계층적 데이터 포맷을 이용한 공력 데이터 베이스 활용성 증대

Improvement of Aerodynamic Database Usability using Hierarchical Data Format

최대산*, 김성태

㈜넥스트폼 기술연구소

목차

Ⅰ. 연구배경

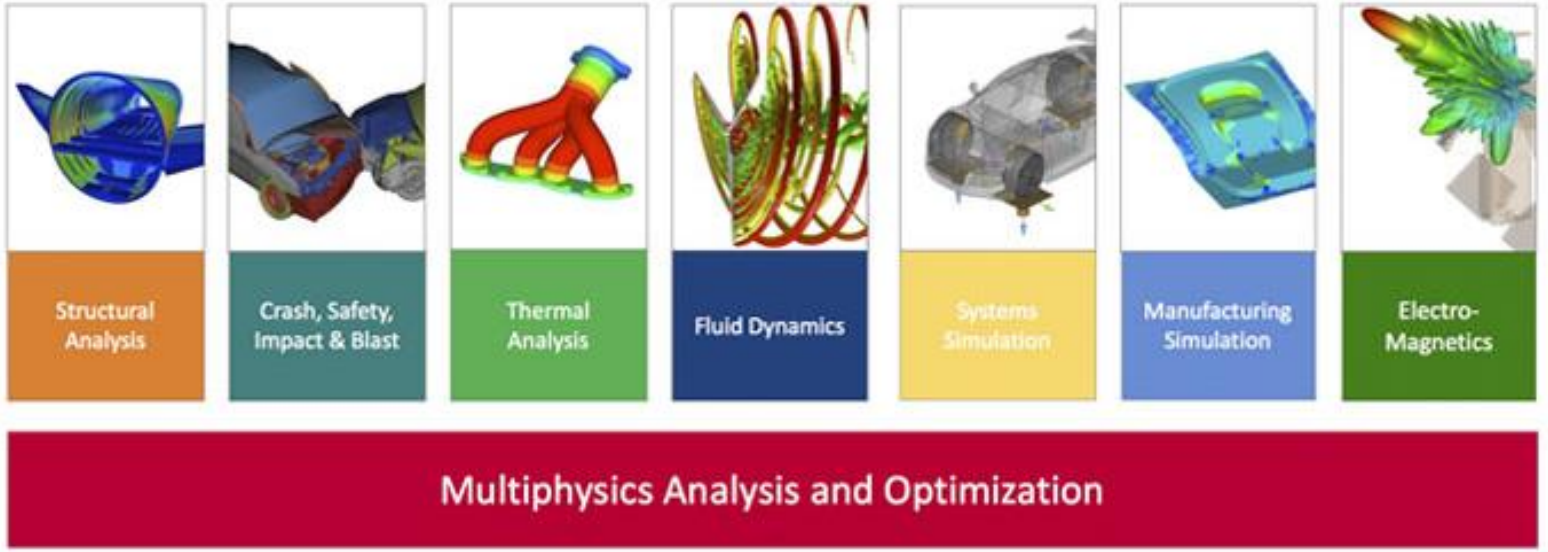
1. CFD 연구 환경의 변화
2. 공력 데이터 베이스 구축 연구
3. 연구 필요성 및 연구 목표

NEXTfoam open source CFD consulting

연구배경

- **CFD 연구 환경의 변화**
  - **기존에는 단일 연구 항목 / 단일 플랫폼에서 연구가 주로 수행**
  - **복합 물리 현상 해석 / 고도화된 유체 해석 / 다양한 연구 단계**
    - 유체해석 + 동역학 / 구조 / 전자기장 / 열전달 / 화학반응, 다상유동, 소음해석
    - 수치 시뮬레이션 / 설계 / 최적화 / 데이터마이닝 / 대체모델 / 기계학습

  - **CFD와 EFD 결합**
    - 수치 시뮬레이션 / 풍동 시험 / 비행 시험



<다양한 수치 시뮬레이션 및 활용 [1]>

연구배경

- **CFD 연구 환경의 변화**
  - **다양한 해석 도구 / 연구 플랫폼이 결합**
    - Salome / SolidWorks / CATIA / AutoCAD
    - Gmesh / Netgen / Paraview / ANSA / SnappyMesh
    - FAMUS / Ansys Fluent / STAR-CCM+ / OpenFOAM / COMSOL Multiphysics / In-house
    - Tecplot / Paraview
  - **다양한 계산 플랫폼**
    - Linux OS / Windows / MacOS
    - 서버의 경우 OS 버전 / 라이브러리 버전 상이한 경우 빈번
  - **수 백 케이스 이상의 대규모 해석 프로젝트가 빈번하게 나타남**

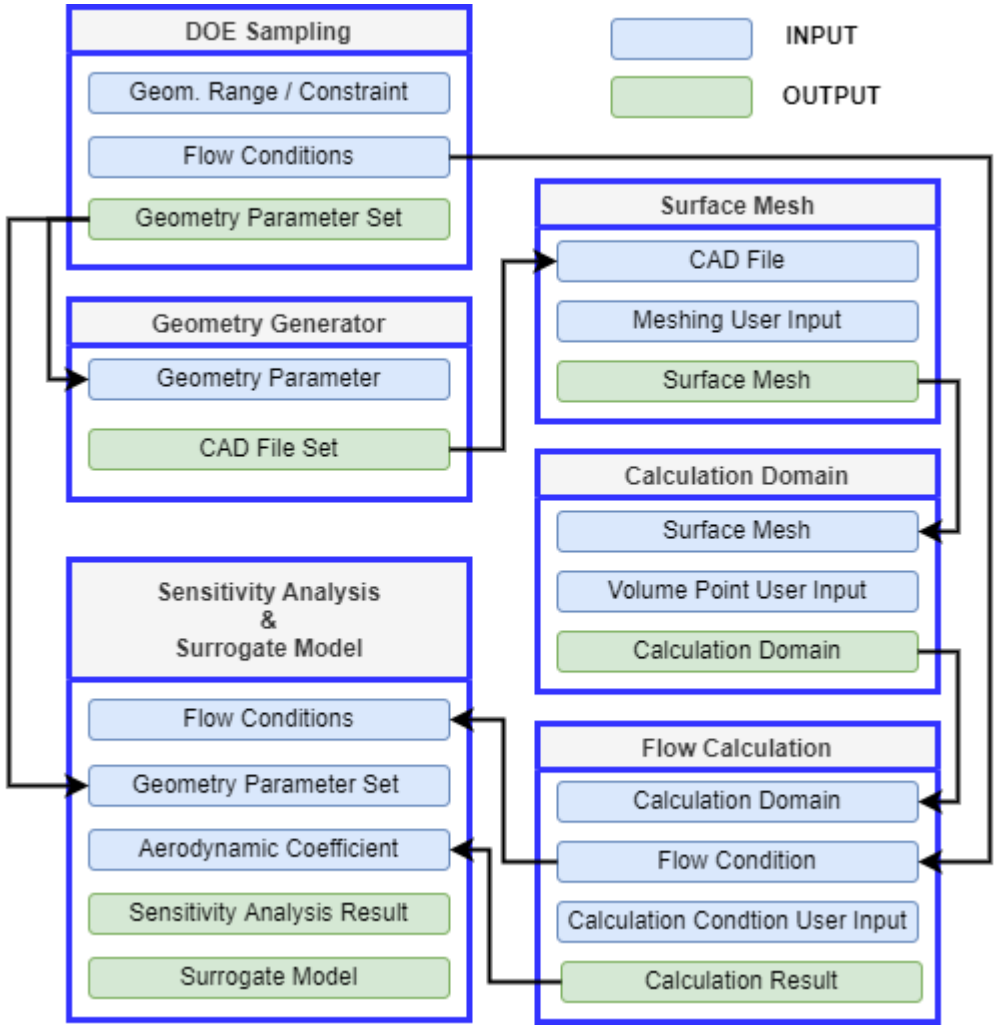
연구배경

- **프로젝트 데이터 관리 중요성 증대**
  - **프로젝트 / 데이터 정보 메타 데이터**
  - **복잡해진 각 연구 항목들의 관계 정리**
  - **대규모 데이터 관리 / 호환성 / 휴대성**

- **기존의 OS 제공 파일 관리 시스템, 프로그램 제공 I/O 의 한계**
  - **각각의 플랫폼 / 계산환경 마다 상이한 포맷과 데이터 저장 방식**
  - **데이터들 간의 연결성을 구조적으로 보여주기 어려움**
  - **공유 및 협업 차원**
    - OS에 따라 호환의 어려움: 리눅스 기반 OS, 윈도우, 맥 with various version
    - 연구자간 공유 : 대용량, 전송 속도의 한계
  - **상용 툴 제공 기능의 한계 (ex. Tecplot Chorus)**
    - 디자인 공간 분석 툴
    - 수 백 케이스 / 이종의 데이터를 처리하기에는 한계
    - 지원하지 않는 연구 도구
    - 유료

연구배경

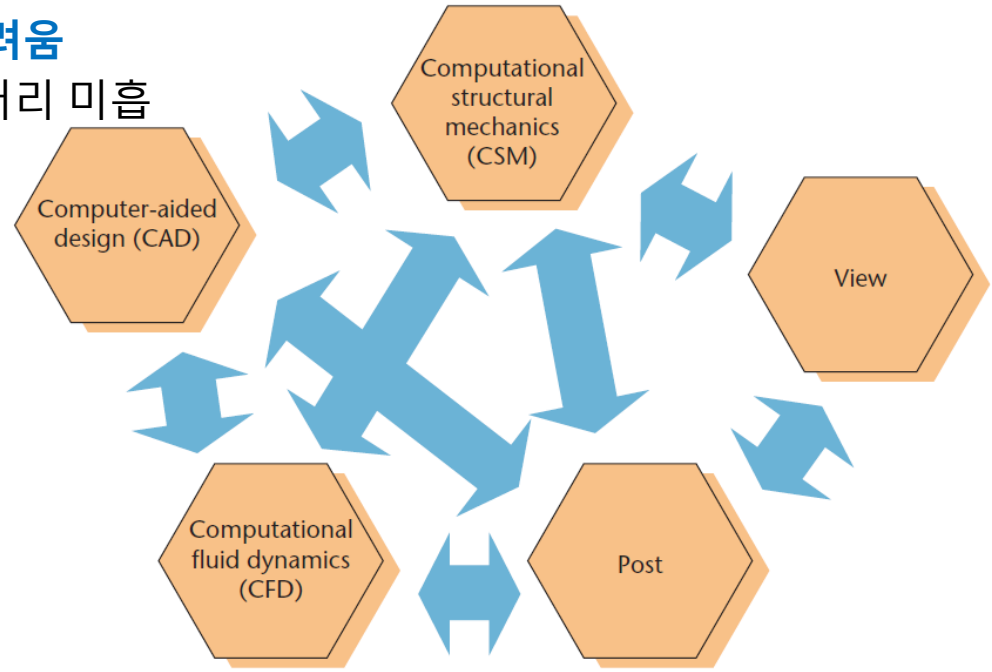
- **공력 데이터 베이스 구축 연구**
  - **다양한 연구 항목**
    - DOE 기반 샘플링 (수 백 케이스 이상)
    - 파라메트릭 자동 CAD 파일 생성
    - 파라메트릭 자동 해석 도메인 생성
    - 유동 해석
    - 대체모델 생성 / 평가 (공력 DB 생성)
    - 민감도 분석
  - **각 연구 항목 별 상이한 연구 플랫폼**
    - Python library (DOE, 전후처리, 분석)
    - Salome (CAD)
    - FAMUS (CFD solver)
    - MEDOC (Kriging model)
  - **각 연구 플랫폼 간 데이터 교환**
    - 각 연구 플랫폼 상호 종속적 관계



<공력 DB 구축 연구의 항목 간 데이터 흐름도>

연구배경

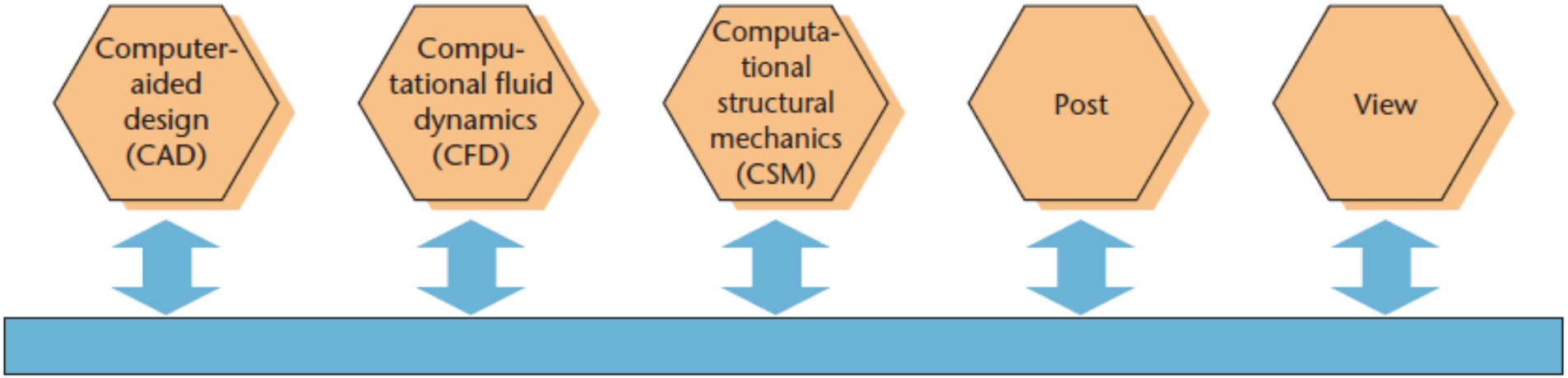
- **공력 데이터 베이스 구축 연구**
  - **기존의 방안**
    - xml / sheet 형식으로 메타 데이터 정리
    - 각 연구 결과 및 입력 값 windows file system 기반 정리
    - **사용자는 메타 데이터를 통해 전반적인 데이터를 검토 → 필요시 주요 결과값을 오픈**
  - **기존의 데이터 구축 방식은 한계가 명확**
    - **방대한 양의 데이터 확인 및 공유의 어려움**
    - 계층/종속적으로 구성된 데이터 파일 처리 미흡
    - **중복 데이터 → 저장공간 비효율**
    - **이질적 데이터 처리 미흡**
      (다양한 포맷, 상이한 실행 환경 등)



<복잡한 모듈 간 인터페이스, 컨버터 필수적 [2]>

연구배경

- 공력 데이터 베이스 구축 연구
  - 공력 데이터 베이스 및 연구 결과의 관리 및 활용성 증대 필요
    - 방대한 양의 데이터 신속하게 확인
    - 데이터 간 종속성 / 계층구조 처리
    - 저장공간 비효율 제거
    - 이질적 데이터를 하나의 플랫폼에서 확인



<통일된 데이터 / 인터페이스 포맷, 컨버터 불필요 [2]>

NEXTfoam open source CFD consulting

연구배경

- 연구 필요성
  - DOE / CAD / CFD / Surrogate model / Sensitivity analysis 가 결합된 공력 DB 구축 과제 수행
  - 상기 과제 특성 상, 수 많은 이질적 연구 데이터가 종속적 / 계층적으로 구성
  - 기존 I/O 및 파일 시스템 기반 단순 데이터 파일 저장
    → 활용성 / 저장공간 등 극심한 비효율 발생

- 연구 목표
  - Python 언어 및 HDF 라이브러리를 활용
  - 효율적인 프로젝트 관리 / 공력 DB 구축 데이터 저장 방법 도출
  - 공력 DB 활용성 및 효율성 향상

Ⅱ. 연구방법

1. 계층적 데이터 포맷 - HDF

2. HDF 핵심 기능

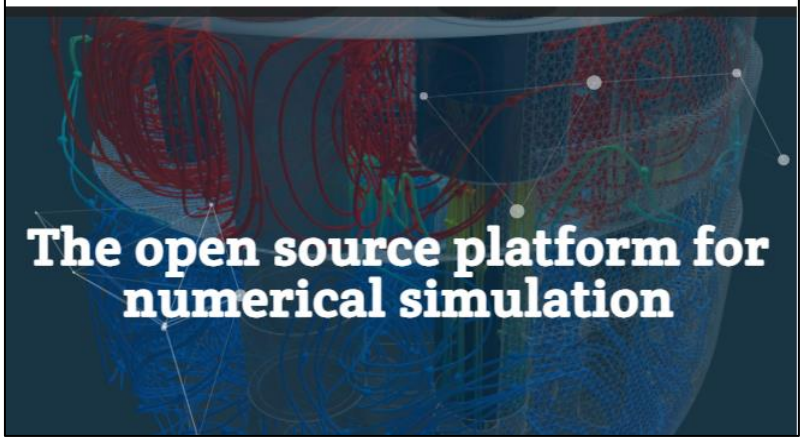
계층적 데이터 포맷 - HDF

- 과학적 데이터 처리
  - 기존에는 기본적인 수준의 파일 관리 시스템 / 파일 입출력 기능 활용
  - 점차 복잡하고 방대한 데이터를 효과적으로 처리하고 관리 능력 요구
    - CGNS (CFD General Notation System): CFD 데이터 규격 + 규격 관련 오픈 소프트웨어
    - HDF 활용 데이터 처리 및 관리: Fluent(2020R1 이후), Mathlab, Salome, Paraview 등

CGNS

CFD General Notation System
Switch to HDF5

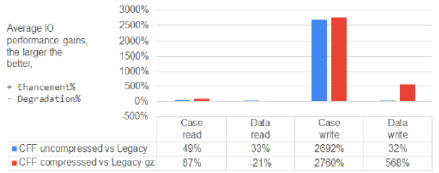
SALOME



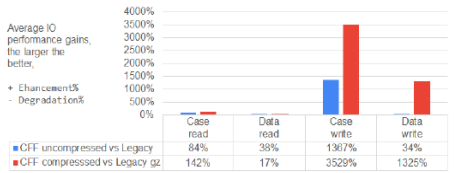
Other Migration Considerations

- Case/Data files will now be written in the *.cas.h5 / *.dat.h5 (CFF) format by default, which offers much better performance, esp. for large cases in parallel
  - Benchmark results averaged over cases 4M to 150M cells and 40 cores to 1280 cores
    - Case write performance average of ~10x to ~20x faster than legacy
    - Case read performance ~50% - ~100% improvement
    - Data write average ~5x faster for compressed files, ~30% faster for uncompressed files
    - Data read ~15% - ~30% faster, except for small compressed cases where some degradation ~20% is observed.

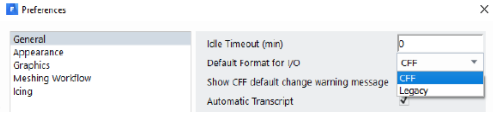
Average IO performance gains
4 to 140 million cells, 6 cases, t40~t1280



Average IO performance gains for the larger cases
71 to 140 million cells, 2 cases, t160~t1280



User Preference available to revert to legacy format if desired

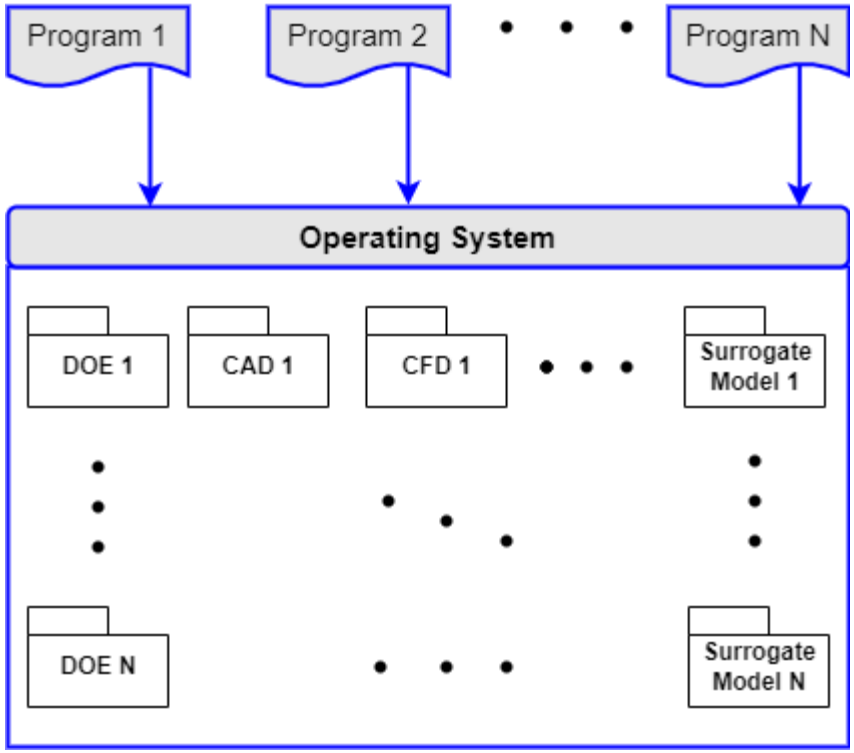


64 | © 2019 ANSYS, Inc. ANSYS Confidential ANSYS

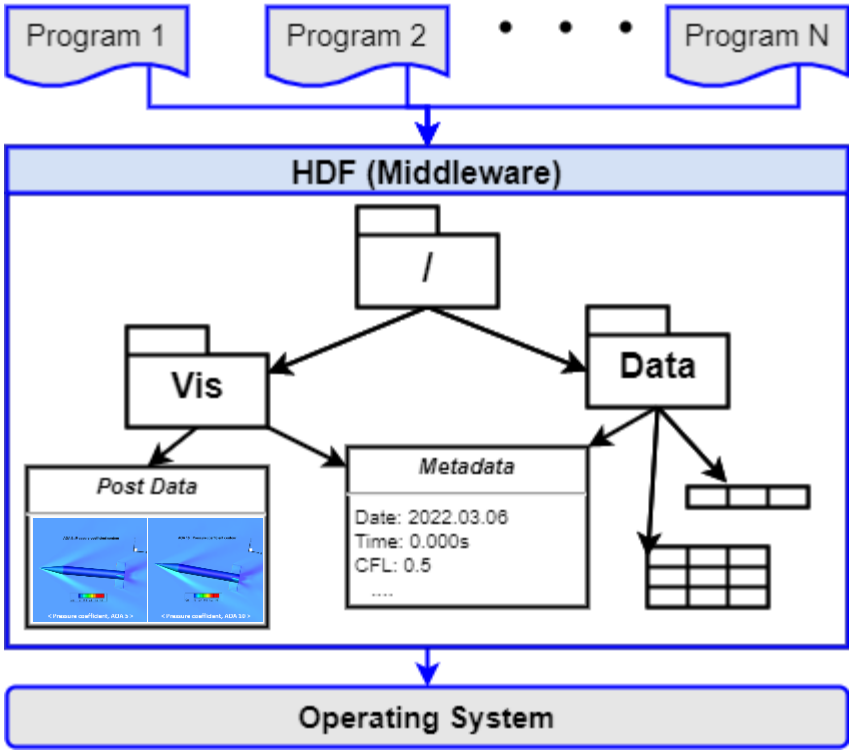
<HDF 적용 사례 [3], [4], [5]>

계층적 데이터 포맷 - HDF

- **HDF (Hierarchical Data Format)**
  - **과학적 데이터 저장을 위한 표준 계층적 파일 포맷**
  - **프로그램과 OS의 파일 시스템을 연결하는 수치해석 전용 미들웨어**
    - 일반적인 DB 관리 시스템은 정보 전송, 색인, 검색에 초점



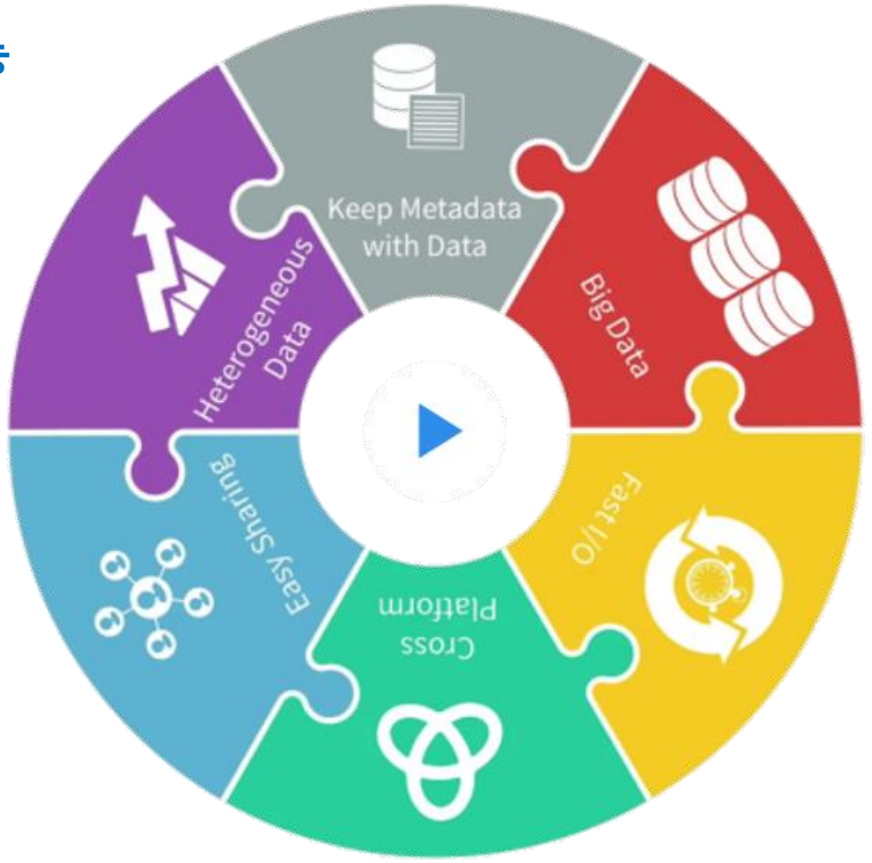
<기본 I/O 및 파일 시스템으로 구축된 데이터 구조>



<HDF5 포맷으로 구축된 결과파일 구조>

계층적 데이터 포맷 - HDF

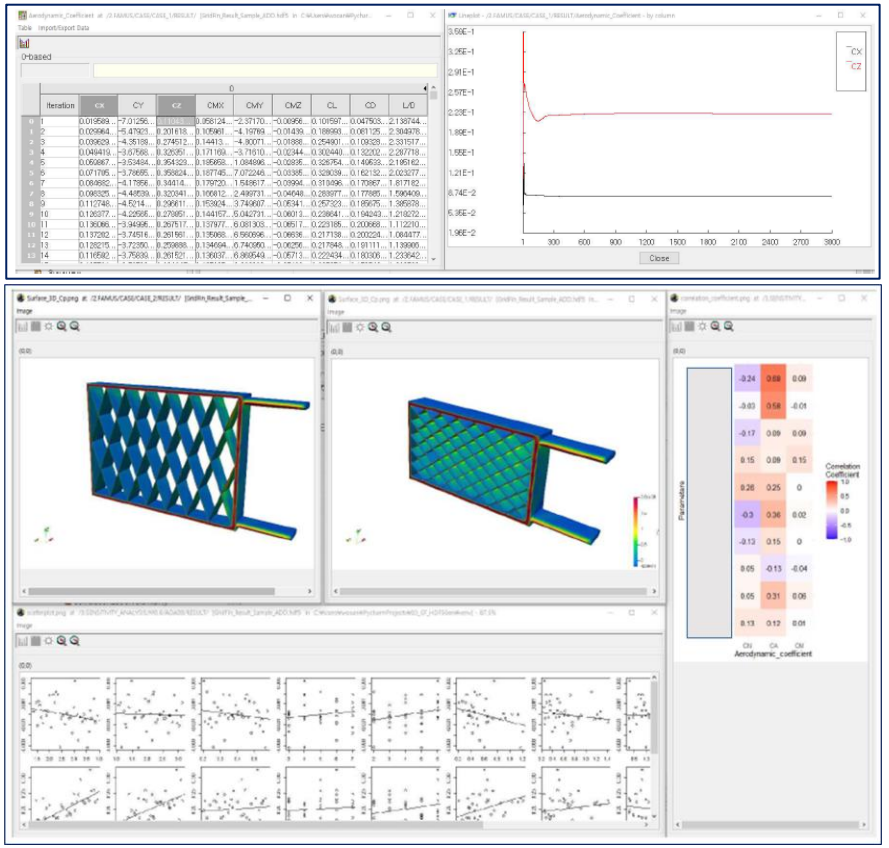
- **HDF 특징**
  - **자기 기술적 / 계층적 구성 / 이질적 데이터 결합 용이**
  - **데이터 처리**
    - **데이터 중복 저장을 방지하는 데이터 링크 기능**
    - 방대한 데이터 처리 용이 / 압축 기능
  - **활용성**
    - CFD 포함 여러 분야에서 인증되고 널리 사용
    - 다양한 프로그래밍 언어 API / 프로그램 지원
    - C, Fortran, Java, R, Python
    - Matlab, Tecplot, Paraview
  - **호환성 / 휴대성**
    - OS 독립적 / 하나의 파일로 관리 가능



<HDF5 특징 [6]>

계층적 데이터 포맷 - HDF

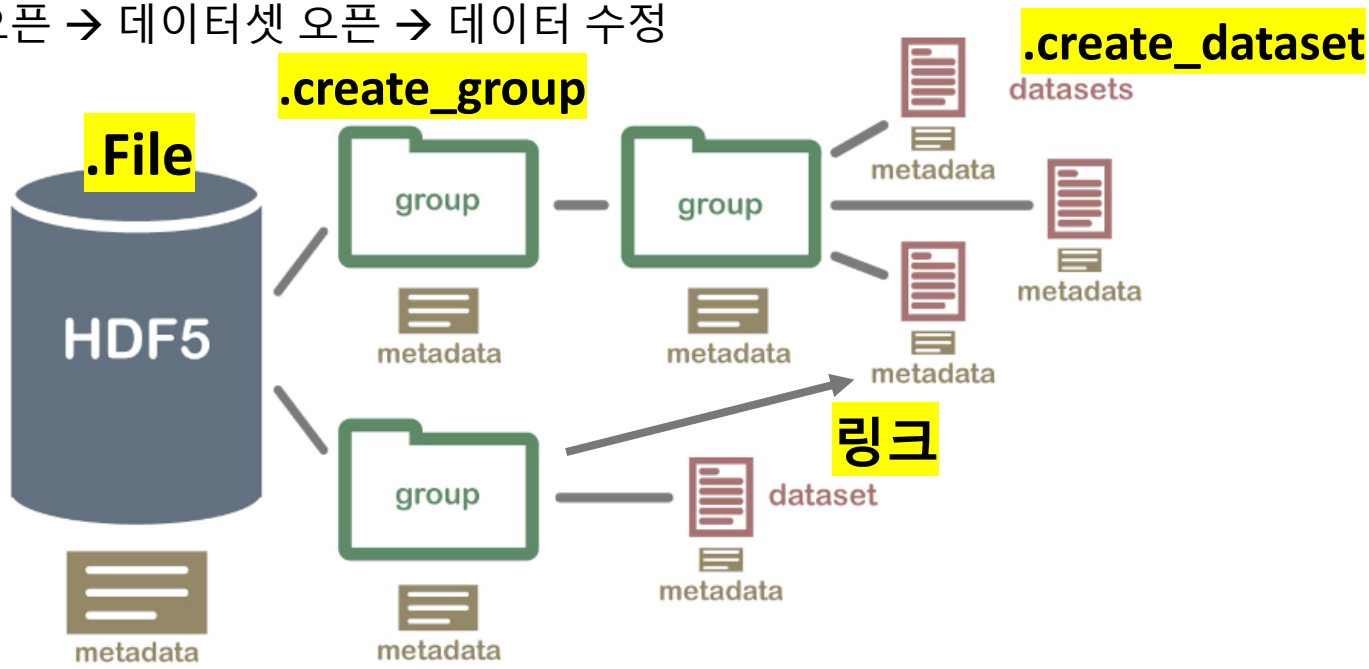
- **HDF 포맷 샘플 파일**
  - **각 연구 항목 별 데이터가 계층적으로 구성**
    - 하나의 플랫폼 안에서 주요 정보 확인
  - **데이터 오브젝트 공유 (링크)를 통한 저장공간 효율화**



<HDF5 포맷 결과 샘플 파일>

HDF 핵심 기능

- 개요
  - 탑-다운 방식
  - UNIX 파일 시스템과 유사
  - 데이터 생성
    - 파일 생성 → (그룹 생성 → 데이터스페이스 생성 →) 데이터셋 생성
  - 데이터 읽기
    - 파일 오픈 → 데이터셋 오픈 → 데이터 수정



<HDF 기본 구조 [7] 및 주요 함수>

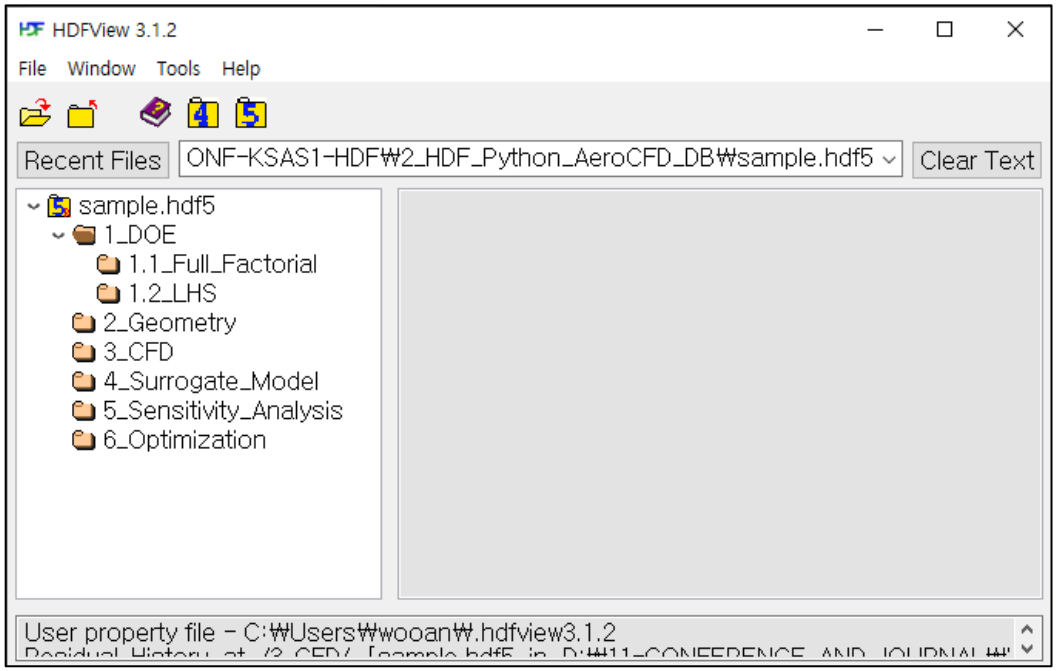
HDF 핵심 기능

▪ 핵심 기능 예제 (Python API)

핵심 기능	주요 문법
라이브러리	import h5py
파일 생성	hdf_file = h5py.File(경로, 'w')
그룹 생성	group = {parent}.create_group('그룹 이름')

```
import h5py                    # v3.2.1 or later

# 1. HDF 파일 생성
fileLocation = "D:\\11-CONFERENCE_AND_JOURNAL\\W2022
 -CONF-KSAS1-HDF\\2_HDF_Python_AeroCFD_DB\\sample.hdf5"
hdf_file = h5py.File(fileLocation, 'w')


# 2. 데이터 그룹 생성 (like 폴더)
g1 = hdf_file.create_group("1_DOE")
g11 = g1.create_group("1.1_Full_Factorial")
g12 = g1.create_group("1.2_LHS")
g2 = hdf_file.create_group("2_Geometry")
g3 = hdf_file.create_group("3_CFD")
g4 = hdf_file.create_group("4_Surrogate_Model")
g5 = hdf_file.create_group("5_Sensitivity_Analysis")
g6 = hdf_file.create_group("6_Optimization")
```

<샘플 코드 – HDF 파일 생성 / 데이터 그룹 생성>



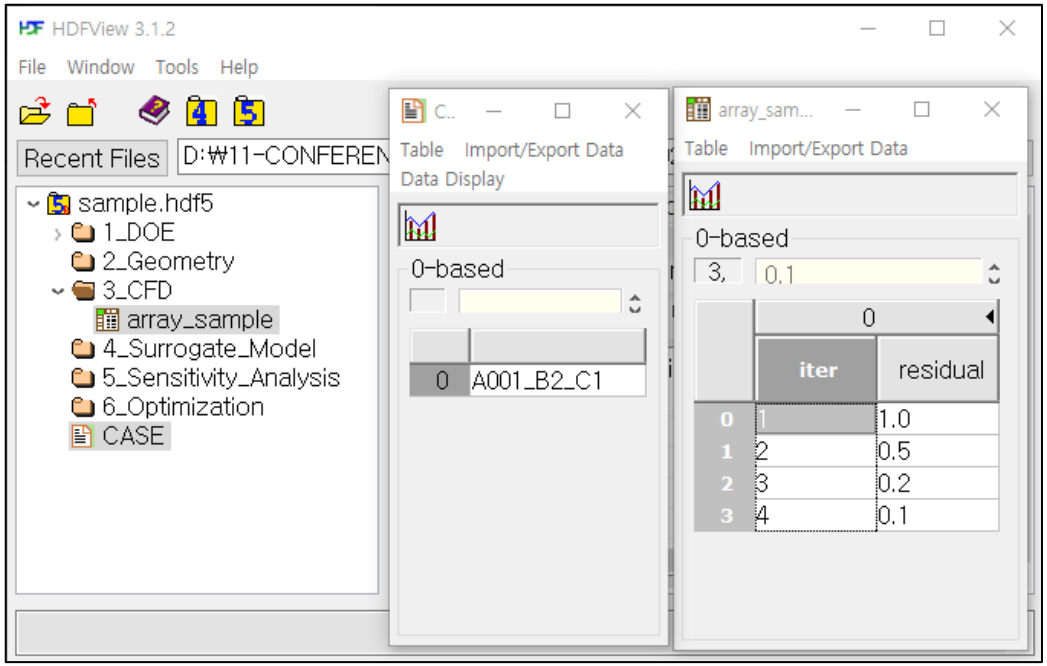
<HDF5 포맷으로 구축된 결과파일 구조>

▪ 핵심 기능 예제 (Python API)

핵심 기능	주요 문법
문자열 생성	str = {parent}.create_dataset('데이터셋 이름', data="문자열")
어레이 생성	arr = {parent}.create_dataset('데이터셋 이름', data='어레이')

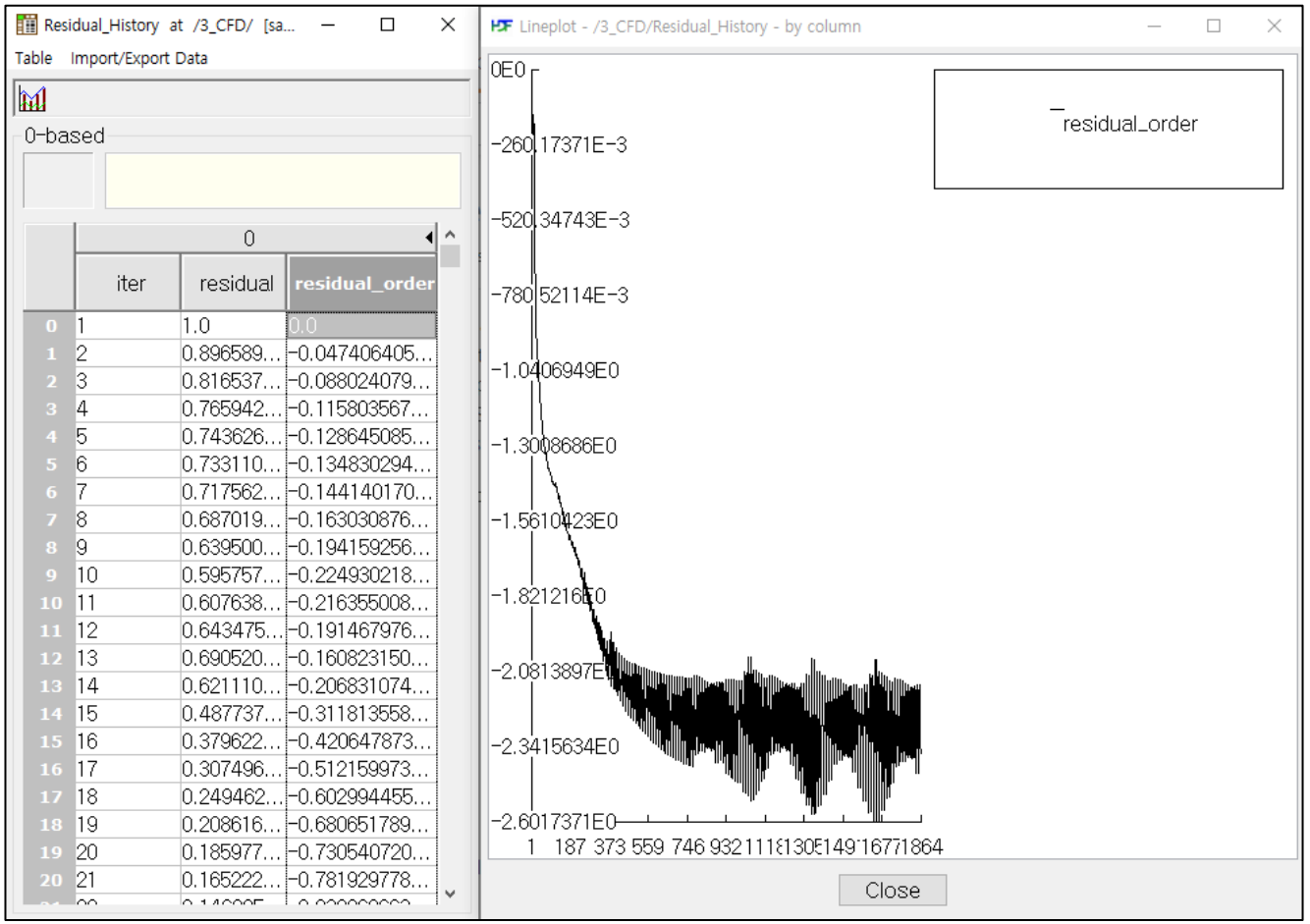
```
# 3. 데이터셋 생성 (like 파일)

## 3.1. 문자열 데이터셋 생성
str = hdf_file.create_dataset('CASE', data="A001_B2_C1")

## 3.2. 어레이 데이터셋 생성
chartData = [ [1, 1.0],
              [2, 0.5],
              [3, 0.2],
              [4, 0.1] ]
list2D_tuple = [tuple(x) for x in chartData]
dtype = np.dtype({'names':['iter','residual'],
 'formats':['i4','f8']})
arr = np.array(list2D_tuple, dtype=dtype)

g3.create_dataset('array_sample', data=arr)
```

<샘플 코드 – 문자열 및 어레이 데이터셋 생성>



<생성된 데이터셋 확인>

- 핵심 기능 예제 (Python API)
  - 2D 어레이 데이터셋 예시 – Residual History

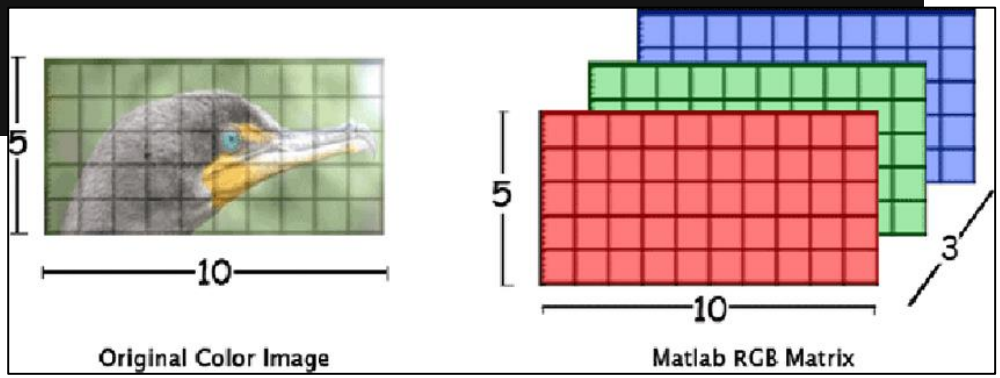


<저장된 2D array 데이터셋 / 간단한 후처리 기능>

▪ 핵심 기능 예제 (Python API)

핵심 기능	주요 문법
이미지 생성	img = {parent}.create_dataset('데이터셋 이름', data="RGB array", dtype='uint8')

```
## 3.4. 이미지 데이터셋 생성
imagePate = 'D:\\11-CONFERENCE_AND_JOURNAL\\W2022-CONF-KSAS1-HDF\\2_HDF_Python_AeroCFD_DB\\RESOURCE\\Image_Sample.png'
img = Image.open(imagePate)
img = img.convert("RGB") # 색상이 있는 RGB로 변환
data = np.asarray((img), dtype="uint8")
g3.create_dataset('Image_Sample', data=data, dtype='uint8', compression='gzip', compression_opts=9)
dset = g3.get('Image_Sample')
dset.attrs['CLASS'] = np.string_('IMAGE')
dset.attrs['IMAGE_VERSION'] = np.string_('1.2')
arr = np.asarray([0, 255], dtype=np.uint8)
dset.attrs['IMAGE_MINMAXRANGE'] = list(arr)
dset.attrs['IMAGE_SUBCLASS'] = np.string_('IMAGE_TRUECOLOR')
dset.attrs['INTERLACE_MODE'] = np.string_('INTERLACE_PIXEL')
```

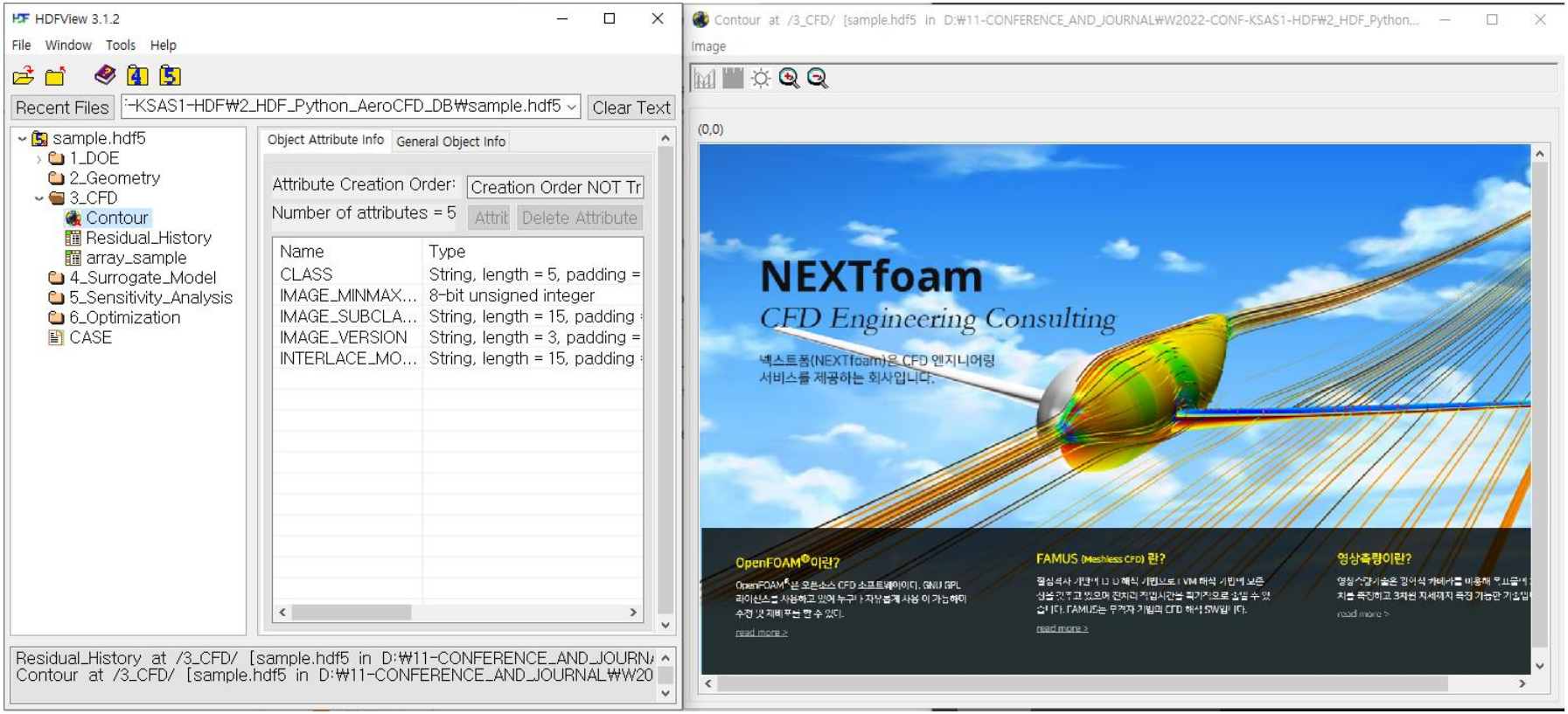


<이미지 데이터셋 생성 코드와 이미지 파일 저장 방식 [8]>

NEXTfoam open source CFD consulting

▪ 핵심 기능 예제 (Python API)

핵심 기능	주요 문법
이미지 생성	img = {parent}.create_dataset('데이터셋 이름', data="RGB array", dtype='uint8')



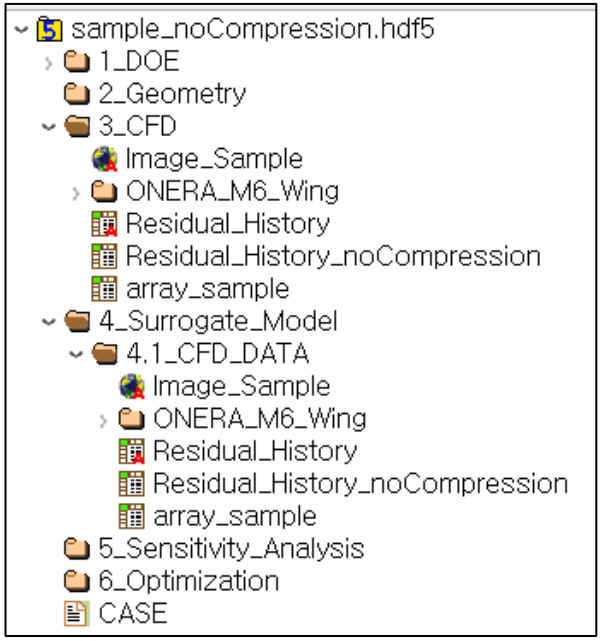
<생성된 이미지 데이터셋 [9]>

HDF 핵심 기능

- 기타 유용한 기능
  - 링크
    - 그룹 / 데이터셋 오브젝트에 직접 할당하는 방식

```
## 데이터 링크
g3 = f.create_group("3_CFD")

g4['4.1_CFD_DATA'] = g3 # g3 데이터셋 오브젝트를 공유
```

  - 압축
    - gzip 기반 압축 기능 제공
    - 각 데이터셋 별 압축 여부 / 압축 수준 선택 가능




```
## 3.3.2. 레지듀얼 데이터셋 생성
list2D_tuple = [tuple(x) for x in residualData]
dtype = np.dtype({'names':['iter','residual','residual_order'], 'formats':['i4','f8','f8']})
arr = np.array(list2D_tuple, dtype=dtype)
# Residual = g3.create_dataset('Residual_History', data=arr)
Residual = g3.create_dataset('Residual_History', data=arr, compression='gzip', compression_opts=9)
```

Ⅲ. 연구 결과

1. HDF 데이터 베이스 생성 코드

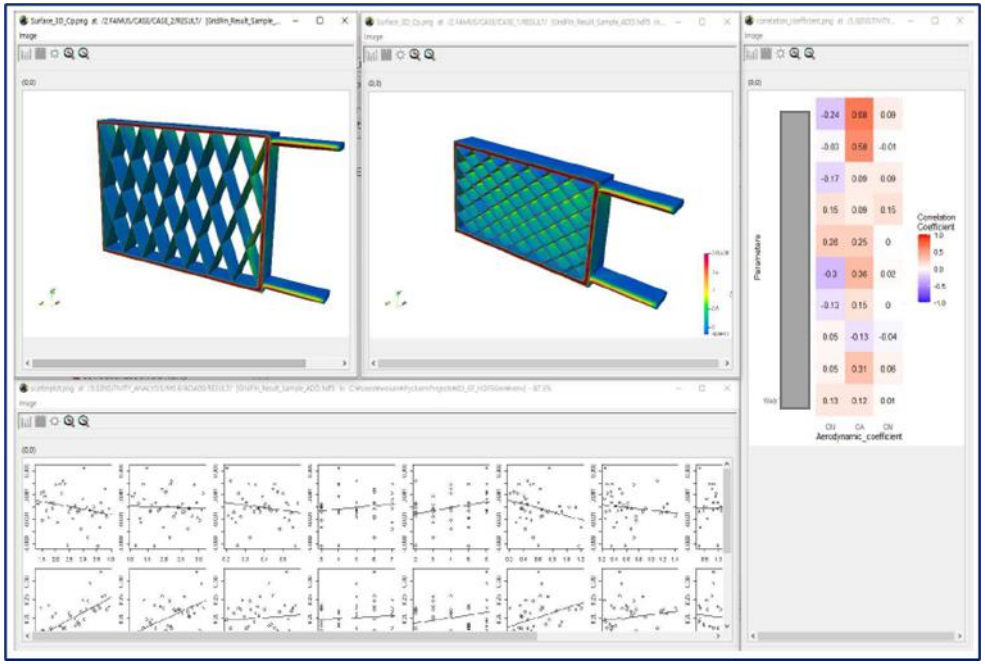
2. 구축 결과

HDF 데이터 베이스 생성 코드

- **DB 구축 방법 연구**
  - **파이썬 언어 기반 스크립트**
    - 입력값: DB 데이터 root 경로 / 각 데이터 별 처리 여부
    - 출력값: **주요 결과 추출 및 정리 / HDF5 DB 파일 생성**
  - **주요 라이브러리**
    - h5py, numpy, natsort, pillow

```
class USER_INPUT:
    """
    HDF DB 구축에 필요한 사용자 입력
    """
    def __init__(self):

        # Parameter level
        self.root_data_folder  =
         'D:\\12-PROJECT\\W2020-01-Grid_Fin_DB\\20220111_DB_HDF_builder\\DB_sample\\'
        self.HDF_output_folder =
         'D:\\12-PROJECT\\W2020-01-Grid_Fin_DB\\20220111_DB_HDF_builder\\'
        self.HDF_output_name   = 'GridFin_DB'

        self.OnOff_sampling    = True
        self.OnOff_case        = True
        self.OnOff_kriging     = True
        self.OnOff_sensitivity = False
```

<사용자 입력값>

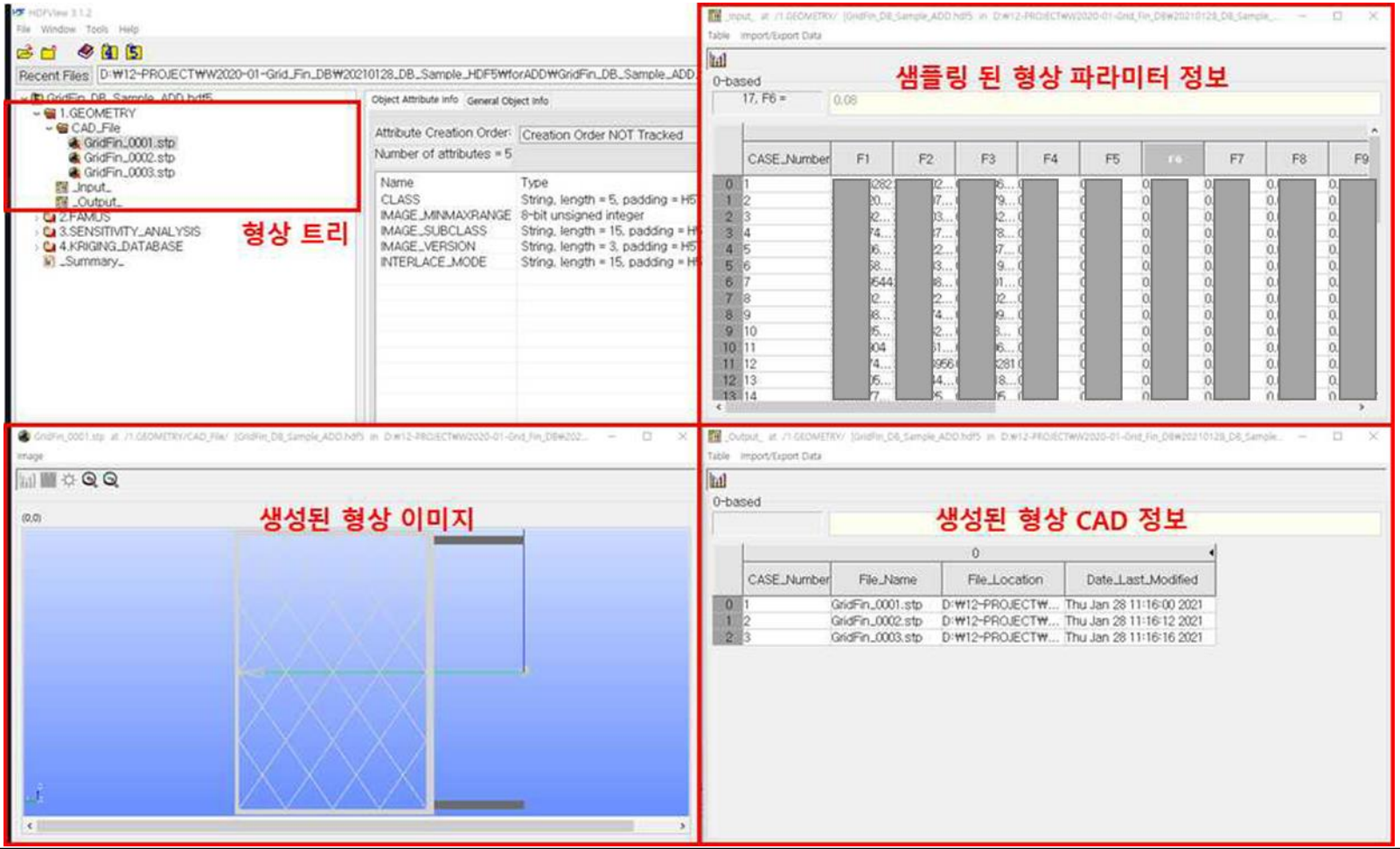


<HDF5 viewer 이용, 단일 플랫폼에서 확인 가능한 결과>

NEXTfoam open source CFD consulting

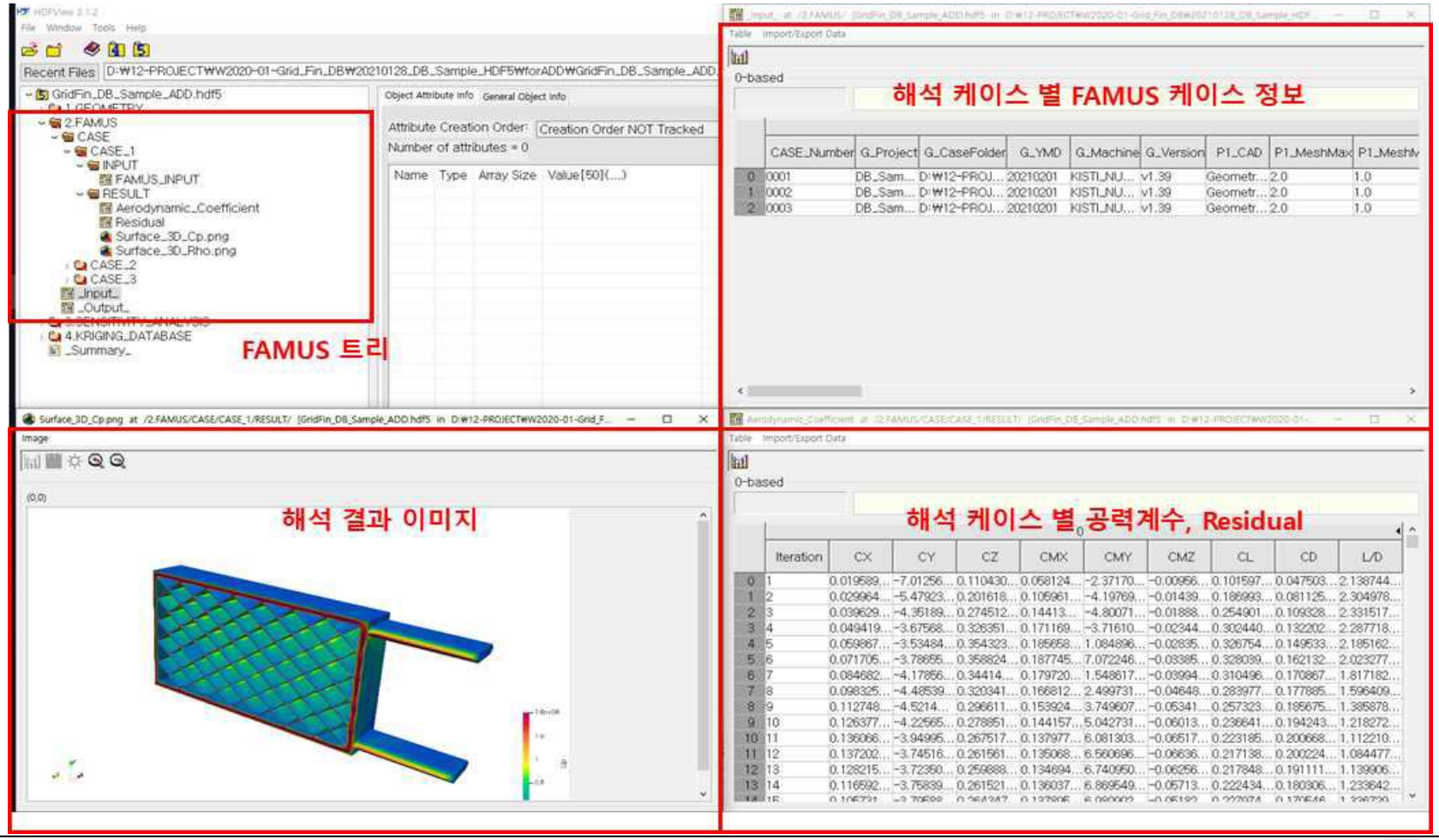
구축 결과

- **형상 및 CAD 데이터**
  - **DOE 기반 샘플링 파라미터 (형상/유동조건), 생성된 형상 이미지, CAD 파일 정보**



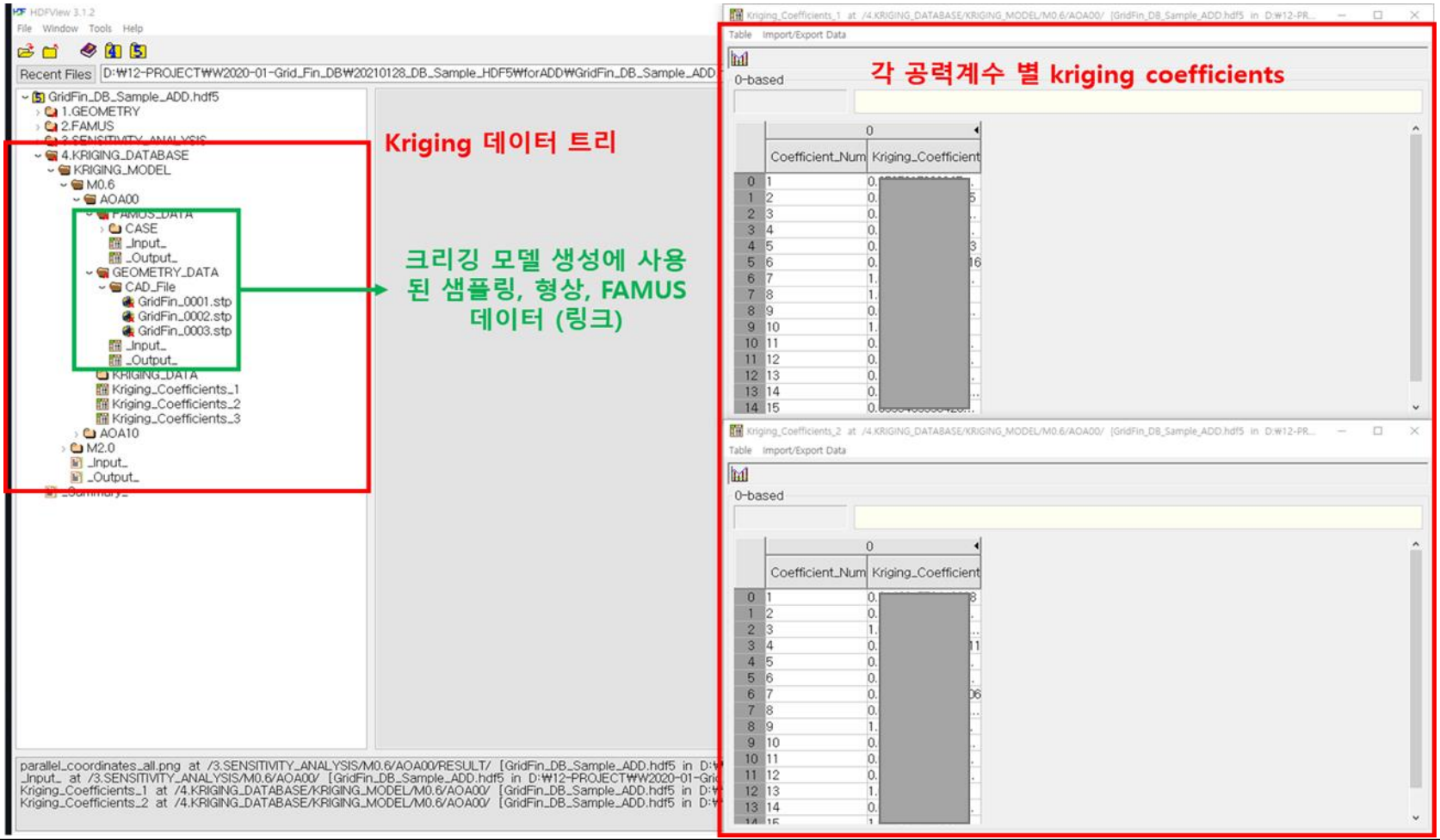
구축 결과

- 유동해석 (FAMUS) 데이터
  - 해석 케이스 별 유동해석 솔버 입력값, 레지듀얼 히스토리, 공력계수, 표면 압력 분포 결과, 핵심 관찰값 정리



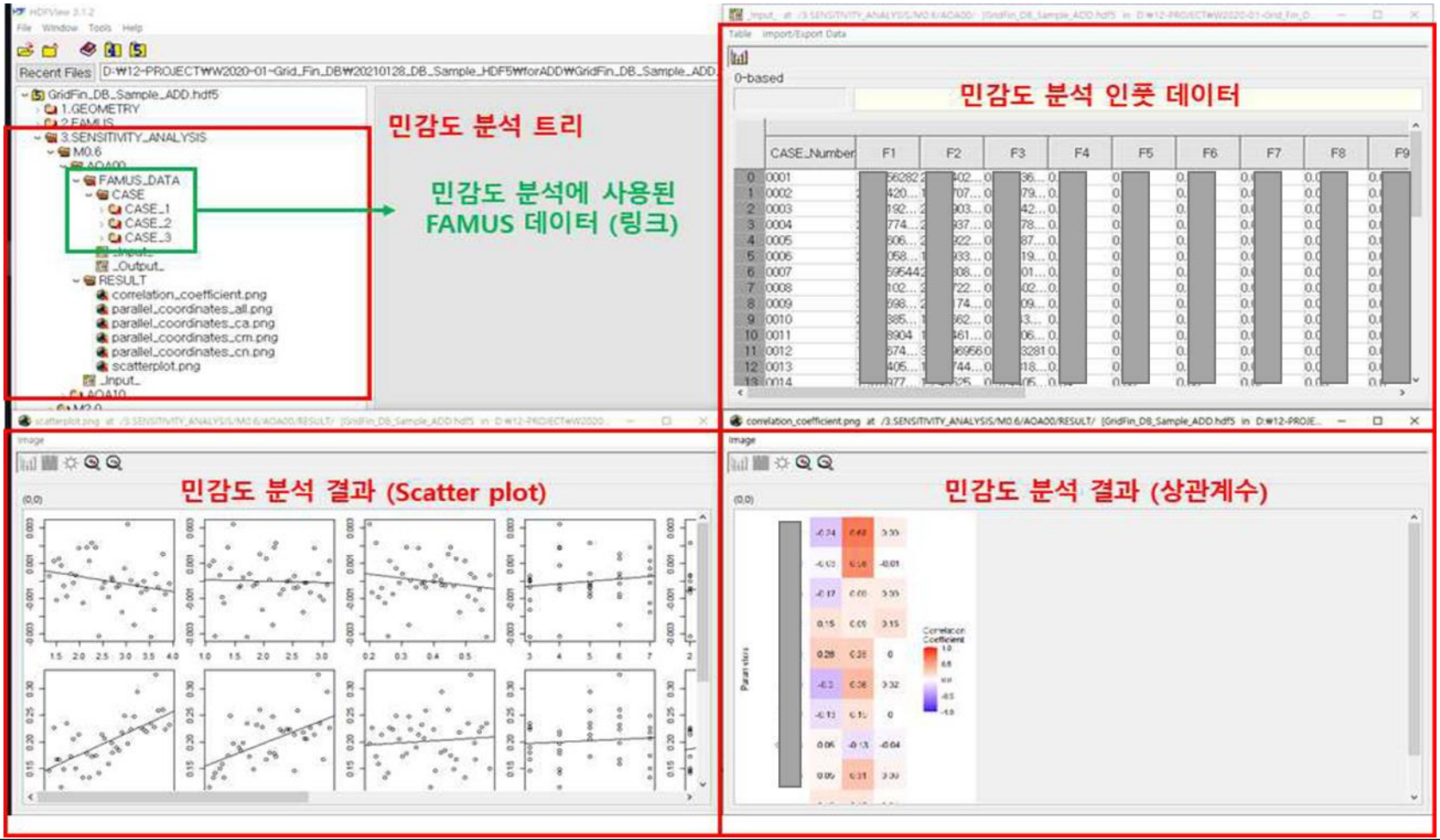
구축 결과

- 크리깅 모델 데이터
  - 샘플 포인트 정보, 유동해석 케이스 링크, 크리깅 모델 상관계수, 모델 검증 결과



구축 결과

- 민감도 분석 데이터
  - 민감도 분석 입력 값 (샘플 파라미터, 공력계수 결과), 유동해석 케이스 링크, 민감도 분석 결과



IV. 결론

1. 요약
2. 후속 연구 내용
3. 관련 라이브러리

NEXTfoam open source CFD consulting

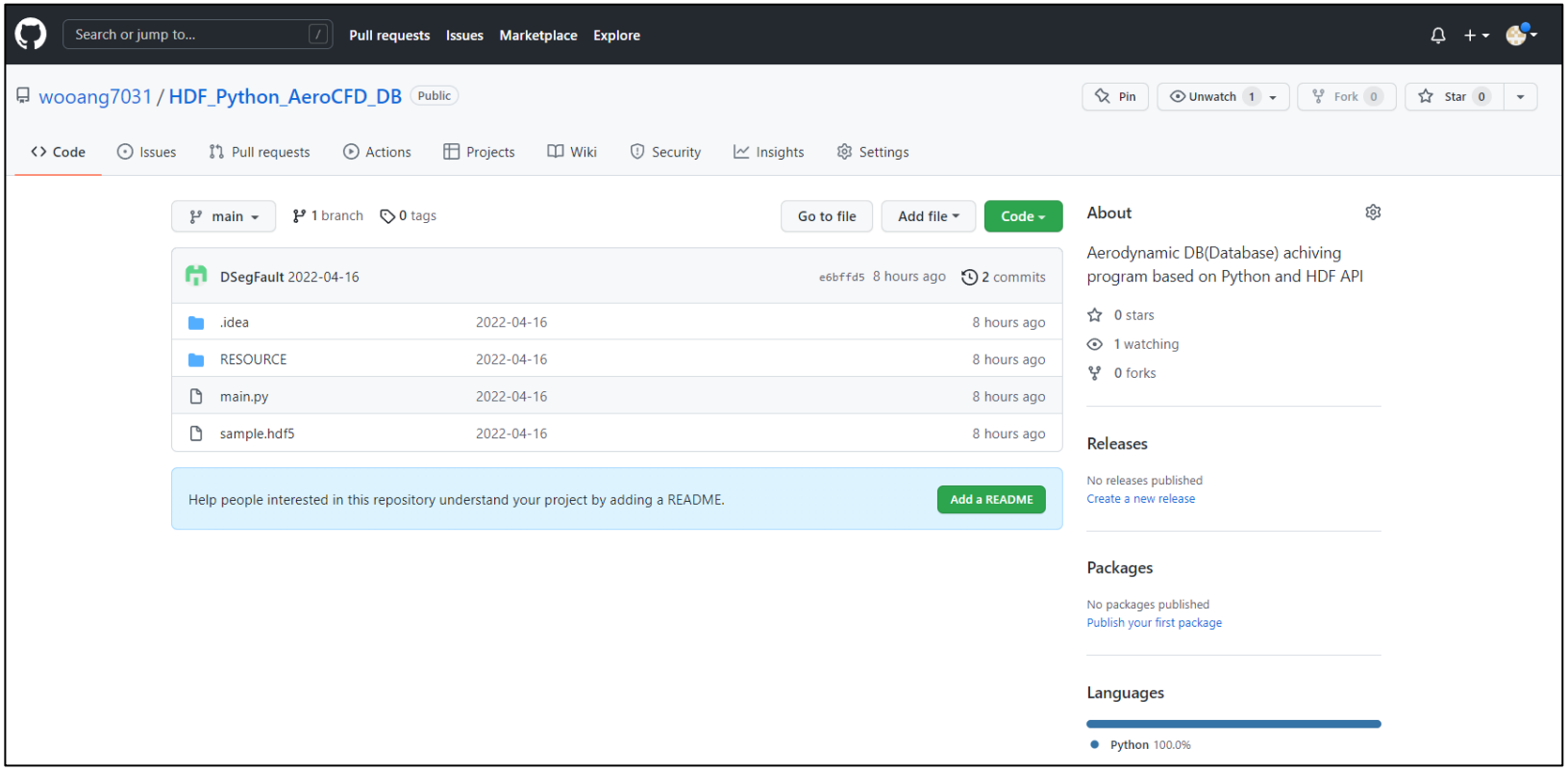
결론

- **요약**
  - **계층적 데이터 포맷을 이용해서 공력 DB 구축 데이터 정리**
  - **데이터의 이질적, 종속적 특성에 기인한 기존 어려움 해소**
  - **공력 데이터 베이스의 관리 및 활용성 향상**

- **후속 연구 내용**
  - **Raw 데이터(필드 데이터) 단계에서부터 데이터 통합 방안 연구**
    - 도출된 데이터의 **후처리 개념(archiving)을 넘어서**, 데이터 도출 단계 (모듈 Run 단계) 에서부터 HDF 데이터 저장
    - 병렬 처리
  - **프로그램 통합 / 프로젝트 관리 및 구동 효율성 증대**
    - **각 모듈 간 데이터 전달을 위한 컨버터 최소화**
    - **산발적 입력 데이터 통합**
    - 계산 결과 계층적으로 HDF 저장
  - **복잡성 해소 / 데이터 처리 효율성 향상**

결론

- 관련 라이브러리
  - 공력 DB 구축에 필요한 핵심 기능 고도화 / 예제 (Python) 정리
  - Github 저장소
    - https://github.com/wooang7031/HDF_Python_AeroCFD_DB.git



경청해 주셔서 감사합니다.

A. 참고문헌

참고문헌

[1] https://www.altair.com/

[2] Poinot, M. (2010). Five good reasons to use the hierarchical data format. Computing in Science & Engineering, 12(5), 84-90.

[3] https://cgns.github.io/hdf5.html

[4] https://www.salome-platform.org/

[5] Fluent, Release 2020 R1 Update

[6] https://www.hdfgroup.org/

[7] https://www.neonscience.org/resources/learning-hub/tutorials/about-hdf5

[8] Pratap Singh, Bhupendra. (2015). The project is based on emerging field Image Processing, In this project A Graphical User Interface has been designed using the software Labwindows. which can process both type of Real time Image Processing and Also many formats of images like .JPG, .DAT , .BMP etc..

[9] http://nextfoam.co.kr