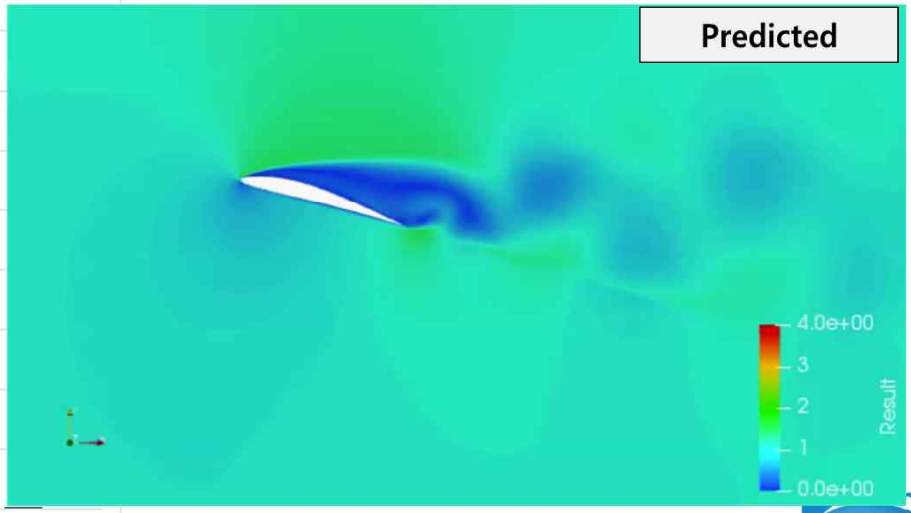
$$X = q(\xi,t) - \overline{q}(\xi) = \sum_j a_j(t)\phi_j(\xi)$$

$$a_j(t) = \langle q(\vec{\xi},t) - qm\,eann\,(\vec{\xi}), \phi_j \rangle = \langle \vec{x}(t), \phi_j \rangle$$

$$\vec{a}(t^*) = \vec{a}(t_k) + (\vec{a}(t_{k+1}) - \vec{a}(t_k))\frac{(t^* - t_k)}{(t_{k+1} - t_k)}$$

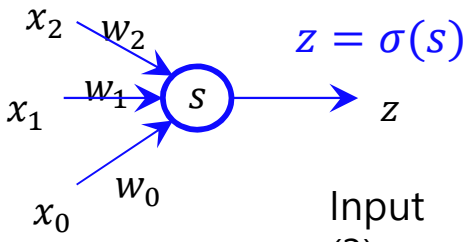
where t: time

$$q(\vec{\xi},t) = q_{mean}(\vec{\xi}) + \sum_j a_j(t)\phi_j(\vec{\xi})$$



[2] Fully-Coonected DNN 축소모형

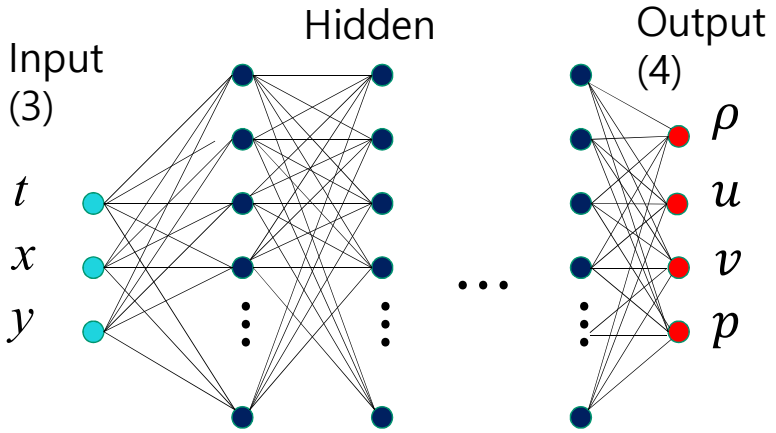
형상 고정



✓ **Network 정보**

- 입력층: 형상, 시간, 공간
- 출력층: 유동장
- 은닉층: 20 layers with 40 neurons

(NVIDIA P100 사용, 20시간, iteration당 0.5~1초 소요)

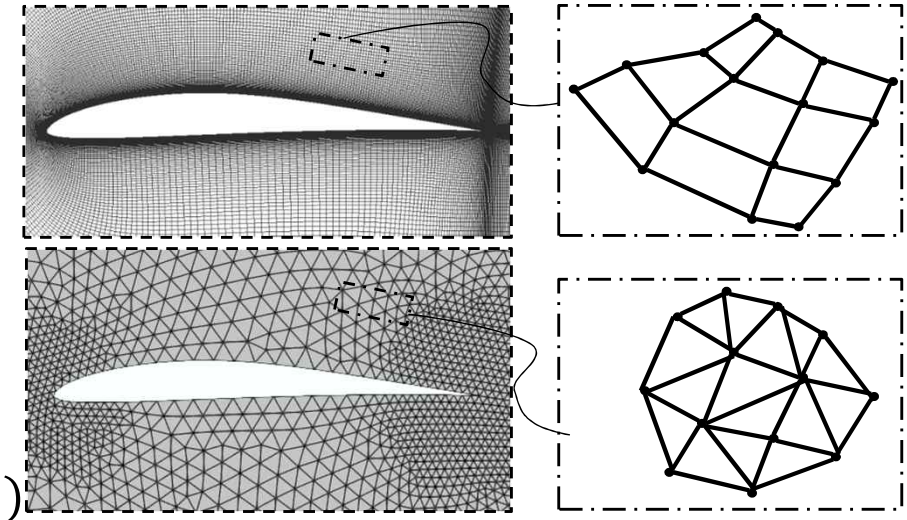


✓ **Loss function**

$$MSE_q = \frac{1}{N_q}\sum_{i=1}^{N_q}\left|q_{pred}\left(t^i, x^i, y^i\right) - q_{HF}\right|^2$$

$where \quad q = [\rho \;\; u \;\; v \;\; p]$

pred: 추론값, *HF*: high-Fidelity해석

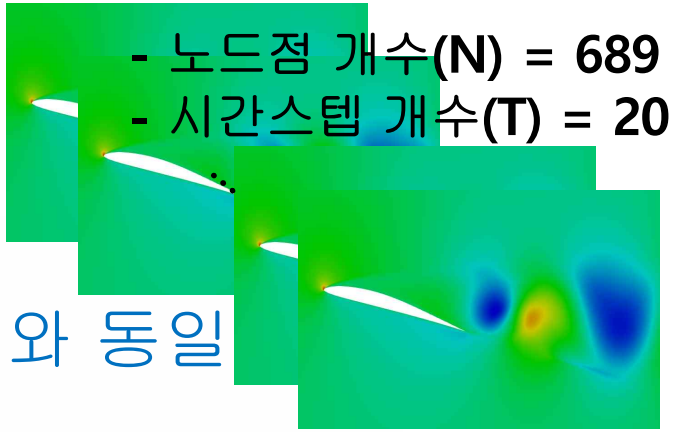


✓ **Optimizer & Activation function**

- Adam optimizer with learning rate 10^{-2}~10^{-5}
- $swish := x\,sigmoid(x) = x/(1-\exp(-x))$

FCDNN_OOO.py

- 노드점 개수(N) = 689 x 145
- 시간스텝 개수(T) = 20



앞의 POD 문제와 동일

FCDNN-Unsteady.py

```
if __name__ == "__main__":
    with tf.device('/gpu:0'):
        batch_size = 40000
        layers = [3] + 10*[4*10] + [4]

        # Load Data
        sim_data_path = "~/AIRFOIL/Unsteady/Eppler387/sol01_RANS3"
        # list of file names
        filenames = []

                                    ⋮
        #Time steps
        t_star = np.arange(initial_time, initial_time+numd*inc_time+1, inc_time)*dt # 1xT(=1)
        infer_times = np.arange(initial_time+1, initial_time+numd*inc_time+2, inc_time)*dt

                                    ⋮
        ###
        saved_npz = np.load("./array4.npz")

        DC_star = saved_npz['DC']
        UC_star = saved_npz['UC']
        VC_star = saved_npz['VC']
        PC_star = saved_npz['PC']
```

FCDNN_OOO.py

airfoil surface nodes 추출

```
### extract airfoil surface nodes
idx_bottom = np.where(xydata[:,0] == xydata[0,0])[0]
for i in range(idx_bottom[1]):
    if(xydata[i,1] != xydata[idx_bottom[1]-i,1]):
        break
idx_tip = [i-1, idx_bottom[1]-i+1]
idx_sur = []
for j in range(zone1_j-2):
    idx_sur[j] = idx_sur.append(np.arange(idx_tip[0]+(j+1)*(zone1_i-2*cuttail), \
                                          idx_tip[1]+(j+1)*(zone1_i-2*cuttail)+1))
idx_x_sur = np.arange(idx_tip[0],idx_tip[1]+1)
```

⋮

FCDNN_OOO.py

in-out 변수, training 정의

```
T_star = np.tile(t_star[:,None], (1,N))
X_star = np.tile(xydata[:,0], (T,1))
Y_star = np.tile(xydata[:,1], (T,1))
d_data = DC_star.T.flatten()[:,None]
u_data = UC_star.T.flatten()[:,None]
v_data = VC_star.T.flatten()[:,None]
p_data = PC_star.T.flatten()[:,None]
t_data = T_star.flatten()[:,None]
x_data = X_star.flatten()[:,None]
y_data = Y_star.flatten()[:,None]

# Training
model = DLROM(t_data, x_data, y_data,
              d_data, u_data, v_data, p_data,
              layers, batch_size, Pec = 1000, Rey = 10)

model.train(total_time = 40, learning_rate=1e-2)
```

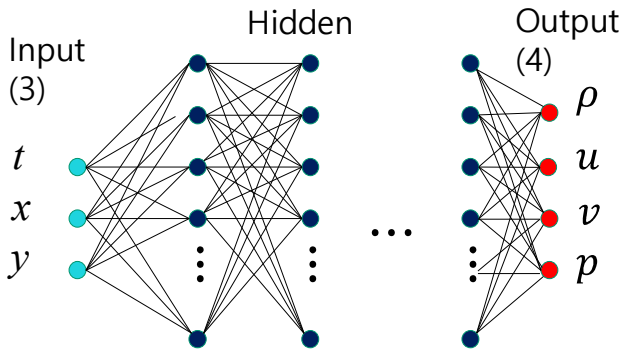
FCDNN_OOO.py

```
import tensorflow.compat.v1 as tf
import numpy as np
import time
import sys
import os
import pandas as pd

from CFDFunctions3 import neural_net, tf_session, mean_squared_error, relative_error

tf.compat.v1.disable_eager_execution()

class DLROM(object):

    def __init__(self, t_data, x_data, y_data, l_data,
                 d_data, u_data, v_data, p_data,
                 layers, batch_size, Pec, Rey):
```

FCDNN_OOO.py

network, loss, optimizer 정의

```
# placeholders
[self.d_data_tf, self.u_data_tf, self.v_data_tf, self.p_data_tf] = [tf.placeholder(tf.f
[self.t_data_tf, self.x_data_tf, self.y_data_tf, self.l_data_tf] = [tf.placeholder(tf.f

# neural networks
self.net_duvp = neural_net(self.t_data, self.x_data, self.y_data, layers = self.layers)

[self.d_data_pred,
 self.u_data_pred,
 self.v_data_pred,
 self.p_data_pred] = self.net_duvp(self.t_data_tf,
                                   self.x_data_tf,
                                   self.y_data_tf)

# loss
self.loss = mean_squared_error(self.d_data_pred, self.d_data_tf) + \
            mean_squared_error(self.u_data_pred, self.u_data_tf) + \
            mean_squared_error(self.v_data_pred, self.v_data_tf) + \
            mean_squared_error(self.p_data_pred, self.p_data_tf)

# optimizers
self.global_step = tf. Variable(0, trainable = False, name='global_step')
self.learning_rate = tf.placeholder(tf.float32, shape=[])
self.optimizer = tf.train.AdamOptimizer(learning_rate = self.learning_rate)
self.train_op = self.optimizer.minimize(self.loss, global_step=self.global_step)

self.sess, self.saver = tf_session()
```



KSCFE

FCDNN_OOO.py

train 함수

```
def train(self, total_time, learning_rate):

    N_data = self.xydata.shape[0]
    start_time = time.time()
    running_time = 0
    it = 0
    while running_time < total_time:

        idx_data = np.random.choice(N_data, min(self.batch_size, N_data))
        if it == 50000:
            learning_rate = 1e-3
        if it == 100000:
            learning_rate = 5e-4
        if it == 150000:
            learning_rate = 1e-4
        (t_data_batch,
         x_data_batch,
         y_data_batch,
         d_data_batch, u_data_batch,
         v_data_batch, p_data_batch) = (self.t_data[idx_data,:],
                        self.x_data[idx_data,:],
                        self.y_data[idx_data,:],
                        self.d_data[idx_data,:],
                        self.u_data[idx_data,:],
                        self.v_data[idx_data,:],
                        self.p_data[idx_data,:])

        tf_dict = {self.t_data_tf: t_data_batch,
                   self.x_data_tf: x_data_batch,
                   self.y_data_tf: y_data_batch,
                   self.u_data_tf: u_data_batch,
                   self.v_data_tf: v_data_batch,
                   self.d_data_tf: d_data_batch,
                   self.p_data_tf: p_data_batch,
                   self.learning_rate: learning_rate}

        self.sess.run([self.train_op], tf_dict)

        # Print
        if it % 10 == 0:
            elapsed = time.time() - start_time
            running_time += elapsed/3600.0
            [loss_value,
             learning_rate_value] = self.sess.run([self.loss,
                                                   self.learning_rate], tf_dict)
            print('It: %d, Loss: %.3e, Time: %.2fs, Running Time: %.2fh, Learning Rat
                  %(it, loss_value, elapsed, running_time, learning_rate_value))
            sys.stdout.flush()
            start_time = time.time()
        it += 1
    # save model parameters
    self.saver.save(self.sess, './model0/dnn.ckpt', global_step = self.global_step)
```

FCDNN_OOO.py

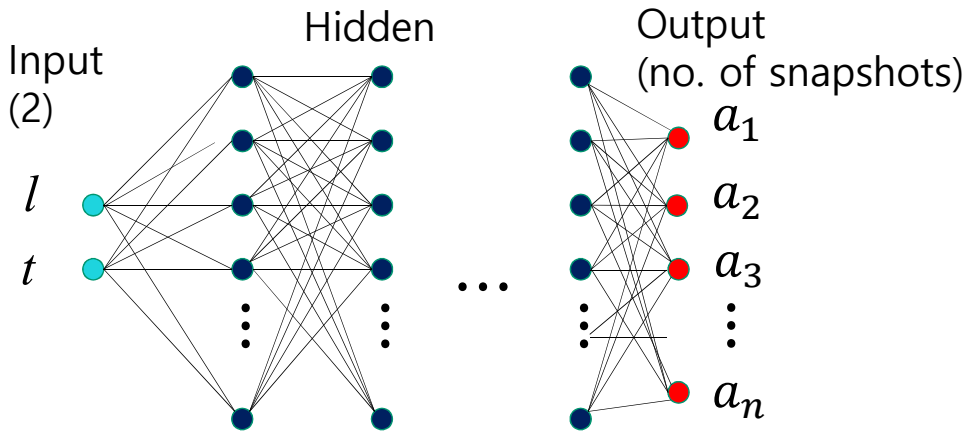
infer, 출력/쓰기

```
# Test Data
t_rom_test = infer_times[:,None]
T_test = np.tile(t_rom_test, (1,N))

# Write the predictions
for i in range(20):
    t_test = T_test.T[:,i:i+1]
    x_test = X_star.T[:,i:i+1]
    y_test = Y_star.T[:,i:i+1]

    # Prediction
    d_pred, u_pred, v_pred, p_pred = model.predict(t_test, x_test, y_test)
    p3d_result = np.hstack((xydata[:,0][:,None], xydata[:,1][:,None], d_pred, u_pred, v_pred, p_pred))
    np.savetxt("./Case_flo8_RANS_NN_cuttail2_t="+str(i)+".dat", p3d_result, delimiter=" ", \
               header="variables = X, Y, c, u, v, p \n zone i="+str(zone1_i-2*cuttail)+" j="+str(glayer) \
               +" ", comments=' ')
    # Error
    error_d = relative_error(d_pred, DC_star[:,i][:,None])
    error_u = relative_error(u_pred, UC_star[:,i][:,None])
    error_v = relative_error(v_pred, VC_star[:,i][:,None])
    error_p = relative_error(p_pred , PC_star[:,i][:,None])
    print('Error d: %e, u: %e, v: %e, p: %e' % (error_d, error_u, error_v, error_p))
```

PODNN_OOO.py

확장 - POD basis 학습

```
[self.a0_data_pred, self.a1_data_pred, self.a2_data_pred,
        self.a3_data_pred, self.a4_data_pred, self.a5_data_pred,
        self.a6_data_pred, self.a7_data_pred, self.a8_data_pred,
        self.a9_data_pred, self.a10_data_pred,
        self.a11_data_pred, self.a12_data_pred,
        self.a13_data_pred, self.a14_data_pred,
        self.a15_data_pred, self.a16_data_pred,
        self.a17_data_pred, self.a18_data_pred,
        self.a19_data_pred] = self.net_pod(self.l_pod_data, self.t_pod_data_tf)
```



[4단계] 공력성능 계산 CLCD.py

```
# Surface Data for CL/CD calculation
idx_bottom = np.where(single_data[:,0] == single_data[0,0])[0]
print(idx_bottom)
for i in range(idx_bottom[1]):
    if(single_data[i,1] != single_data[idx_bottom[1]-i,1]):
        break
idx_tip = [i-1, idx_bottom[1]-i+1]
idx_t_bd = np.concatenate([np.array([0]), np.random.choice(T-2, t_bd-2, replace=False)+1, np.array([T-1])])
idx_x_sur = np.arange(idx_tip[0],idx_tip[1]+1)
idx_x_sur2 = np.arange(idx_tip[0]+zone1_i,idx_tip[1]+zone1_i+1)
idx_t_bd = np.sort(idx_t_bd.astype('int32'))
idx_xc_bd = idx_x_sur #idx_x_bd
t_sur = t_star[idx_t_bd]
x_sur = single_data[:,0][idx_x_sur]
y_sur = single_data[:,1][idx_x_sur]
x_sur2 = single_data[:,0][idx_x_sur2]
y_sur2 = single_data[:,1][idx_x_sur2]
u_sur = UC[idx_xc_bd,:][:, idx_t_bd]#.flatten()[:,None]
v_sur = VC[idx_xc_bd,:][:, idx_t_bd]#.flatten()[:,None]
p_sur = PC[idx_xc_bd,:][:, idx_t_bd].flatten()[:,None]
d_sur = DC[idx_xc_bd,:][:, idx_t_bd].flatten()[:,None]
u_sur2 = UC[idx_x_sur2,:][:,idx_t_bd]#.flatten()[:,None]
v_sur2 = VC[idx_x_sur2,:][:,idx_t_bd]

# CL/CD calculation
F_L, F_D = predict_drag_lift(t_sur, x_sur, y_sur, x_sur2, y_sur2, u_sur, v_sur, p_sur, u_sur2, v_sur2, d_sur)
result = np.hstack((F_L[:,None], F_D[:,None]))

#Write
np.savetxt(sim_data_path+"FOM_CLCD_RANS3_dt4.dat", result, delimiter=" ", header="variables = CL, CD")
```

SCFE

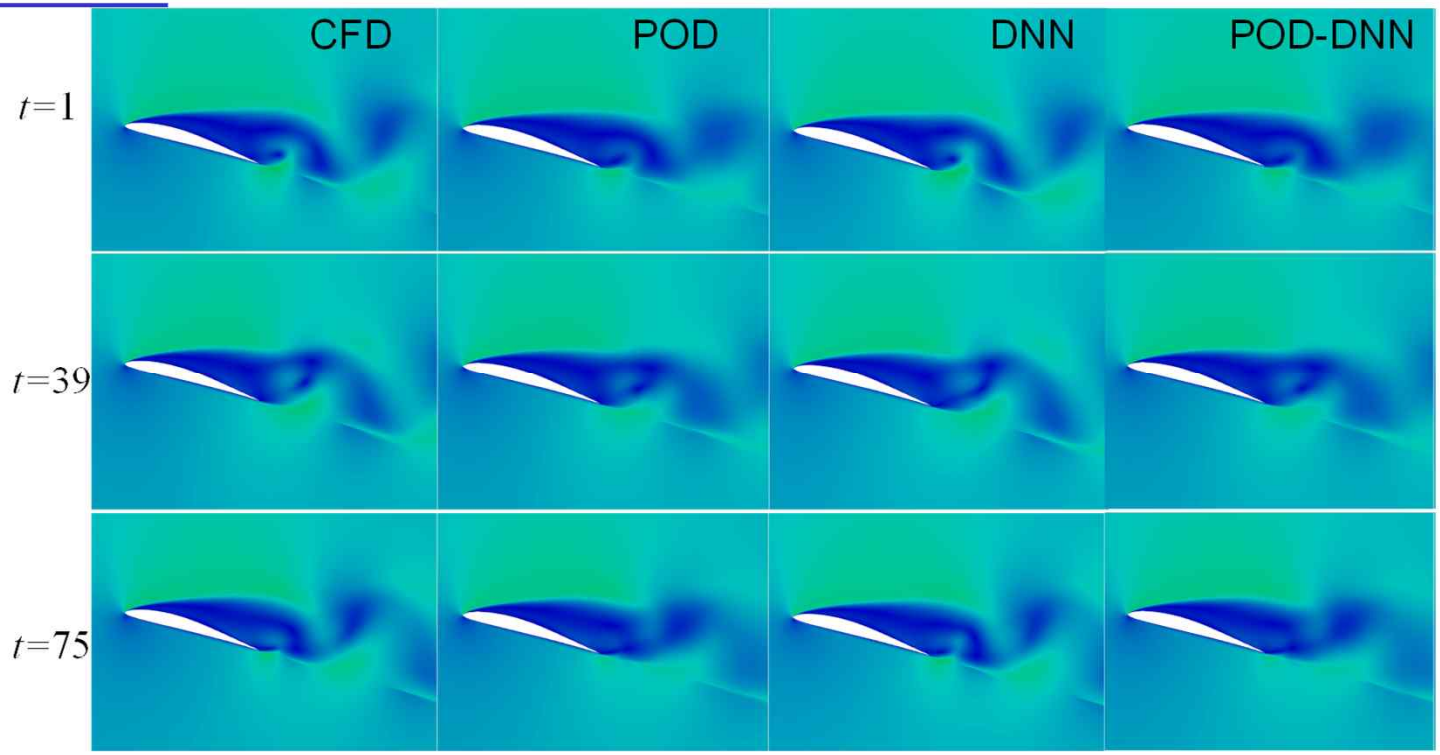
CLCD.py

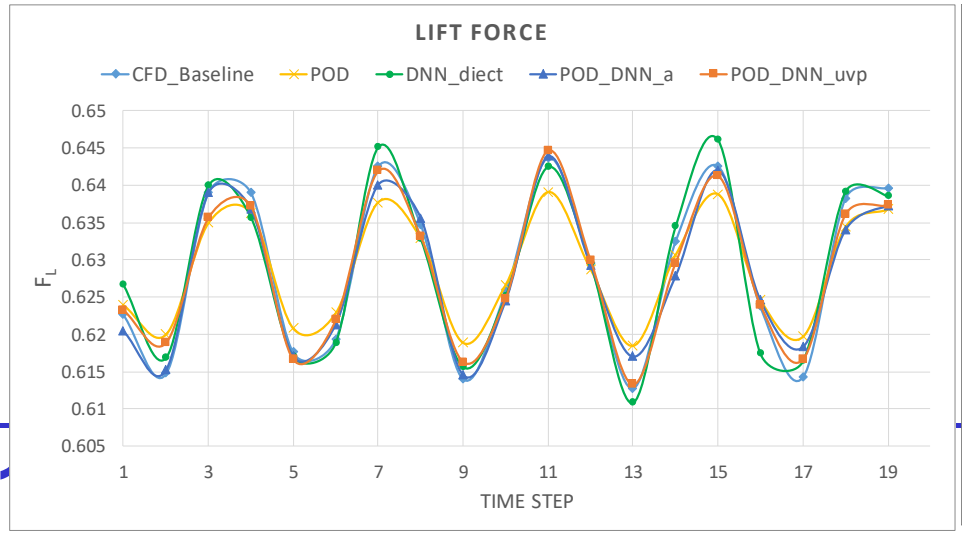
predict_drag_lift 함수

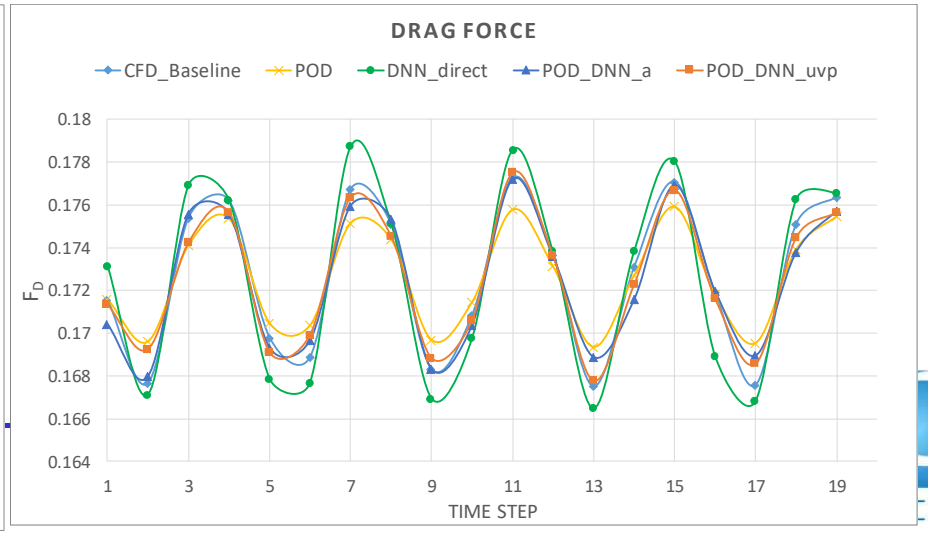
```
### verical and parallel vectors on the aifoil surface
tck, u = interpolate.splprep([xm,ym],k=3,s=0)
out = interpolate.splev(u,tck)
der = interpolate.splev(u,tck, der=1)
mag_der = np.sqrt(der[0]*der[0]+der[1]*der[1])
[nx,ny] = [-der[1]/mag_der,der[0]/mag_der]
ds = np.sqrt(np.square(x_sur[0:-1]-x_sur[1:])+np.square(y_sur[0:-1]-y_sur[1:]))
nx_star = np.tile(nx, (Tsur,1)).T
ny_star = np.tile(ny, (Tsur,1)).T
ds_star = np.tile(ds, (Tsur,1)).T
dn = np.sqrt(np.square(xm2-xm)+np.square(ym2-ym))
dn_star = np.tile(dn, (Tsur,1)).T

### integration
F_D = []
F_L = []
INT0 = (-P_star[0:Nsur-1,:]*nx_star[0:Nsur-1,:]*0 + vis[0:Nsur-1,:]*Re_inv*(3*(
INT1 = (-P_star[1:Nsur,:]*nx_star[0:Nsur-1,:]*0 + vis[1:Nsur,:]*Re_inv*(3*(u_su
F_D = np.append(F_D, 0.5*np.sum(INT0 + INT1, axis = 0)) # F_D = Csur x T

INT0 = (-P_star[0:Nsur-1,:]*ny_star[0:Nsur-1,:]*0 + vis[0:Nsur-1,:]*Re_inv*((u_
INT1 = (-P_star[1:Nsur,:]*ny_star[0:Nsur-1,:]*0 + vis[1:Nsur,:]*Re_inv*((u_sur2
F_L = np.append(F_L, 0.5*np.sum(INT0 + INT1, axis = 0)) # F_L = Csur x T


return F_L, F_D
```


KSCFE

DNN 결과

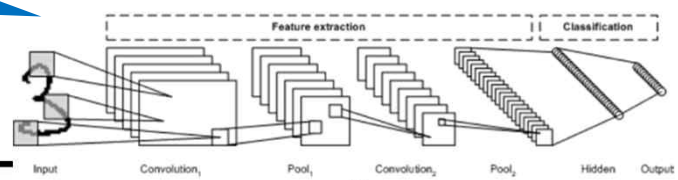

CFD
POD
DNN
POD-DNN
t=1
t=39
t=75


LIFT FORCE
CFD_Baseline
POD
DNN_diect
POD_DNN_a
POD_DNN_uvp
0.65
0.645
0.64
0.635
0.63
0.625
0.62
0.615
0.61
0.605
F_L
1
3
5
7
9
11
13
15
17
19
TIME STEP

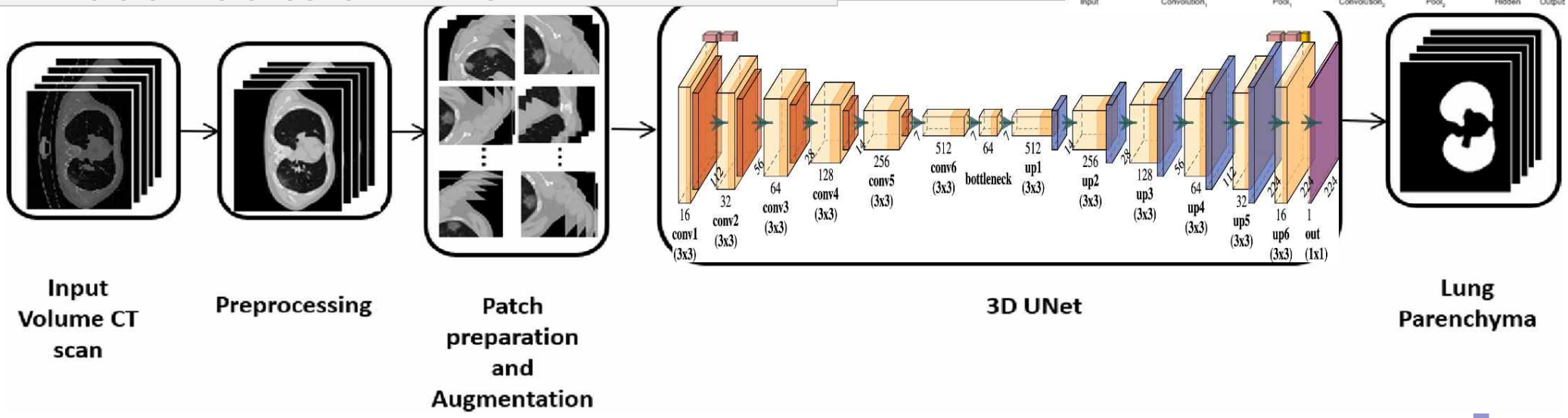

DRAG FORCE
CFD_Baseline
POD
DNN_direct
POD_DNN_a
POD_DNN_uvp
0.18
0.178
0.176
0.174
0.172
0.17
0.168
0.166
0.164
F_D
1
3
5
7
9
11
13
15
17
19
TIME STEP

[3] CNN 축소모형

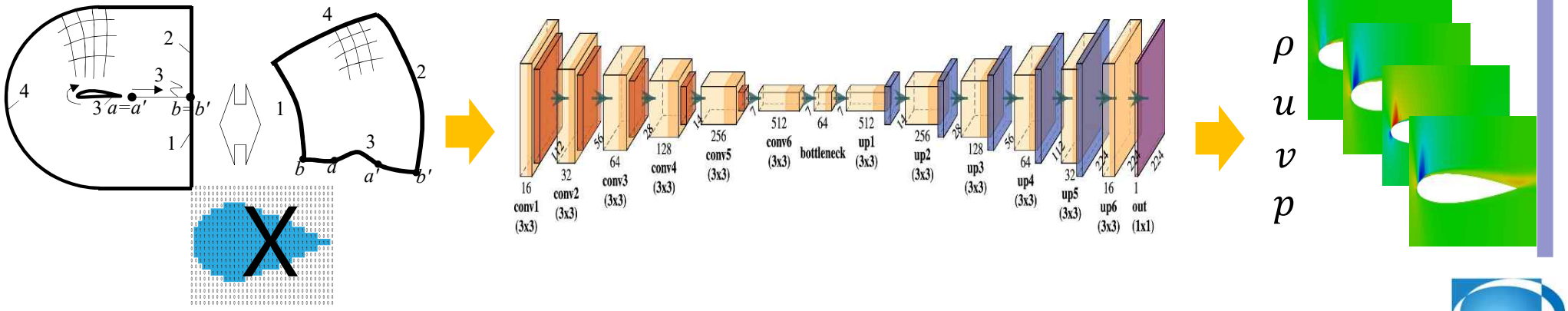
임의 형상



- CT 이미지 분석에 적용되는 CNN 기반 U-network

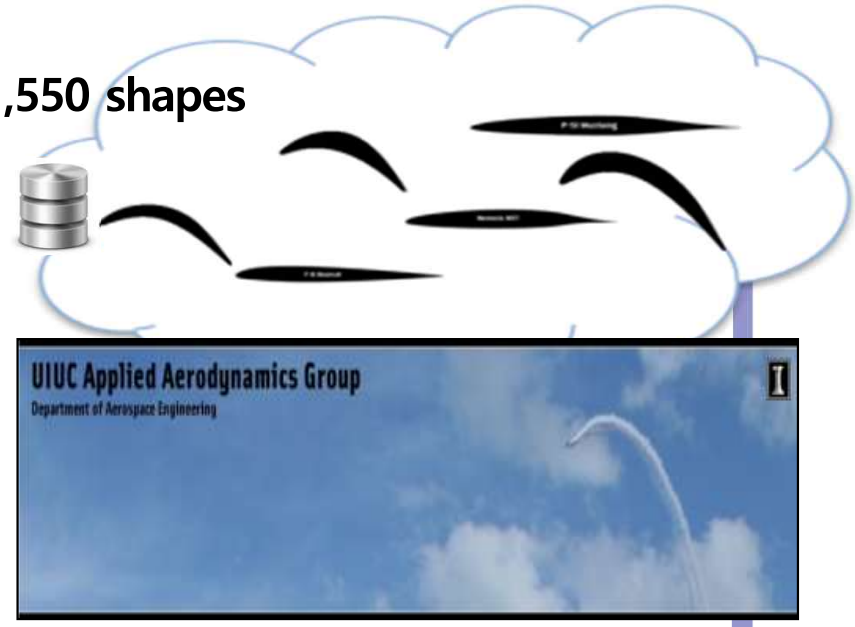


- 형상에 따른 유동장 고정밀 예측에 유사한 딥러닝 기법 채용



Unet_UIUC.py

```
import os
import time
import random
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, losses, optimizers
from tensorflow.keras.datasets import fashion_mnist
from tensorflow.keras.models import Model

from CFDLib import mean_squared_error, relative_error

# Data path
DATA_PATH = "C:\\SimData\\UIUC_ML\\array_foil_UIUC_cuttail0064.npz"

noConcernVar = 4
zone1_i = 401
zone1_j = 81
glayer = 64 #zone1_j
cuttail = 0
```



Unet_UIUC.py

in-out 변수 정의

```
# Model parameters
BATCH_SIZE = 1530
EPOCHS = 6000
IMAGE_SHAPE = [zone1_i, zone1_j]
```

```
XI_star = np.reshape(XC_star.T, [numd, glayer, zone1_i-2*cuttail])[0:BATCH_SIZE,:,:]
XI_field = XI_star[:,:,:-1]
YI_star = np.reshape(YC_star.T, [numd, glayer, zone1_i-2*cuttail])[0:BATCH_SIZE,:,:]
YI_field = YI_star[:,:,:-1]
XT_star = np.reshape(XC_star.T, [numd, glayer, zone1_i-2*cuttail])[BATCH_SIZE:numd,:,:]
XT_field = XT_star[:,:,:-1]
YT_star = np.reshape(YC_star.T, [numd, glayer, zone1_i-2*cuttail])[BATCH_SIZE:numd,:,:]
YT_field = YT_star[:,:,:-1]
UI_star = np.reshape(UC_star.T, [numd, glayer, zone1_i-2*cuttail])[0:BATCH_SIZE,:,:]
UC_field = UI_star[:,:,:-1]
VI_star = np.reshape(VC_star.T, [numd, glayer, zone1_i-2*cuttail])[0:BATCH_SIZE,:,:]
VC_field = VI_star[:,:,:-1]
PI_star = np.reshape(PC_star.T, [numd, glayer, zone1_i-2*cuttail])[0:BATCH_SIZE,:,:]
PC_field = PI_star[:,:,:-1]
UT_star = np.reshape(UC_star.T, [numd, glayer, zone1_i-2*cuttail])[BATCH_SIZE:numd,:,:]
UT_field = UT_star[:,:,:-1]
VT_star = np.reshape(VC_star.T, [numd, glayer, zone1_i-2*cuttail])[BATCH_SIZE:numd,:,:]
VT_field = VT_star[:,:,:-1]
PT_star = np.reshape(PC_star.T, [numd, glayer, zone1_i-2*cuttail])[BATCH_SIZE:numd,:,:]
PT_field = PT_star[:,:,:-1]
```

Unet_UIUC.py

Data loading on memory

```python
@tf.function
def dataloader(paths):
    dataset = tf.data.Dataset.from_tensor_slices(paths)
    dataset = dataset.batch(BATCH_SIZE)
    dataset = dataset.prefetch(1)
    return dataset

datasetI = dataloader(Input_field)
train_inp = []
for batch in datasetI.take(1):
    for i, img_inp in enumerate(batch):
        img_inp_np = img_inp.numpy()
        train_inp.insert(i, img_inp_np)
train_inp = np.array(train_inp, dtype="float32")
print(train_inp.shape)
```

```python
datasetO = dataloader(Field_star) #UC_field
train_env = []
for batch in datasetO.take(1):
    for i, img in enumerate(batch):
        img_np = img.numpy()
        train_env.insert(i, img_np)
train_env = np.array(train_env, dtype="float32")
print(train_env.shape)

datasetT = dataloader(Test_field)
train_test = []
for batch in datasetT.take(1):
    for i, img_test in enumerate(batch):
        img_test_np = img_test.numpy()
        train_test.insert(i, img_test_np)
train_test = np.array(train_test, dtype="float32")
```

Unet_UIUC.py

CNN network 정의

```
# Build CNN network
input_e = tf.keras.Input(shape=(glayer, zone1_i-2*cuttail-1, 2))
pooling_size = 2
n_ch = 12   #18
conv1 = layers.Conv2D(n_ch, (3,3), activation='elu', padding = 'same')(input_e)
mp1 = layers.MaxPooling2D((pooling_size,pooling_size))(conv1)

conv2 = layers.Conv2D(n_ch*2, (3,3), activation='elu', padding = 'same')(mp1)
mp2 = layers.MaxPooling2D((pooling_size,pooling_size))(conv2)

output_e = layers.Conv2D(n_ch*4, (3,3), activation='elu', padding = 'same')(mp2)

convt1 = layers.Conv2DTranspose(n_ch*2, (3,3), activation='elu', padding='same')(output_e)
upsamp1 = layers.UpSampling2D((pooling_size,pooling_size))(convt1)
skipcon1 = layers.Concatenate(axis=3)([conv2, upsamp1])
conv3 = layers.Conv2D(n_ch*2, (3,3), activation = 'elu', padding='same')(skipcon1)

convt2 = layers.Conv2DTranspose(n_ch*1, (3,3), activation='elu', padding='same')(conv3)
upsamp2 = layers.UpSampling2D((pooling_size,pooling_size))(convt2)
skipcon2 = layers.Concatenate(axis=3)([conv1, upsamp2])
conv4 = layers.Conv2D(n_ch*1, (3,3), activation = 'elu', padding='same')(skipcon2)

output_d = layers.Conv2DTranspose(3, (3,3), activation='elu', padding='same')(conv4)
```

Unet_UIUC.py

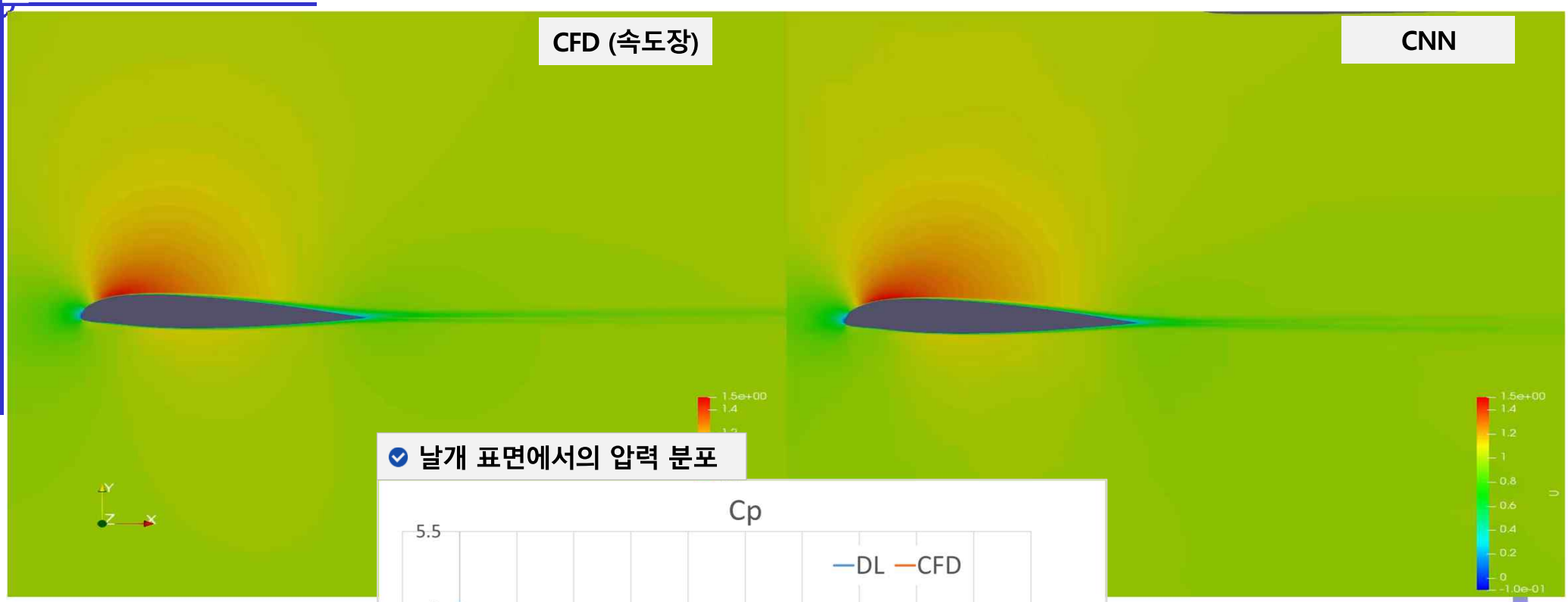
Custom loss 정의

```
### Loss
def custom_mse(idx_x_bd1, idx_x_bd2):

    def loss(y_true,y_pred):
        # Extract boundary values
        train_bd1 = y_pred[:,0:1,:,:][:,:,idx_tip[0]:1:-1,:]
        train_bd2 = y_pred[:,0:1,:,:][:,:,idx_tip[1]:idx_bottom[1],:]
        # calculating squared difference between target and predicted values
        loss1 = tf.keras.backend.square(y_true - y_pred) # (batch_size, 3)
        loss2 = tf.keras.backend.square(train_bd1 - train_bd2)
        # summing both loss values along batch dimension
        loss1 = tf.keras.backend.mean(tf.keras.backend.sum(loss1, axis=1))
        loss2 = tf.keras.backend.mean(tf.keras.backend.sum(loss2, axis=1))
        return loss1 + loss2

    return loss
```

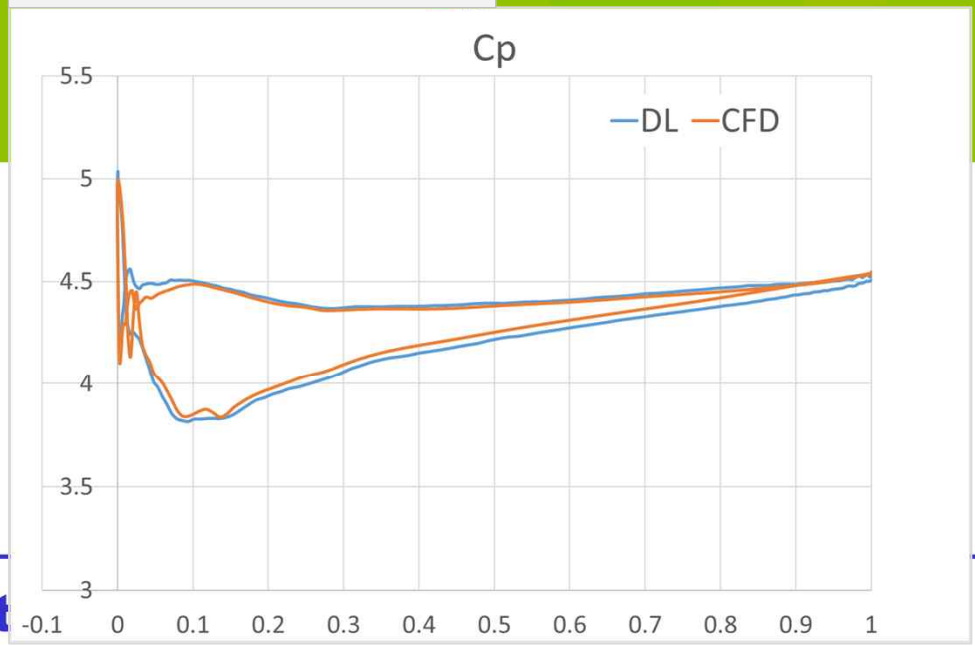
Unet_UIUC.py

```
# Train or Infer
unet = Model(inputs=input_e, outputs=output_d)
unet.compile(optimizer='adam', loss=custom_mse(idx_x_bd1,idx_x_bd2), metrics=["mse"])
#loss=losses.MeanSquaredError(), custom_mse(train_env, output_d, idx_x_bd1, idx_x_bd2)
unet.fit(train_inp, train_env, validation_split=0.1, epochs=3000, verbose=2)
unet.save('my_model.h5')
'''
# Load saved model from checkpoint directory for inference
unet = tf.keras.models.load_model('my_model.h5', compile=False)
'''
decoded_imgs = unet([train_test]).numpy() #,time_test
```

CNN 결과



날개 표면에서의 압력 분포



결 론

Conclusions

- 본 강좌에서 **CFD-AI** 튜토리얼 문제 및 코드에 관해 설명하였음

  → **Repository** 참고하여 실습 권고

- **AI**에는 다양한 기법들이 존재하고 다양한 적용처가 존재함

  → 본 실습에서는 익형의 **CFD** 축소모형 구축에 관한 것임

- 문의: **shandy77@kisti.re.kr**

경청해 주셔서
감사합니다.