적합직교분해를 이용한 고속열차 전두부 형상최적설계 및 공력성능 향상기술 연구

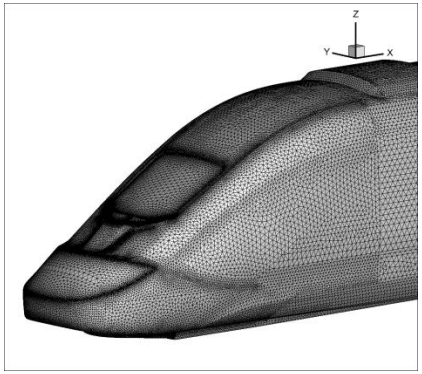
2024. 9. 26

이웅현 선임연구원

NEXTFOAM

연구 동기

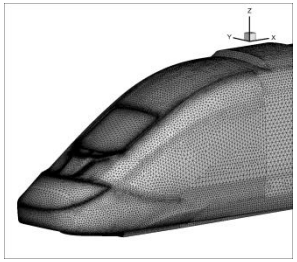
- 기반 과제 : 370kph급 고속열차 전두부 형상 최적화 설계도구 제작 용역
- 2023. 6 ~ 2023. 12
- 전두부 형상 변동에 따른 공력 성능 및 안정성 예측 필요
- 높은 자유도를 갖는 전두부 형상의 모사를 위해 다수의 매개변수로 세밀하게 형상을 정의할 수 있는 모델링 툴의 개발이 필요
- 형상 정의 매개변수가 20개 이상으로 매우 많을 것으로 예상, 초기모델 (full-order model) 의 전산유체역학 해석이 아닌 차수축소모델 (reduced-order model) 및 AI 등을 활용하는 최신 최적화 툴의 개발이 필요



구분	6월				7월				8월				9월				10월				11월			
	1주	2주	3주	4주	1주	2주	3주	4주	1주	2주	3주	4주	1주	2주	3주	4주	1주	2주	3주	4주	1주	2주	3주	4주
매개변수를 이용한 전두부 형상 모델링	■	■	■	■																				
전산유체역학을 이용한 데이터베이스 구축	■	■	■	■	■	■	■	■	■	■	■	■												
차수축소모델을 이용한 메타모델 개발					■	■	■	■	■	■	■	■	■	■	■	■								
전두부 형상 최적화					■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■				

전두부 형상 모델링

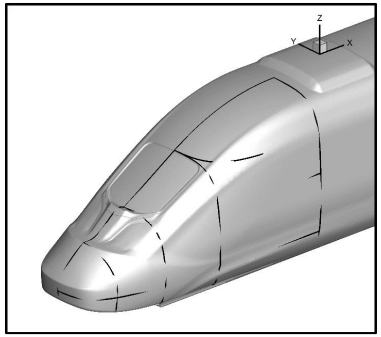
- 형상 매개화 기법 검토
  - 전두부 형상최적설계 선행연구 조사 (한국교통대학교 수행)
    - 형상 곡선/곡면을 대수적으로 계산하여 정의
    - 기 정의된 형상을 Free Form Deformation
    - Etc

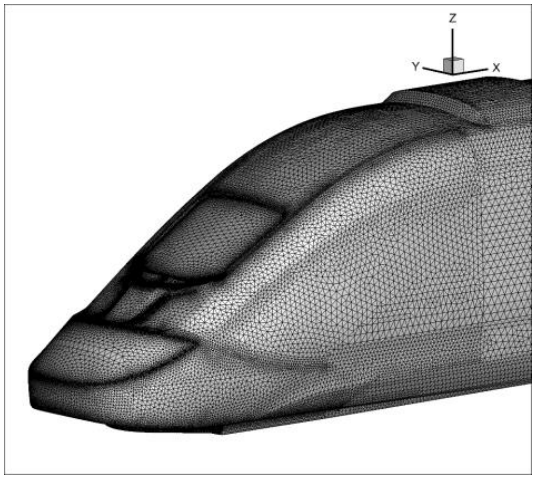


Code	Title	Shape Parameterization			Design Objective			
		dimension	formulation		# of variables	external Parameter	physics	objective function
1997_Iida	Iida, M. (1997). **Effective nose shape for reducing tunnel sonic boom**. *QR of RTRI, 38*(4), 206-211.	Axi-symmetric	Agebric equation	$\frac{A(x)}{\pi b^2} = (1-\alpha_2)\left[(1-\alpha_1)\frac{x}{a} + \alpha_1\sqrt{\frac{x}{a}}\right] + \alpha_2\left(\frac{x}{a}\right)^2$	2	R, a/b	Micro-pressure wave	Min. Pressure gradient of CW
2001_Kwon	Kwon, H. B., Jang, K. H., Kim, Y. S., Yee, K. J., & Lee, D. H. (2001). **Nose shape optimization of high-speed train for minimization of tunnel sonic boom**. *JSME International Journal Series C*	Axi-symmetric	Hicks-Henne shape function		6	Vt, R, a/b	Micro-pressure wave	Min. Pressure gradient of CW
2009_Ku-Dissert	Ku, Y. C. (2009). **Two-step multi-objective nose shape optimization of a high-speed train using the vehicle modeling function** (Doctoral dissertation, Ph. D Thesis, Seoul National	T.B.A						
2009_Rho	Kee, J. D., & Lee, D. H. (2009). **Development of a Vehicle Modeling Function for Three-Dimensional Shape Optimization**. *Journal of Mechanical Design*, 131, 121004-1.							
2010a_Ku	Ku, Y. C., Park, H. I., Kwak, M. H., & Lee, D. H. (2010, January). **Multi-objective optimization of high-speed train nose shape using the vehicle modeling function**. In *48th AIAA Aerospace*	Axi-symmetric 3D	Hicks-Henne shape function / new Vehicle Modeling Function	$Z(x) = \frac{H}{L^{A_1+A_2}}(x-x_p)^{A_1}[2L-(x-x_p)]^{A_2} + z_p$	6 4	a(nose length)	Micro-pressure wave/ Aerodynamic drag	Min. Pressure gradient of CW Min. presure drag coeff.(C_{Dp})
2010b_Ku	Ku, Y. C., Rho, J. H., Yun, S. H., Kwak, M. H., Kim, K. H., Kwon, H. B., & Lee, D. H. (2010). **Optimal cross-sectional area distribution of a high-speed train nose to minimize the tunnel**							
2011_Kikuchi	Kikuchi, K., Iida, M., & Fukuda, T. (2011). **Optimization of train nose shape for reducing micro-pressure wave radiated from tunnel exit.** *Journal of Low Frequency Noise, Vibration*	Axi-symmetric	Agebric equation	$\frac{A_x(s)}{A_0} = (1-\alpha_2)\left[(1-\alpha_1)\frac{s}{l_x} + \alpha_1\sqrt{\frac{s}{l_x}}\right] + \alpha_2\left(\frac{s}{l_x}\right)^2$	2	BUHOOD	Micro-pressure wave	Min. Pressure gradient of CW
2013_Kwak	Kwak, M., Yun, S., Lee, Y., Kwon, H., Kim, K., & Lee, D. H. (2013). **Optimum nose shape of a front-rear symmetric train for the reduction of the total aerodynamic drag**. *Journal of*	3D	Ku's VMF		4	Opt. A-distribution	Aerodynamic drag	Min. total drag coeff.(C_D)
2013_Suzuki	Suzuki, M., & Nakade, K. (2013). **Multi-objective design optimization of high-speed train nose**. *Journal of Mechanical Systems for Transportation and Logistics, 6*(1), 54-64.	3D	B-Spline curves(CS) bilinear Coons patches(Btwn. CS)		4	Opt. A-distribution	Aerodynamic drag Slipstream	Min. nose pressure drag Min. $\triangle C_p$
2013_Yu	Yu, M. G., Zhang, J. Y., & Zhang, W. H. (2013). **Multi-objective optimization design method of the high-speed train head**. *Journal of Zhejiang University SCIENCE A, 14*(9), 631-641.	3D	deformation from baseline with 162 control points		5		Aerodynamic drag Vehicle dynamics	Min. aerodynamic drag(F_d) (?)
2014_Munoz	Munoz-Paniagua, J., García, J., & Crespo, A. (2014). **Genetically aerodynamic optimization of the nose shape of a high-speed train entering a tunnel**. *Journal of wind engineering and*	3D	based on generic train by Sima(20		3		Micro-pressure wave Aerodynamic drag	Min. Pressure gradient of CW
2014_Munoz-Di	Muñoz Paniagua, J. (2014). **Aerodynamic Optimization of the Nose Shape of a High-Speed Train** (Doctoral dissertation, Industriales).	T.B.A						
2016_Li	Li, R., Xu, P., Peng, Y., & Ji, P. (2016). **Multi-objective optimization of a high-speed train head based on the FFD method**. *Journal of Wind Engineering and Industrial Aerodynamics, 152*,	3D	FFD method from base geometry		5		Aerodynamic drag Aerodynamic lift	Min. Cd-total Max. Cl-head and Cl-tail
2017_Sun	Sun, Z., Zhang, Y., & Yang, G. (2017). **Surrogate based optimization of aerodynamic noise for streamlined shape of high speed trains**. *Applied Sciences, 7*(2), 196.	3D	Local Shape Function (LSF) based on FFD method		4		Aerodynamic noise Aerodynamic drag	Min. equivalent A-weighted SP Min. Cd
2019_Munoz	Muñoz-Paniagua, J., & García, J. (2019). **Aerodynamic surrogate-based optimization of the nose shape of a high-speed train for crosswind and passing-by scenarios.** *Journal of Wind*	3D	based on generic train by Sima(2011)		3 (length, slenderness,	passing by / 60° yaw angle	Passing-by Crosswind	Min. pressure pulse($\triangle p$) Min. Side force coeff.(C_s)
2020_Li	Li, R., Xu, P., & Yao, S. (2020). **Optimization of the high-speed train head using the radial basis function morphing method**. *Proceedings of the Institution of Mechanical Engineers,*	3D	RBF morph strategy from baseline		3		Aerodynamic drag	Min. Cd-head / Min. Cd-tail
2020_Munoz	Munoz-Paniagua, J., & García, J. (2020). **Aerodynamic drag optimization of a high-speed train**. *Journal of Wind Engineering and Industrial Aerodynamics*, 204, 104215.	3D	ATM for baseline Bezier curves in side, front, top		25		Aerodynamic drag	Min. Cd-total
2022_He	He, Z., Liu, T., & Liu, H. (2022). **Improved particle swarm optimization algorithms for aerodynamic shape optimization of high-speed train**. *Advances in engineering software, 173*,	3D	FFD method		5		Aerodynamic drag Aerodynamic lift	Min. Cd-total Max. Cl-tail

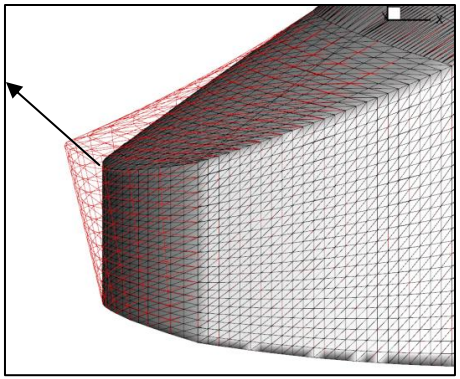
전두부 형상 모델링

- 형상 매개화 기법 검토
  - 형상 곡선/곡면을 대수적으로 계산하여 정의
    - 전두부 형상을 모사할 수 있는 모델 개발 필요





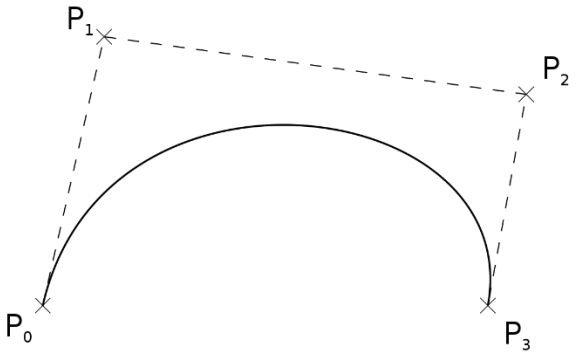
  - 기 정의된 형상을 Free Form Deformation
    - 기하학적 구속조건 (windshield 각도, 직선 조건 등) 을 유지하기 어려움



전두부 형상 모델링

- 매개변수를 통한 3차원 형상 정의
  - 베지에 (Bezier) 곡선
    - n개의 control point 및 0≤t≤1로 정의

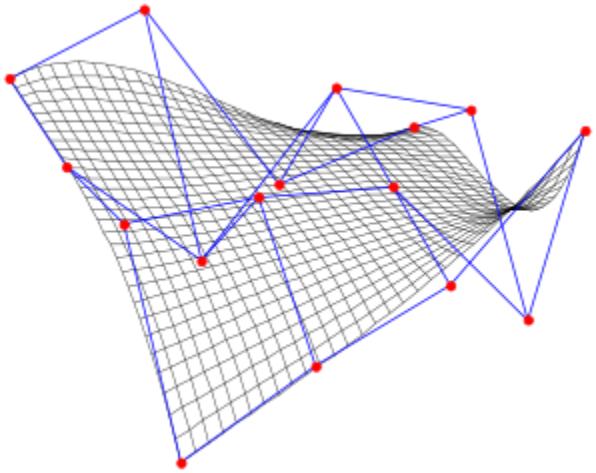
$$\mathbf{B}(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i \mathbf{P}_i$$



  - 베지에 곡면
    - n * m 개의 control point 및 0≤u,v≤1로 정의

$$\mathbf{p}(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_i^n(u) B_j^m(v) \mathbf{k}_{i,j}$$

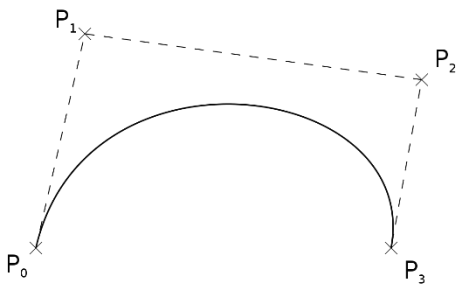
$$B_i^n(u) = \frac{n!}{i!(n-i)!} u^i (1-u)^{n-i}$$

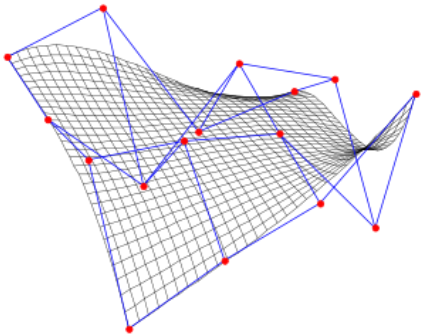


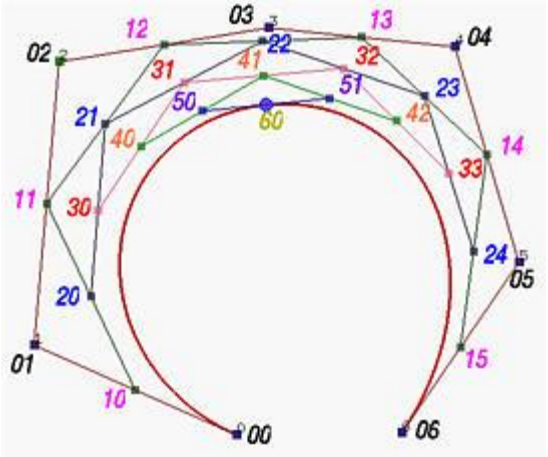
  - 일반화 시 B-spline, NURBS

전두부 형상 모델링

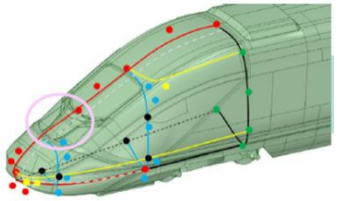
- 매개변수를 통한 3차원 형상 정의
  - 베지에 곡선/곡면 특성
    - 주어진 베지에 곡선을 boundary로 하는 베지에 곡면 정의 가능
    - 곡면상의 특정 위치 (u1, v1) 를 매개변수로 계산 가능
    - 접선 조건 지정 가능
      - 각 끝단으로부터 첫 번째 control point 를 잇는 선분에 접함
    - 베지에 곡면의 slice 또한 베지에 곡선으로 계산 가능
    - 동일 차수의 곡선/곡면 조합으로 분할 가능
      - 한 곡면에 여러 곡면이 인접하는 경우의 처리 용이
    - 상위 차수의 곡선/곡면으로 변환 가능
      - Boundary 베지에 곡선의 차수가 일치하지 않아도 곡면 구성 가능



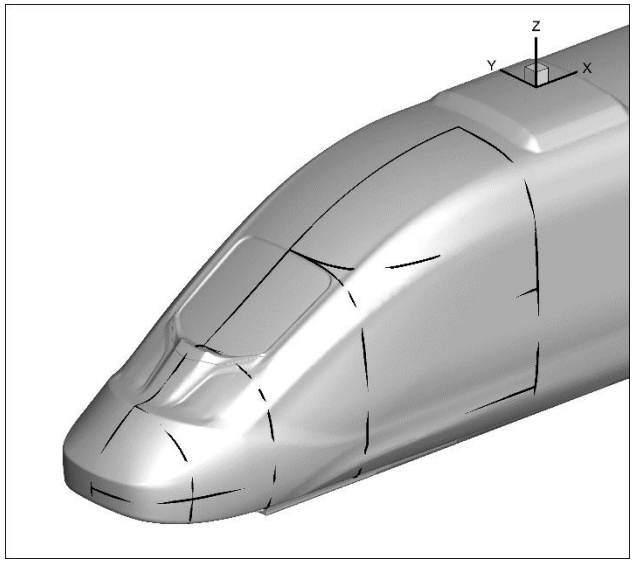


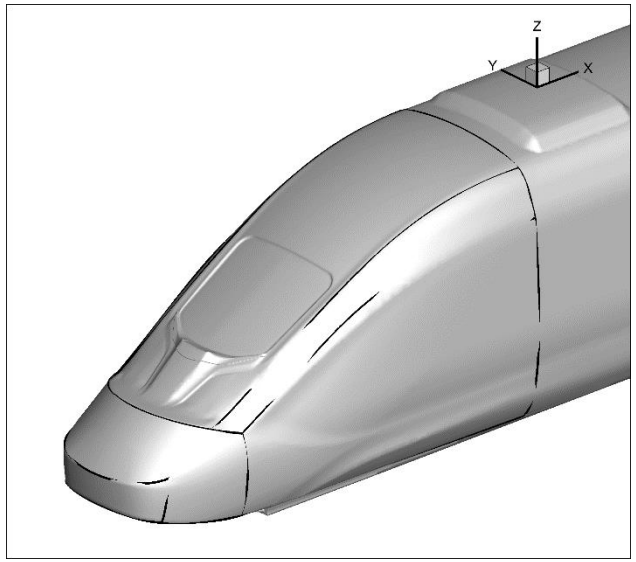


전두부 형상 모델링



- 설계안 생성 및 검토
  - Bezier 곡선의 구성 (개수, 차수, 연결) 차이
  - 좌 : 한국교통대학교, Munoz(2014)가 제시한 모델 개선, 매개변수 27개
  - 우 : 넥스트폼, EMU 형상을 기반으로 신규 생성, 매개변수 37개

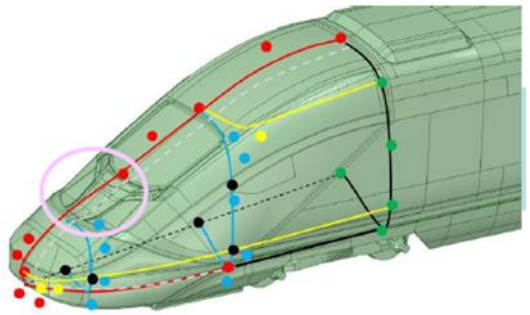


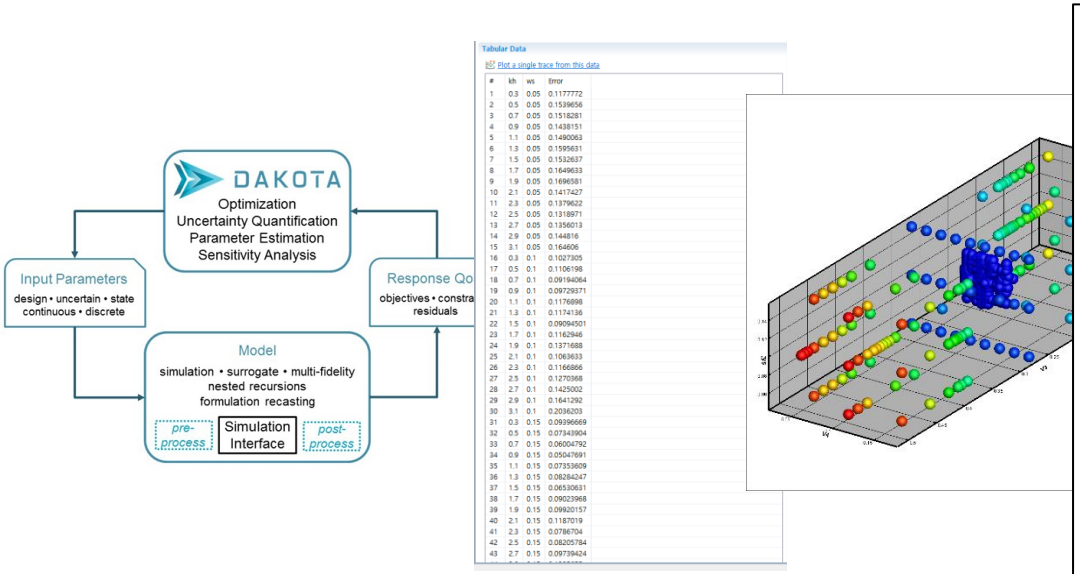


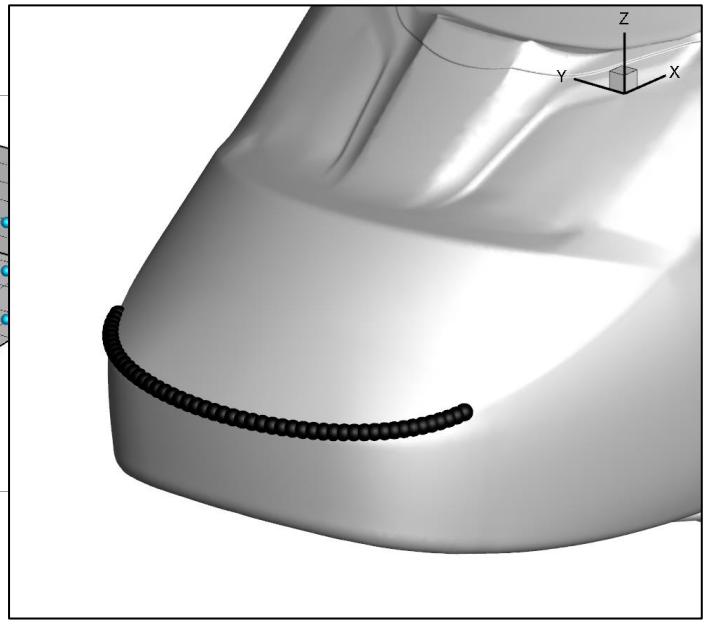
  - EMU-320 형상에 대해 2가지 설계안 (Bezier surface 조합) 의 정확도 검토
    - 최적 매개변수값은 DAKOTA toolkit으로 획득
    - 생성된 곡선 - target surface와의 평균 거리로 정확도 비교
    - 특성상 NextFOAM 모델의 정확도가 근소하게 높음
  - 기하학적 구속조건 적용 난이도는 Munoz 모델이 유리
    - 매개변수 ↔ 기하학적 구속조건의 관계식이 이미 정리되어 있음
    - 기존 고속열차 (ICE, TGV 등) 를 모사할 수 있음이 검증됨
  - 확장성, 구속조건, 매개변수 개수 등을 고려하여 Munoz 방식 채택 (교통대)

전두부 형상 모델링

- 설계안 생성 및 검토
  - 기존 CAD 형상에 가장 가까운 곡선/곡면 생성
    - Bezier curve의 control point 좌표 문제
  - 프로그램 작성
    - 입력 : Bezier curve의 control point의 좌표 (2~5개)
    - 출력 : 생성된 Bezier curve와 기준 CAD 간의 오차 (평균 거리)
  - 작성된 프로그램을 DAKOTA와 연동하여 실행
  - DAKOTA objective : 출력값(오차) 최소화 문제
  - 최적화된 입력매개변수 획득





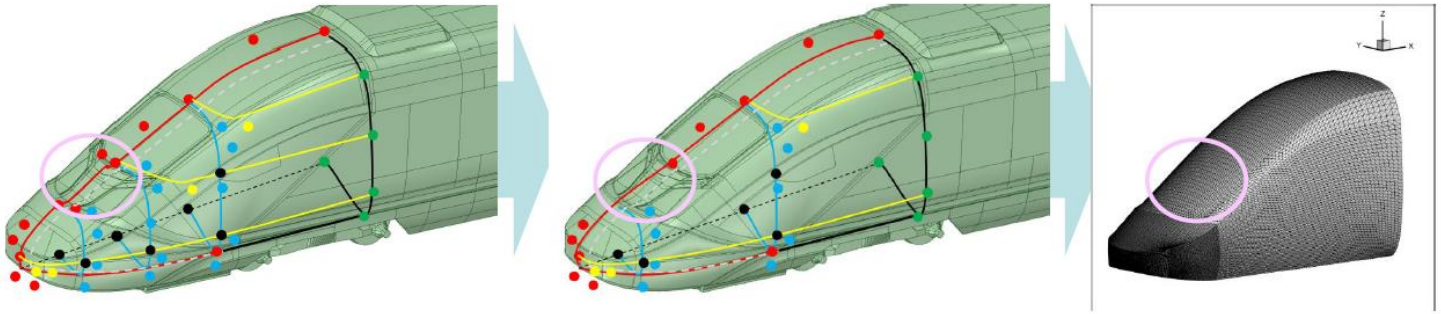


전두부 형상 모델링

- 설계안 확정 (형상매개변수 27개)

최적화 설계 제약사항 검토 및 매개변수 설정

- **매개변수화 결과**
  - 측면 5개, 정면 4개, 상면 3개 총 12개 곡선 정의 필요
    - 전두부 상면(S3, F3), 객차부 단면(F1) 제외 시, **9개 곡선**
  - 측면 14개, 정면 14개, 상면 4개, 접점 7개 총 39개 조절점 정의 필요
    - 이 중, 전두부 상면을 제외할 경우 10 + 9 + 3 + 5 = **27개 조절점**
  - 측면 24개, 정면 20개, 상면 4개, 총 48개 변수 사용
    - 전두부 상면(S3) 측면 7개, 정면 5개, 상면 1개, 객차부 단면(F1) 정면 5개
    - 측면 17개, 정면 10개, 상면 3개 **총 30개 변수** 사용



TSDL Transportation System Design Lab

19

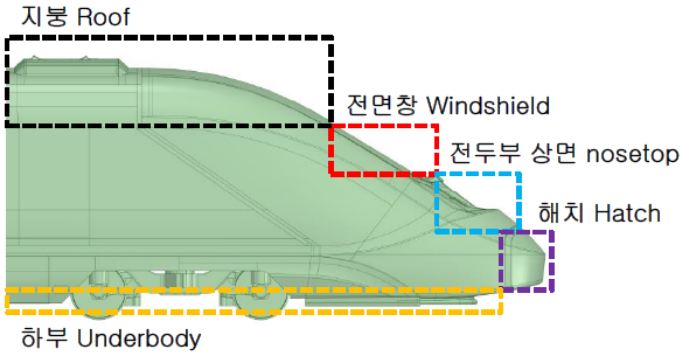
국립한국교통대학교 KOREA NATIONAL UNIVERSITY OF TRANSPORTATION

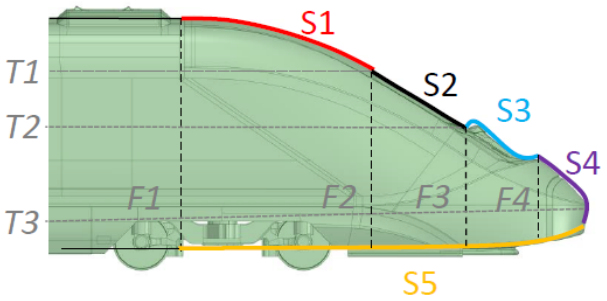
전두부 형상 모델링

- 설계안 확정 (형상매개변수 27개)

최적화 설계 제약사항 검토 및 매개변수 설정

- **매개변수화**
  - 선행 연구 기반(J. Munoz) 구성요소 분리 수행
    - 구성요소를 5개 파트(지붕, 전면창, 전두부 상면, 해치, 하부)로 분리
  - 구성요소별 주요 변곡점 도출
    - 측면 5개(S1 ~ S5), 정면 4개(F1 ~ F4), 상면 3개(T1 ~ T3) **총 12개 곡선** 정의
      - 이 중, F1은 열차 기준 단면으로 고정값으로 볼 수 있음





▲ Lateral-side view of EMU-320

ISDL Transportation System Design Lab

8

국립 한국교통대학교 KOREA NATIONAL UNIVERSITY OF TRANSPORTATION

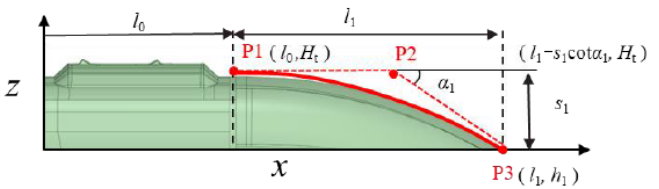
전두부 형상 모델링

- 설계안 확정 (형상매개변수 27개)

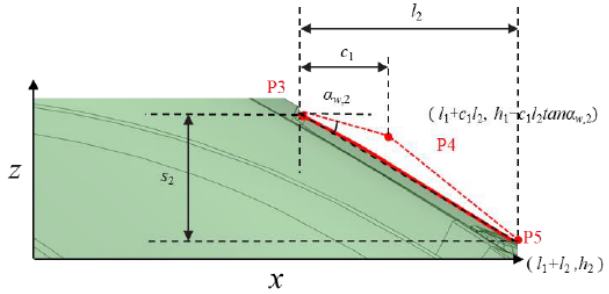
형상 설계 변수 및 제약조건 정의

- **설계 변수 (측면 S1, S2)**
  - 7개 설계 변수

Design Variables				
Surf.	min	Sym.	max	Description
S1	1,590	l_1	2,995	지붕 길이
	5.39	α_1	30.19	지붕 곡률
	150	s_1	925	지붕 높이
S2	900	l_2	3,967	전면창 길이
	0	c_1	0.87	전면창 곡률 조절점 조정 계수
	0	$\alpha_{w,2}$	25.94	전면창 곡률
	610	s_2	1,850	전면창 높이



▲ Coordinate of each point at S1 surface



▲ Coordinate of each point at S2 surface

ISDL Transportation System Design Lab

국립한국교통대학교 KOREA NATIONAL UNIVERSITY OF TRANSPORTATION

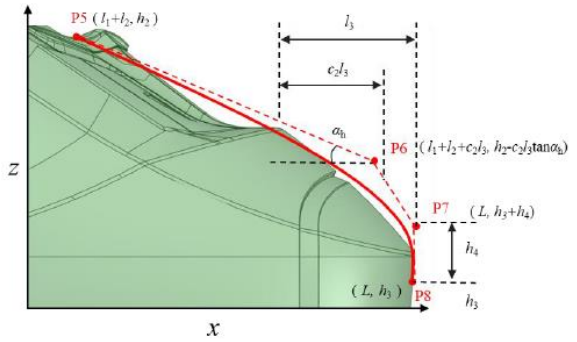
전두부 형상 모델링

- 설계안 확정 (형상매개변수 27개)

형상 설계 변수 및 제약조건 정의

- **설계 변수 (측면 S3, S4)**
  - 7개 설계 변수

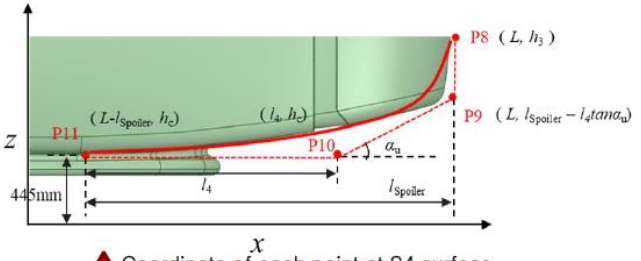
Design Variables				
Surf.	min	Sym.	max	Description
S3	245	h_3	2,895	해치 최대 높이 (mm)
	0	h_4	2,895	해치 중심부 높이
	0	c_2	1	해치 곡률 조절점 조정 계수
	0	α_h	90	해치 곡률
S4	0	l_4	6,530	전두부 하부 ~ 연결기 길이
	455	l_{Sp}	6,530	전두부 하부 길이
	0	α_u	90	전두부 하부 곡률



▲ Coordinate of each point at S3 surface

- **제약 조건 (측면 S1~S4)**
  - 1개 제약 조건

Design Constraints	
Value	Description
$l_1+l_2+l_3 = 6{,}530$ (L)	전두부 길이 *l_3 = 해치 길이



▲ Coordinate of each point at S4 surface

ISDL Transportation System Design Lab

8

국립한국교통대학교 KOREA NATIONAL UNIVERSITY OF TRANSPORTATION

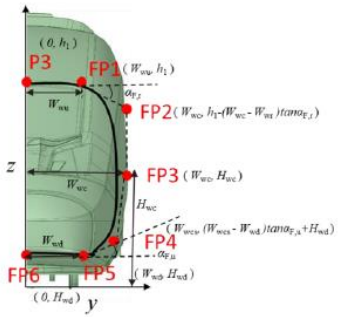
전두부 형상 모델링

- 설계안 확정 (형상매개변수 27개)

형상 설계 변수 및 제약조건 정의

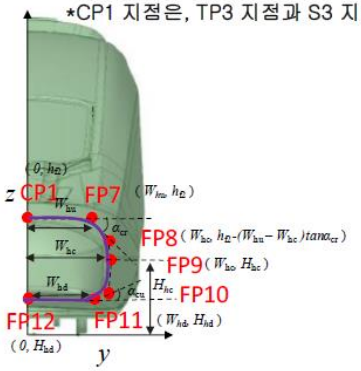
- **설계 변수 (정면 F1, F2)**
  - 12개 설계 변수

Design Variables				
Surf.	min	Sym.	max	Description
F1	655	W_{wu}	1,600	전면창 상부 너비
	1,000	W_{wc}	1,600	전면창 최대 너비
	373.6	H_{wc}	3,564	전면창 최소 높이
	400	W_{wd}	1,300	전면창 하부 최대 너비
	400	W_{wcs}	1,500	전면창 하부 너비
F2	400	W_{hu}	1,500	해치 상부 너비
	400	W_{hc}	1,300	해치 최대 너비
	23.96	α_{cr}	90	해치 상부 곡률
	465	H_{hc}	3,140	해치 최대 너비 수직 높이
	23.96	α_{cu}	90	해치 하부 곡률
	400	W_{hd}	1,300	해치 하부 최대 너비
	400	W_{hcs}	1,300	해치 하부 너비



▲ Coordinate of each point at F1 surface

*CP1 지점은, TP3 지점과 S3 지점으로 추정 가능



▲ Coordinate of each point at F2 surface

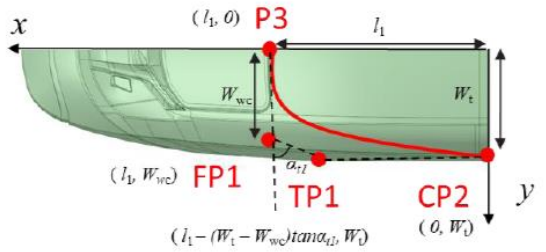
전두부 형상 모델링

- 설계안 확정 (형상매개변수 27개)

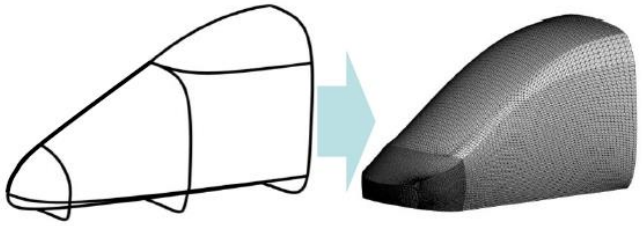
형상 설계 변수 및 제약조건 정의

- **설계 변수 (상면 T1, T2)**
  - 3개 설계 변수

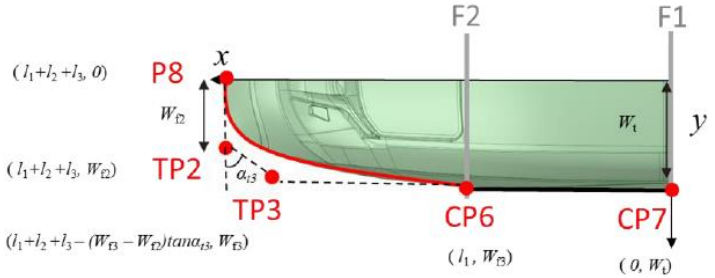
Design Variables				
Surf.	min	Sym.	max	Description
T1	43.09	α_{t1}	71.54	A-pillar 곡률
T2	17.42	α_{t3}	86.75	해치 곡률
	400	W_{f1}	1,450	해치 너비



▲ Coordinate of each point at T1 surface



▲ Final model of EMU-320 ATM (Baseline)



▲ Coordinate of each point at T2 surface

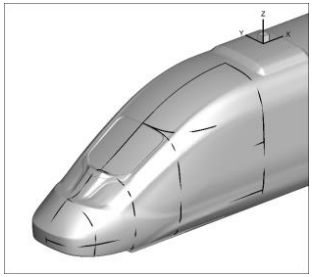
Transportation System Design Lab

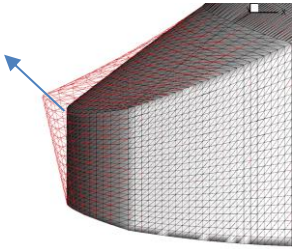
국립한국교통대학교 KOREA NATIONAL UNIVERSITY OF TRANSPORTATION

전두부 형상 모델링

- CAD (STL) 생성 코드 개발
  - 형상 매개변수 파일 입력
  - Control point 계산
  - Bezier curve 계산
  - Surface 계산
    - 매개변수 u, v에 따른 좌표 계산
  - STL 파일 출력

  - 형상최적설계
    반복해석 구성요소 확보





parameterbezier.cpp
parameterbezier.exe
parameterbezier.hpp

```
vector<vector<double>> gen_bezier_with_points(vector<vector<double>> pList) {
    vector<vector<double>> c;

    int widthSurface = pList.size();

    for (int ix=0; ix<resolutionForBezier+1; ix++) {
        vector<double> pxyz {0, 0, 0};
        double rx = double(ix)/double(resolutionForBezier);
        for (int iCP=0; iCP<widthSurface; iCP++) {
            vector<double> currentControlPoint = pList[iCP];
            double currentPX = currentControlPoint[0];
            double currentPY = currentControlPoint[1];
            double currentPZ = currentControlPoint[2];
            pxyz[0] += bernsteinpoly(widthSurface-1,iCP,rx) * currentPX;
            pxyz[1] += bernsteinpoly(widthSurface-1,iCP,rx) * currentPY;
            pxyz[2] += bernsteinpoly(widthSurface-1,iCP,rx) * currentPZ;
        }
        c.push_back(vector<double> {pxyz[0],pxyz[1],pxyz[2]});
    }

    return c;
}
```

input.inp

1	2.6928517848346316	//l1
2	0.5556359338328103	//alpha1
3	0.7792911792406366	//s1
4	1.2862841720458471	//l2
5	0.9294370405811180	//s2
6	0.5207206480549580	//c1
7	0.2600000000000000	//alphaw2
8	0.4046678122913631	//c2
9	0.3882255637186351	//alphah
10	0.4855037716686711	//h3
11	0.4509979185999421	//h4
12	1.1931164990548073	//l4
13	2.3248263664123470	//lsp
14	0.0359355899504188	//alphau
15	1.1185478546920840	//Wwu
16	1.5938098807042711	//Wwc
17	1.0789185619717796	//Hwc
18	1.3135545317013611	//Wwcs
19	1.2007930244034042	//Wwd
20	1.0362571883552170	//alphaT1
21	0.3548545863232508	//alphaT2
22	0.5423981064293664	//Wf2
23	0.4854418089851649	//Whu
24	0.3786412081777248	//alphacr
25	0.4592623585455981	//Whd
26	0.4728558289004981	//Whcs
27	0.2791267624522286	//alphacu

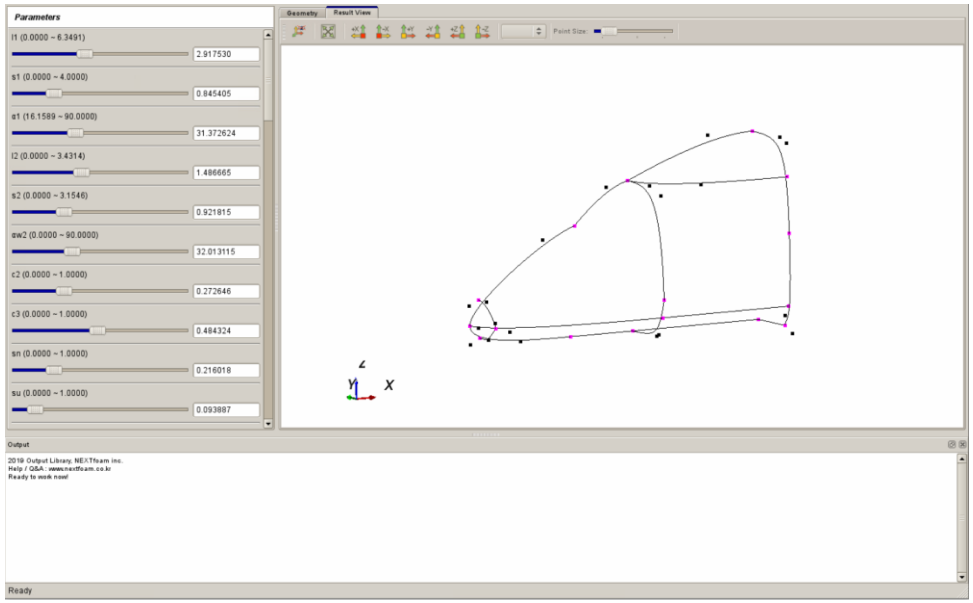
```
102     vector<double> pHeadPri {xpHeadPri, 0., zpHeadPri};
103     vector<double> pInterP {xInter, yInter, zInter};
104     vector<double> pInterM {xInter, -yInter, zInter};
```

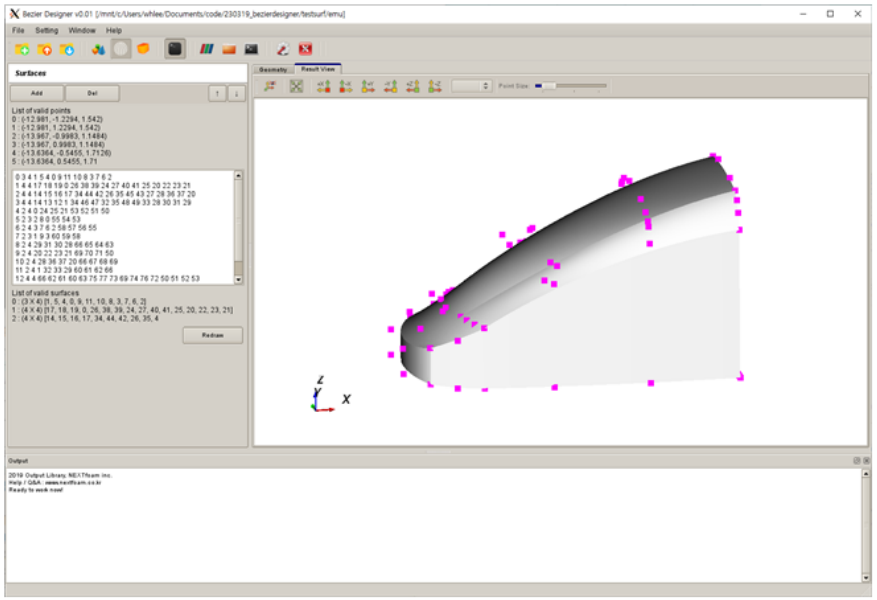
```
vector<vector<double>> lSliceM2M3 = gen_bezier_with_points(vector<vector<double>> {pSliceM2, cpSliceM2M3, pSliceM3});
vector<vector<double>> lSliceP2P3 = gen_bezier_with_points(vector<vector<double>> {pSliceP2, cpSliceP2P3, pSliceP3});
vector<vector<double>> lCurveM2 = gen_bezier_with_points(vector<vector<double>> {pInterM, pCurveM2_1, pCurveM2_2, pSliceM2});
vector<vector<double>> lCurveP2 = gen_bezier_with_points(vector<vector<double>> {pInterP, pCurveP2_1, pCurveP2_2, pSliceP2});
```

```
vector<STLtriangle> sCenterUp = gen_stl_with_curves(reflectionIVec(lCurveHeadUp), lCurveHeadSideM, lCurveCenterUp, reflectionIVec(lCurveHeadSideP), alphaSHead);
vector<STLtriangle> sHeadPriSec = gen_stl_with_curves(reflectionIVec(lCurveHeadDown), lHeadSideZM, lCurveHeadUp, reflectionIVec(lHeadSideZP), 0);
vector<STLtriangle> sInterM = gen_stl_with_curves(reflectionIVec(lSliceM1M2), reflectionIVec(lCurveM1), reflectionIVec(lSplittedCenterUpM), lCurveM2, 0);
```

전두부 형상 모델링

- 전두부 형상 생성 GUI 개발

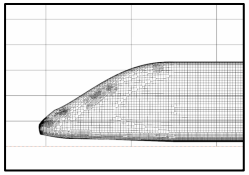
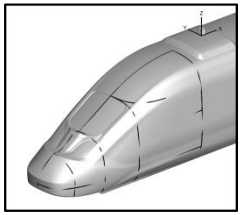
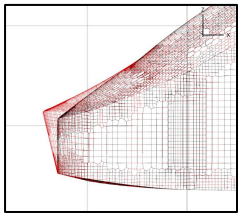
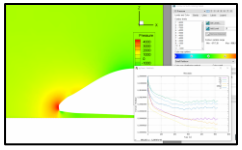


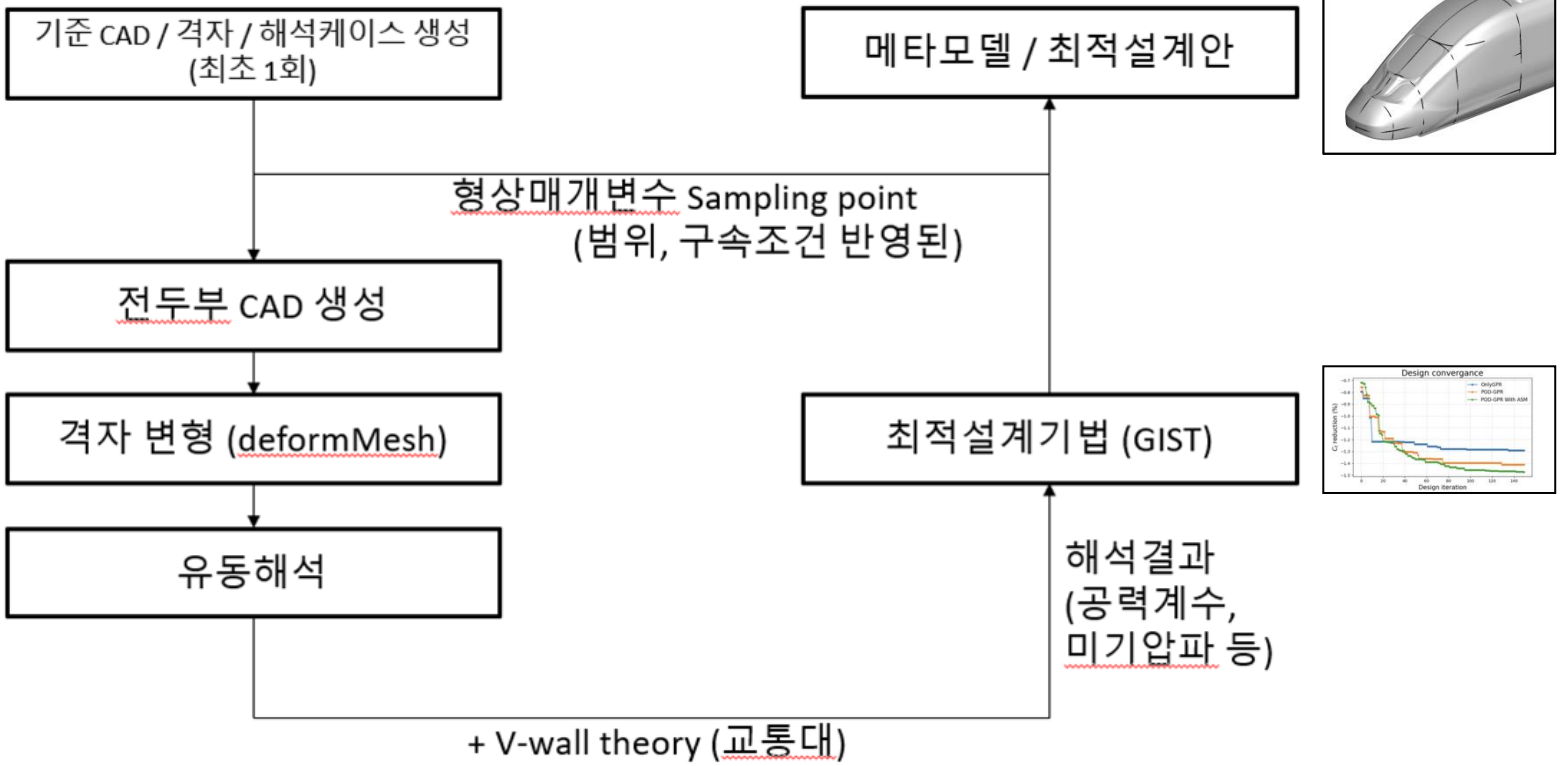


- 스크롤바 조작을 통한 형상매개변수 제어
- 구속조건 범위 내에서 변경 가능
- 타 매개변수에 종속인 구속조건을 자동으로 업데이트
- 매개변수 변경에 따른 형상 곡선 변동 실시간 확인
- 곡면 업데이트는 초 단위 지연 시간 소요
- CAD 생성 코드 연동
- 고속열차 전두부 최적설계 시 형상 모델링 기반 확보

CFD 해석 및 DB 구축

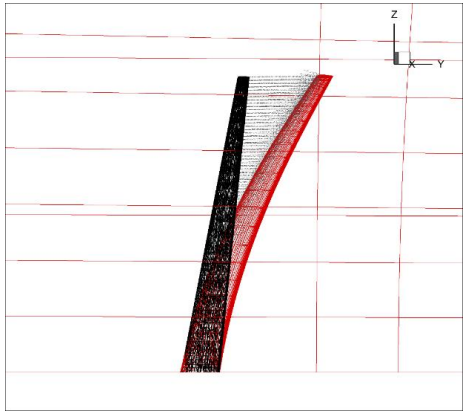
- 해석 계획 수립
  - 반복해석 수행
  - 27개 매개변수 / 대략 500개 표본 예상
  - 적합직교분해 (POD) 기반 메타모델 생성 목표
    - 격자 구조 유지 필요, 공간격자 재생성 없이 기준 격자를 deform하여 사용

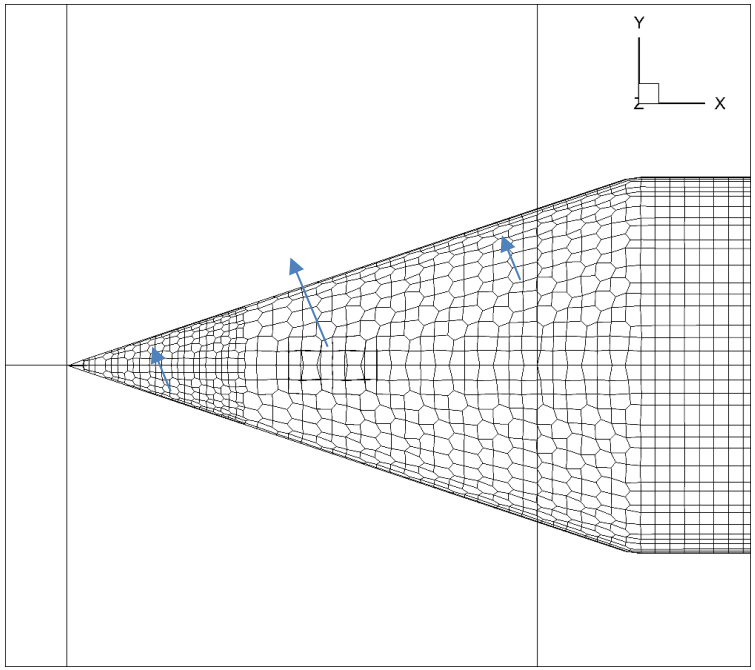





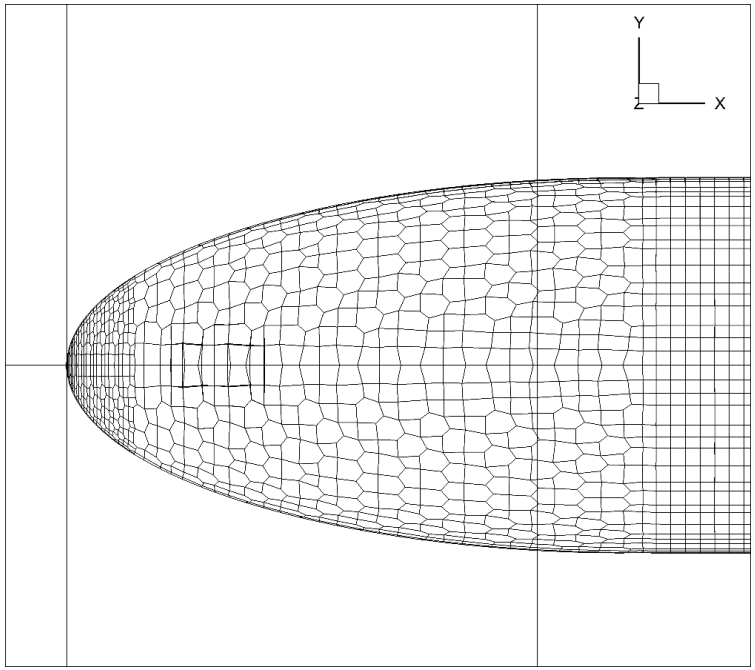


CFD 해석 및 DB 구축

- 형상 변형 (deformMesh)
  - deformMesh - OpenFOAM Utility (넥스트폼 자체 개발)
  - control point 좌표 및 변위 벡터를 입력하여 변형
  - RBF, IDW, RBFIDW 방식 지원
  - 경계층 격자 품질 유지
  - 최초 1회 생성한 격자를 반복 사용 가능

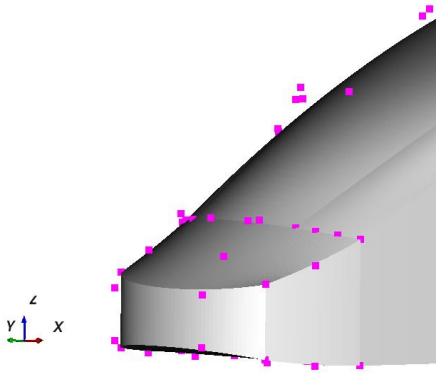




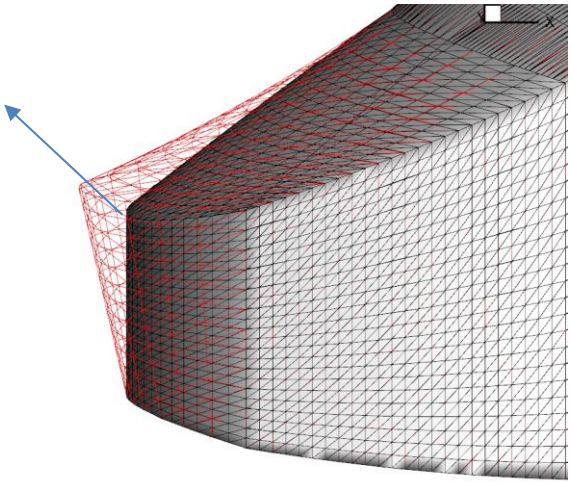


CFD 해석 및 DB 구축

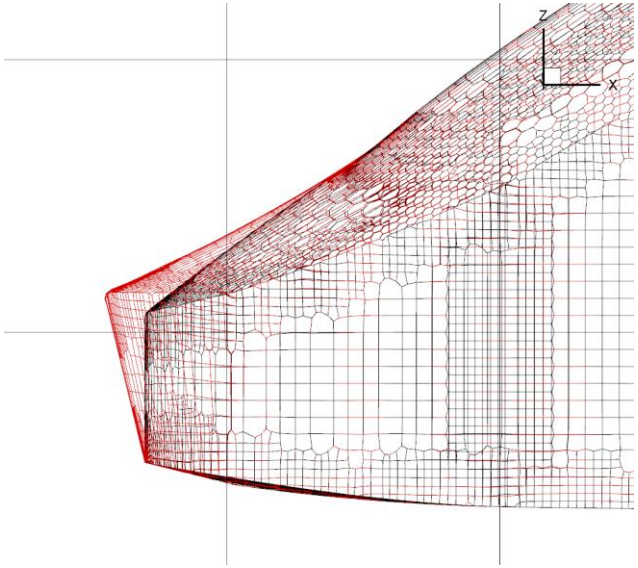
- 형상 변형 (deformMesh)
  - 형상 매개변수 변동에 따른 베지에 곡면 변화 추적
  - 변동 전후의 베지에 곡면 상에서 동일 매개변수값 (u,v)에 해당하는 점들의 변위로부터 격자 변형에 필요한 input 계산
  - 최초 생성한 격자계를 변형하여 반복 해석에 사용 가능
  - 격자의 topology 및 index가 유지되므로 적합직교분해 (POD) 기반 차수축소모델 구성 가능



매개변수화된 형상



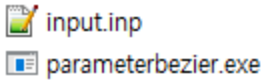
매개변수 변동에 따른
베지에 곡면 형상 변화



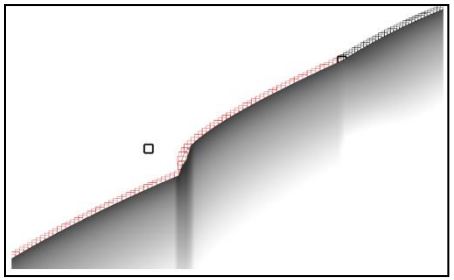
실제 해석 격자 변형 결과

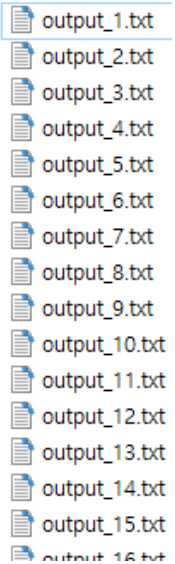
CFD 해석 및 DB 구축

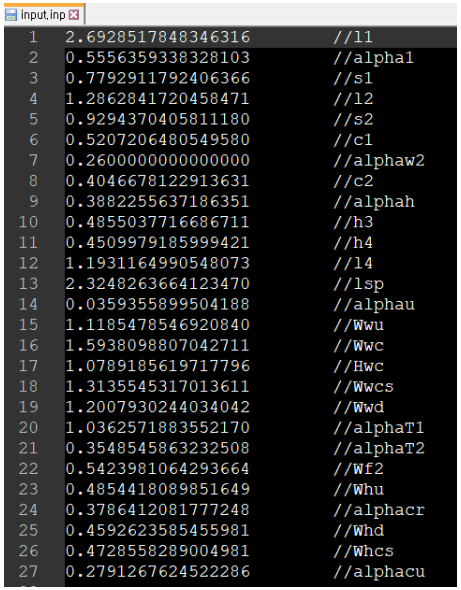
- 형상매개변수 표본 선정 (GIST와 협업)
  - 27개 매개변수로부터 전두부 CAD를 출력하는 프로그램을 GIST에 송부
    - input.inp 파일 편집 후 exe 파일 실행



  - 매개변수의 범위 내 변동에 따른 형상 변화를 조사하여 해석 표본 선정
    - 비정상적인 형상 제외
  - 선정된 해석 표본 (500개) 을 넥스트폼이 전달받아 반복 해석 수행

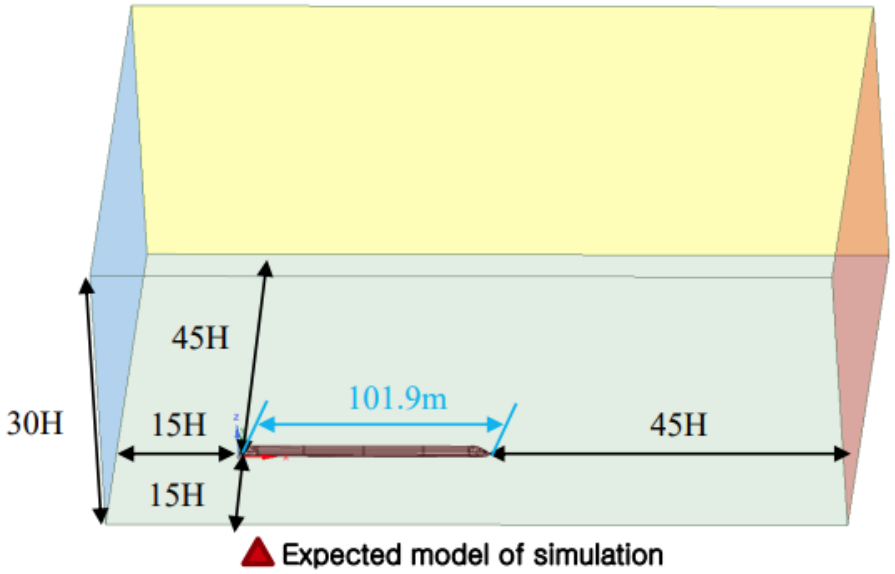






CFD 해석 및 DB 구축

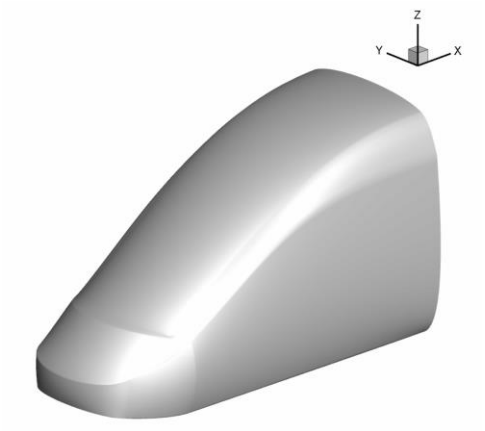
- 해석 문제 정의
  - 열차 길이 : T(26.65m)+M(24.30m)+M+T 구조, 101.9m 고정
  - 원방 길이 : 60H (W) X 30H (H) X 60H + 101.9m (L)
    - 여기서, H는 열차 높이
  - 개활지 해석의 경우 symmetry를 적용한 ½ 크기의 해석 케이스 사용
  - 주행속도 400km/h + 측풍 40m/s
  - 경계조건
    - 개활지 : inlet/outlet/sym
    - 측풍 : inlet/outlet

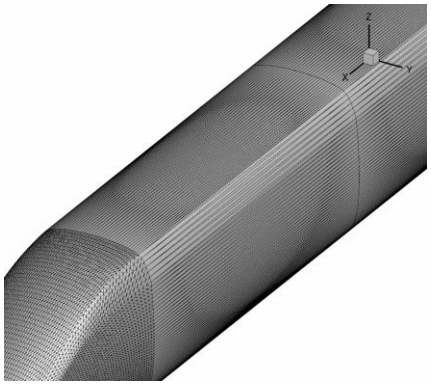


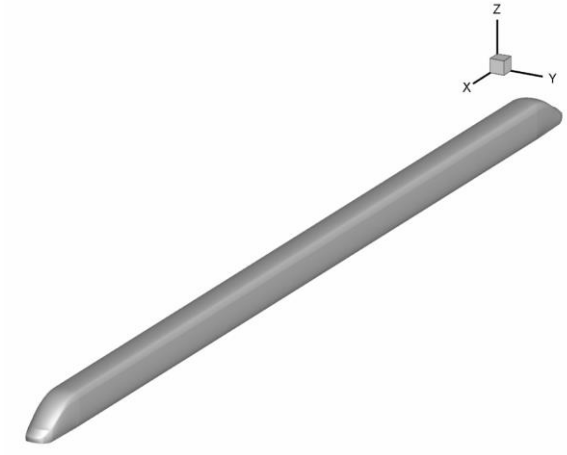
CFD 해석 및 DB 구축

- 열차 CAD 생성
  - 전두부 CAD로부터 T+M+M+T 구조 CAD를 생성하는 코드 작성
    - 최전방 지점 + 26.65 (T) + 24.30 (M) 기준 대칭
    - 연결부는 임의로 구성
    - 전두부 CAD가 교체되어도 바로 적용 가능

maketrain.cpp
maketrain.hpp

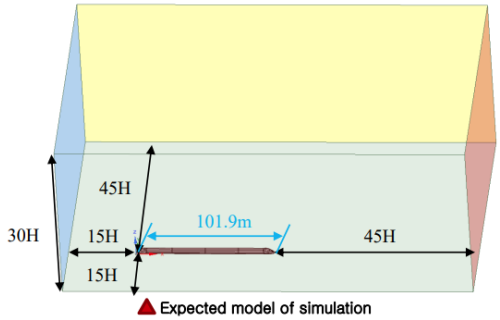






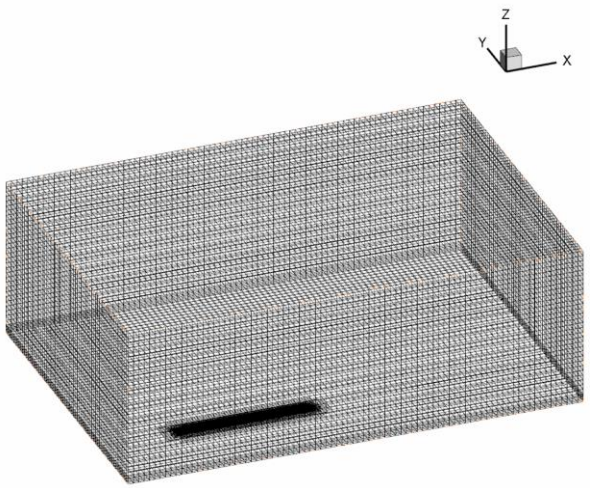
CFD 해석 및 DB 구축

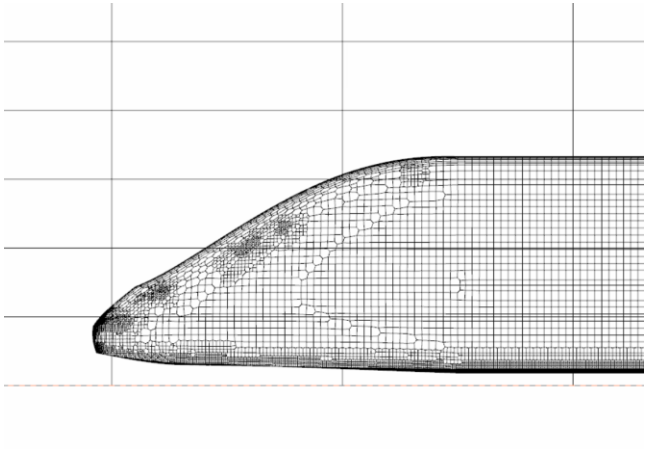
- 격자 생성 (blockMesh + snappyHexMesh)
  - 격자 개수 100~200만개 수준
  - 6코어 기준 250.76초 소요




```
mesh : cells:2225960  faces:6979173  points:2535735
```

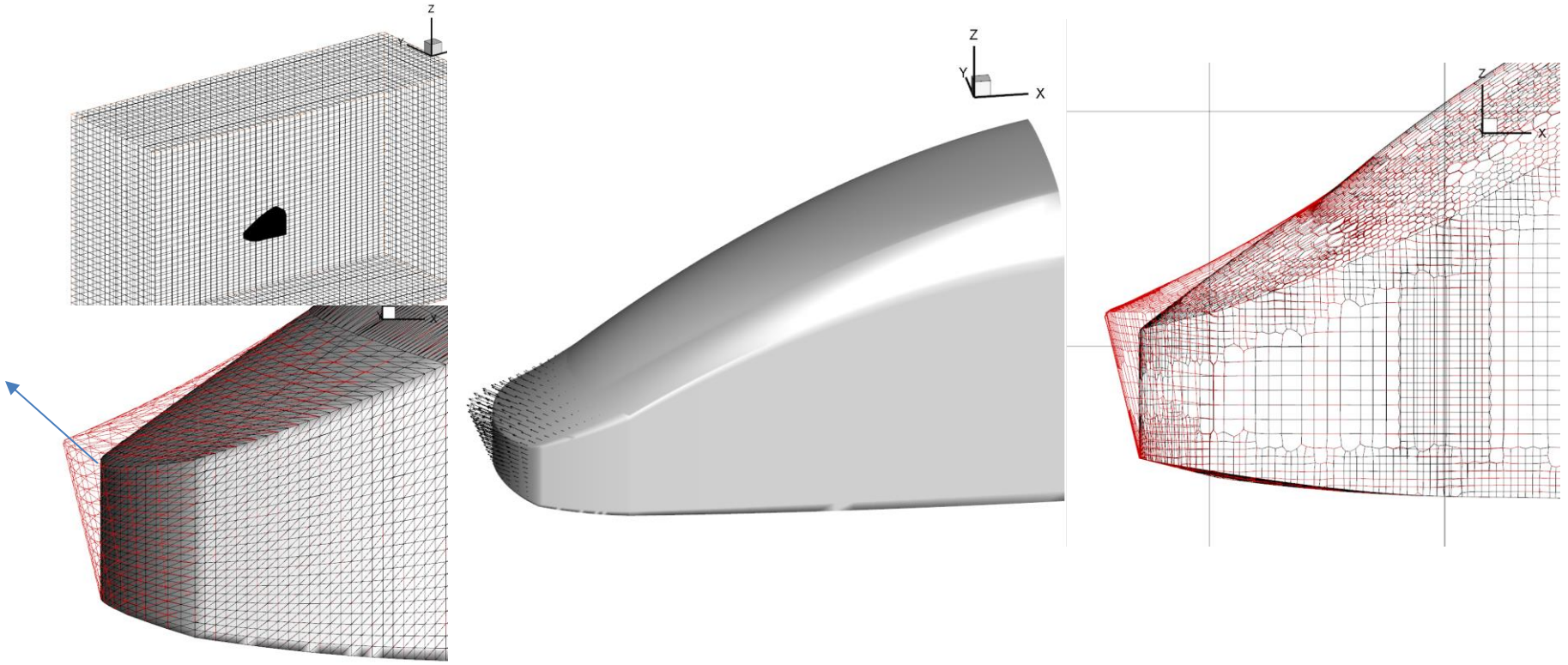
```
Finished meshing without any errors
Finished meshing in = 250.76 s.
```





CFD 해석 및 DB 구축

- deformMesh
  - 기본 격자계 생성 (최초 1회)
  - 형상 매개변수 값 변경으로 인한 CAD 형상 차이 발생
  - Bezier surface로부터 계산되는 u,v 위치별 형상 변위를 deformMesh input으로 작성
  - 격자 변형 확인



CFD 해석 및 DB 구축

- 반복해석 수행
  - Python 스크립트로 500개 케이스 연속 실행

```
[foamer@node4 231009_sizetest]$ cat runPara.py
import sys
import os

if len(sys.argv) < 2:
        exit()

i = int(sys.argv[1]) + 150
while 150 < i <= 250:
        os.system("cp -r ./new-solver ./para_%s && cp ./Results/output_%s.txt ./para_%s/STLGen/input.inp && cd ./para_%s
 && ./run | tee logpara.log" % (i, i, i, i))
        i += 6
```

```
cd STLGen
./parameterbezier
./maketrain
cd
mv STLGen/currentSTL.stl constant/triSurface/motorBike1.stl
mv STLGen/bridgeList1.stl constant/triSurface/motorBike2.stl
mv STLGen/bridgeList2.stl constant/triSurface/motorBike3.stl
mv STLGen/reflectedTriangles.stl constant/triSurface/motorBike4.stl
rm STLGen/*.stl
surfaceFeatureExtract
blockMesh
[ ! -d 0 ] && cp -r 0.orig 0
snappyHexMesh -overwrite
patchSummary
decomposePar -force
mpirun -np 8 buoyantSimpleNFoam -parallel
```

  - 최초 1개 케이스는 snappyHexMesh로 격자가 생성되어 있음
    - 이후 케이스에서는 격자를 복제하여 사용
  - CAD 생성
  - deformMesh
    - 기존 생성되어 있던 격자를 CAD간의 변위차를 기반으로 변형
  - 해석조건 부여 및 OpenFOAM solver (buoyantSimpleNFoam) 실행
  - 일정 개수 해석케이스 누적 시 적합직교분해 ROM 생성, 해석 초기조건을 추정하여 수렴 가속
    - potentialFoam과 유사, 자유류 초기조건 대비 적은 iteration으로 수렴 가능

  ……

  - deformMesh의 성능 불안정
    - 1개 케이스 격자 변형에 2시간 이상 소요 (격자 생성에 4분 소요되는 해석케이스)
    - 병렬화 여부, 시간복잡도 등 원인 검토 시도
    - 과제 기간 내 해결 여부 불투명

CFD 해석 및 DB 구축

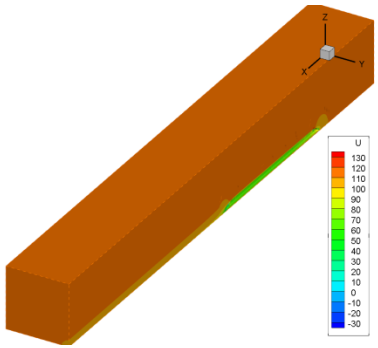
- 반복해석 수행
  - Python 스크립트로 500개 케이스 연속 실행

```
[foamer@node4 231009_sizetest]$ cat runPara.py
import sys
import os

if len(sys.argv) < 2:
        exit()

i = int(sys.argv[1]) + 150
while 150 < i <= 250:
        os.system("cp -r ./new-solver ./para_%s && cp ./Results/output_%s.txt ./para_%s/STLGen/input.inp && cd ./para_%s
 && ./run | tee logpara.log" % (i, i, i, i))
        i += 6
```

```
cd STLGen
./parameterbezier
./maketrain
cd ..
mv STLGen/currentSTL.stl constant/triSurface/motorBike1.stl
mv STLGen/bridgeList1.stl constant/triSurface/motorBike2.stl
mv STLGen/bridgeList2.stl constant/triSurface/motorBike3.stl
mv STLGen/reflectedTriangles.stl constant/triSurface/motorBike4.stl
rm STLGen/*.stl
surfaceFeatureExtract
blockMesh
[ ! -d 0 ] && cp -r 0.orig 0
snappyHexMesh -overwrite
patchSummary
decomposePar -force
mpirun -np 8 buoyantSimpleNFoam -parallel
```

  - 적합직교분해 (POD) 적용 보류 (과제 종료 후 재시도)
    - 유동장 공간데이터의 실시간 예측은 과제 주 목표가 아니므로 보류
    - 대신 일부 후처리 항목 (v-wall, 열차 표면압력분포) 에 추후 적용

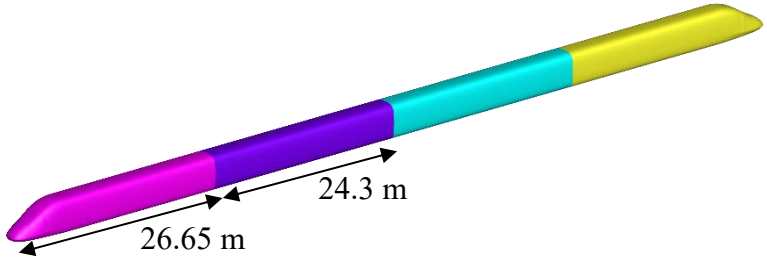
  - CAD 생성
  - snappyHexMesh로 매 케이스마다 신규 격자 생성 (250s/case)
  - 해석조건 부여 및 OpenFOAM solver (buoyantSimpleNFoam) 실행



- 해석 소요 시간(8코어 기준) 및 데이터 용량
  - 개활지 주행 : 2.5m mesh, 4h/1case, 500case * (4h/1case) / (12 nodes) = 167h, 850GB
  - 측풍 : 5m mesh, 8h/1case, 500case * (8h/1case) / (12 nodes) = 334h, 1.7TB

후처리 및 최적화

- 후처리 항목 (1/4)
  - 차량별 공력계수 (항력, 전도모멘트)
    - OpenFOAM 내장 forceCoeffs utility 출력값 취합
    - 낮을수록 공력 성능 우수



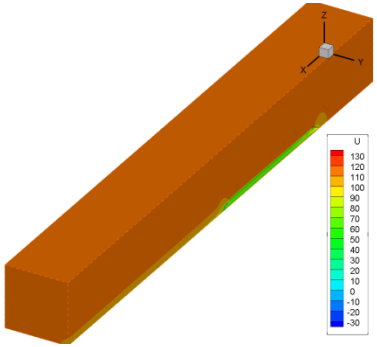

```
resultlines = []
resultlines.append('i Cd_Tc1 Cd_M1 Cd_M2 Cd_Tc2\r\n')
for i in range(500):
    dirPost = './para_%d/postProcessing' % (i+1)
    if os.path.isdir(dirPost) is False:
        continue

    templine = []
    templine.append(str(i+1))
    for j in range(4):
        dirCoefFile = dirPost + '/forceCoeffs_force-%d' % j + '/0/coefficient.dat'
        finput = open(dirCoefFile, 'r')
        lines = finput.readlines()
        finput.close()
        endLine = lines[-1].rstrip('\r\n').split()
        templine.append(endLine[1])

    resultlines.append(' '.join(templine) + '\r\n')

foutput = open('collectedCd.dat', 'w')
foutput.writelines(resultlines)
foutput.close()
```

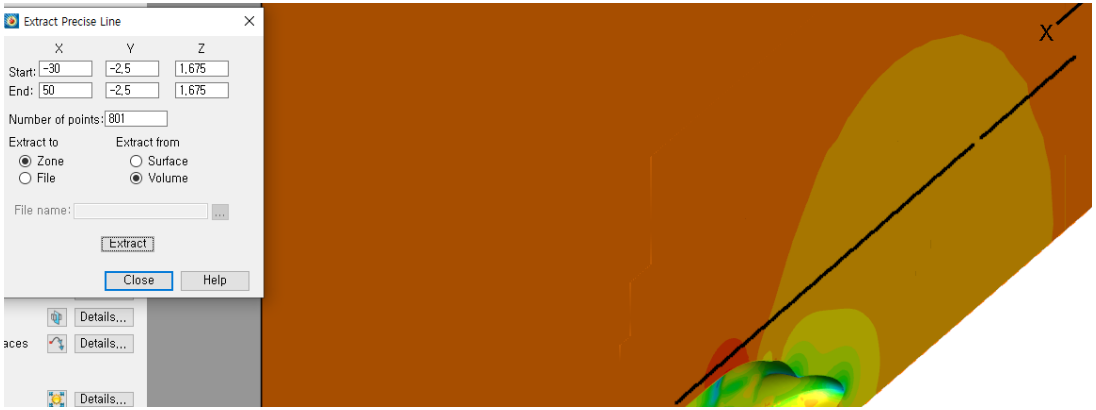
collectedCd.dat

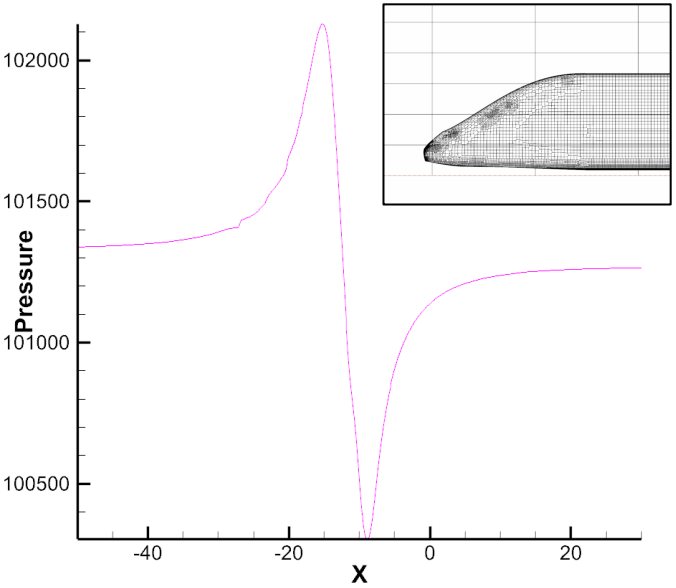
1	i	Cd_Tc1	Cd_M1	Cd_M2	Cd_Tc2
2	1	7.356407e+00	2.244105e-02	2.146825e-02	-7.284230e+00
3	2	7.356407e+00	2.241767e-02	2.142859e-02	-7.287253e+00
4	3	7.356625e+00	2.239833e-02	2.140122e-02	-7.285311e+00
5	4	7.357385e+00	2.245570e-02	2.143472e-02	-7.286323e+00
6	5	7.356197e+00	2.240242e-02	2.140487e-02	-7.288527e+00
7	6	7.356013e+00	2.250666e-02	2.150436e-02	-7.286878e+00
8	7	7.355967e+00	2.248750e-02	2.148072e-02	-7.288834e+00
9	8	7.356200e+00	2.248381e-02	2.148149e-02	-7.289098e+00
10	9	7.355922e+00	2.250629e-02	2.151024e-02	-7.287460e+00
11	10	7.356151e+00	2.242639e-02	2.142878e-02	-7.286494e+00
12	11	7.356608e+00	2.240264e-02	2.140700e-02	-7.285846e+00
13	12	7.356353e+00	2.241855e-02	2.142174e-02	-7.287726e+00
14	13	7.355880e+00	2.245557e-02	2.145376e-02	-7.288331e+00
15	14	7.356306e+00	2.242228e-02	2.142735e-02	-7.285079e+00
16	15	7.356087e+00	2.246646e-02	2.148743e-02	-7.287317e+00
17	16	7.356634e+00	2.251555e-02	2.150822e-02	-7.283047e+00
18	17	7.357026e+00	2.241625e-02	2.140565e-02	-7.284800e+00
19	18	7.356619e+00	2.241024e-02	2.140855e-02	-7.283485e+00
20	19	7.356144e+00	2.248727e-02	2.150089e-02	-7.287282e+00

후처리 및 최적화

- 후처리 항목 (2/4)
  - 전두부 주위 압력변화 데이터
    - Tecplot – extract precise line + macro 사용 (케이스 당 5~10분 소요)
    - 압력 변동폭이 작을수록 공력 성능 우수





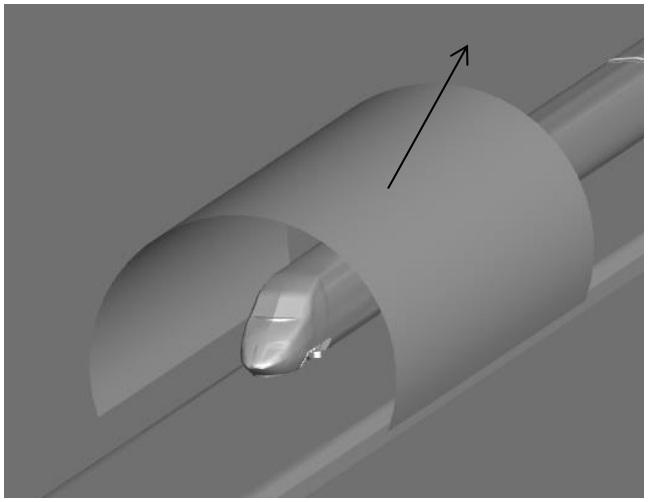


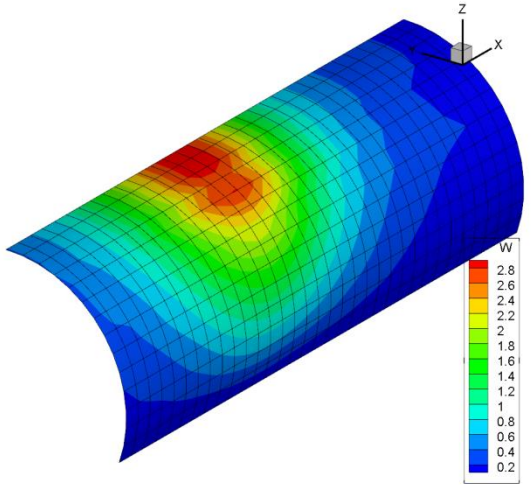
generatedMCR.mcr

```
#!MC 1410
$!VarSet |MFBD| = 'C:\Program Files\Tecplot\Tecplot 360 EX 2017 R2'
$!READDATASET  '"StandardSyntax" "1.0" "FEALoaderVersion" "436" "FILENAM
  DATASETREADER = 'OpenFOAM (FEA)'
$!GLOBALTIME SOLUTIONTIME = 500
$!EXTENDEDCOMMAND
  COMMANDPROCESSORID = 'Extract Precise Line'
  COMMAND = 'XSTART = -50 YSTART = -2.5 ZSTART = 1.675 XEND = 30 YEND =
$!PAGE NAME = 'Untitled'
$!PAGECONTROL CREATE
$!NEWLAYOUT
$!READDATASET  '"StandardSyntax" "1.0" "FEALoaderVersion" "436" "FILENAM
  DATASETREADER = 'OpenFOAM (FEA)'
$!GLOBALTIME SOLUTIONTIME = 500
$!EXTENDEDCOMMAND
  COMMANDPROCESSORID = 'Extract Precise Line'
  COMMAND = 'XSTART = -50 YSTART = -2.5 ZSTART = 1.675 XEND = 30 YEND =
$!PAGE NAME = 'Untitled'
$!PAGECONTROL CREATE
$!NEWLAYOUT
$!READDATASET  '"StandardSyntax" "1.0" "FEALoaderVersion" "436" "FILENAM
```

후처리 및 최적화

- 후처리 항목 (3/4)
  - V-wall data
    - 고속열차의 터널 통과 시 방사되는 미기압파 세기 예측 필요
    - Unsteady 해석을 하는 것이 자연스러우나 계산 부하 증가
    - Steady 해석 결과만으로 미기압파 세기를 예측할 수 있는 기법으로 제시됨 (Ogawa, 1995)
    - 전두부 주위 가상벽면 설정, 해당 벽면에서의 수직 속도 면적분 계산
    - Tecplot interpolation 기능 및 매크로 사용 => 가상벽면에서의 속도장 추출
    - V-wall 계산 python script 작성
    - V-wall_max 값이 낮을수록 공력 성능 우수



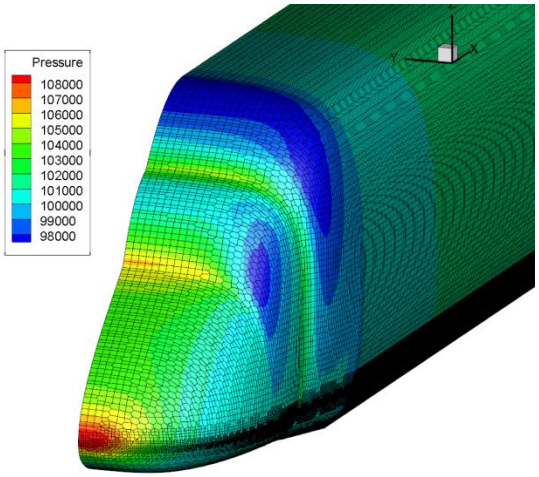


ssCase.py | vwall_data_1.dat

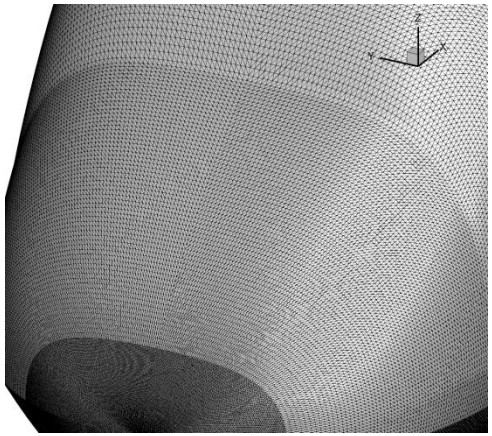
```
V_wall
0.3634993796111509
0.49414830798437387
0.5394528597747201
0.6594630667495659
0.7870266683707354
0.9619956587887156
1.1755625227035984
1.4396865819884386
1.7345833653433345
2.049755117708129
2.354778729527399
2.6097341308306152
2.7626938033451327
2.9186646304693924
2.799356919181012
2.551765417250743
2.2183510744326203
1.853947690093459
1.503638616275351
1.1983705727343772
0.9475018615270234
0.7495735696060536
0.5961790913653215
0.4788134075859955
0.38875928646199776
0.3191542143210275
0.26481608028920983
0.2220858154574486
0.18834910320513057
0.16138138883119724
0.13989365551411587
0.12248381574325662
0.1083073568512302
0.09641328540818457
V_wall_max
2.9186646304693924
```

후처리 및 최적화

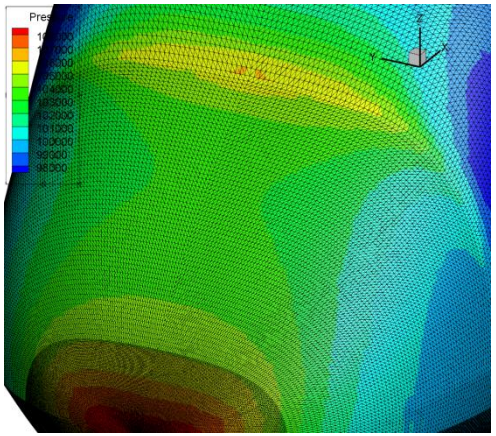
- 후처리 항목 (4/4)
  - 열차 표면압력분포
    - 해석 결과로 출력된 표면압력분포는 snappyHexMesh에 의해 임의 개수/형태의 격자로 변형된 상태
    - 500개 케이스 모두 동일한 topology의 표면격자에 데이터 출력 필요
    - CAD 생성 프로그램에서 출력되는 STL 파일 활용
      - 모두 동일한 Topology
    - Tecplot macro 작성
      - OpenFOAM 케이스 로드 / 대칭 복사 / STL 파일 로드 / interpolation / 파일 출력 500케이스 반복 실행
    - 공력 성능 평가 시 참고자료로 활용



CFD 해석 결과
(임의 topology)



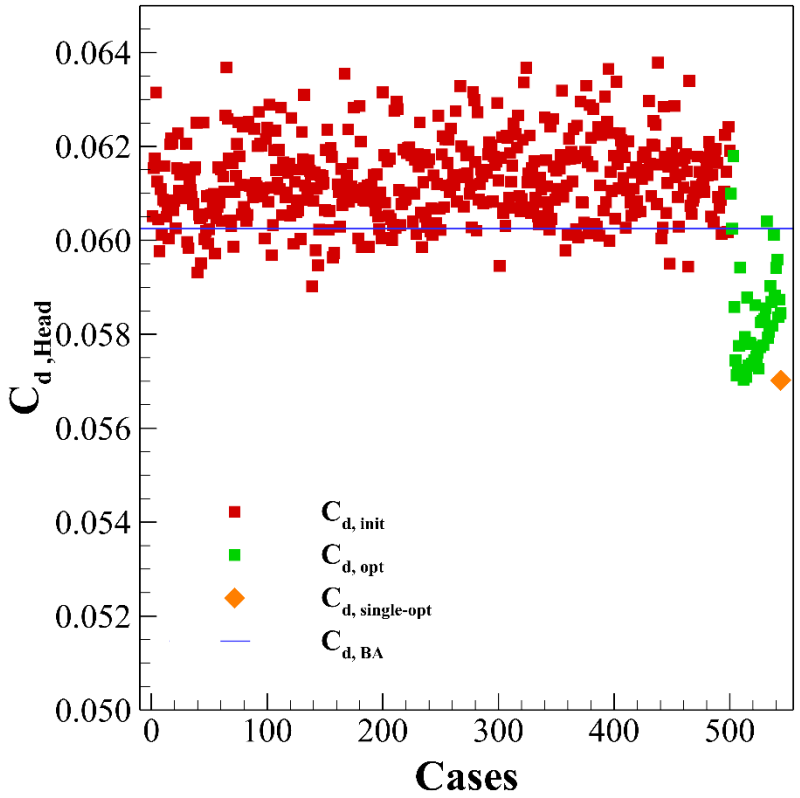
CAD 기반 격자
(통일된 topology)

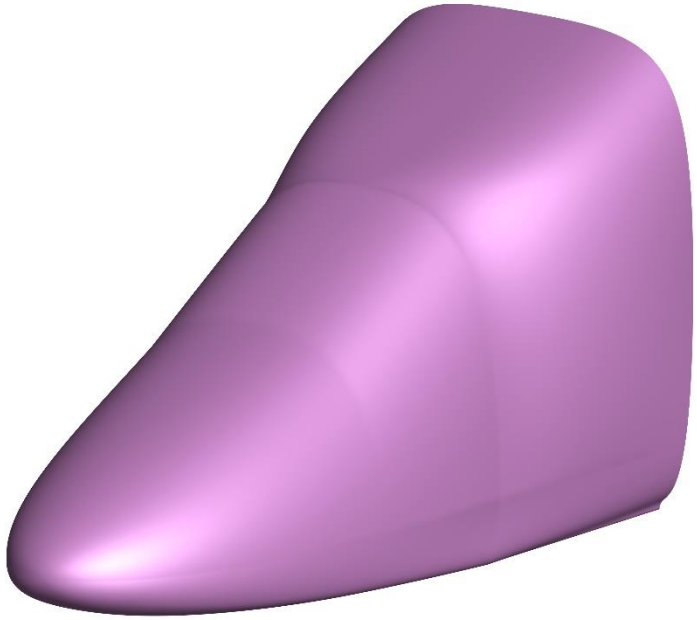


내삽된 표면압력분포

후처리 및 최적화

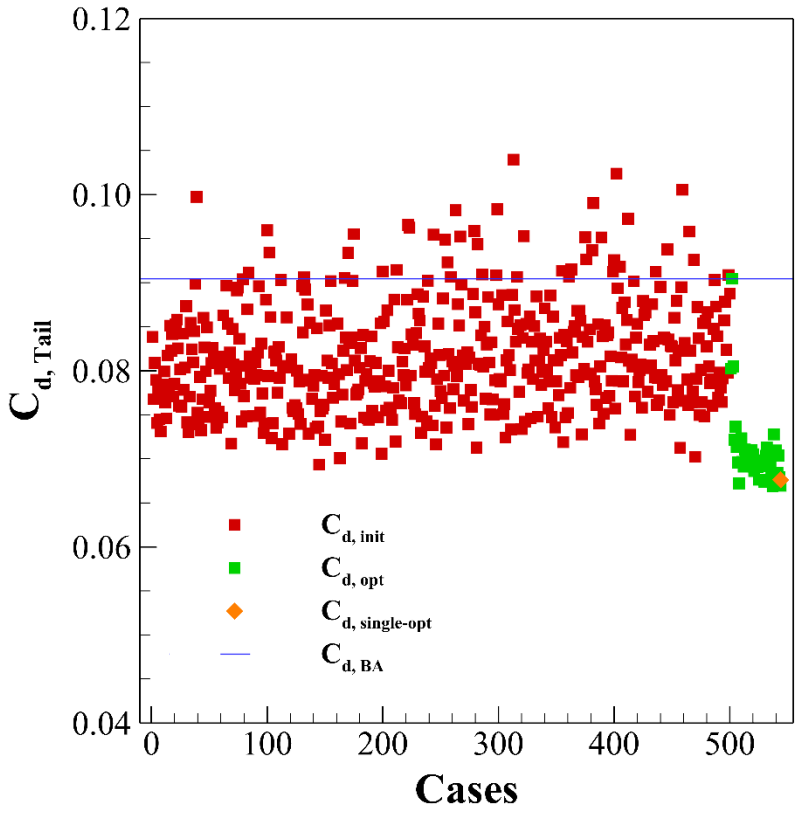
- 표본 학습 및 단일 목적 최적화 (NSGA-II) 결과
  - Cd, Head (BA 0.0603, Cd single opt = 0.057, 5.36% 저감)

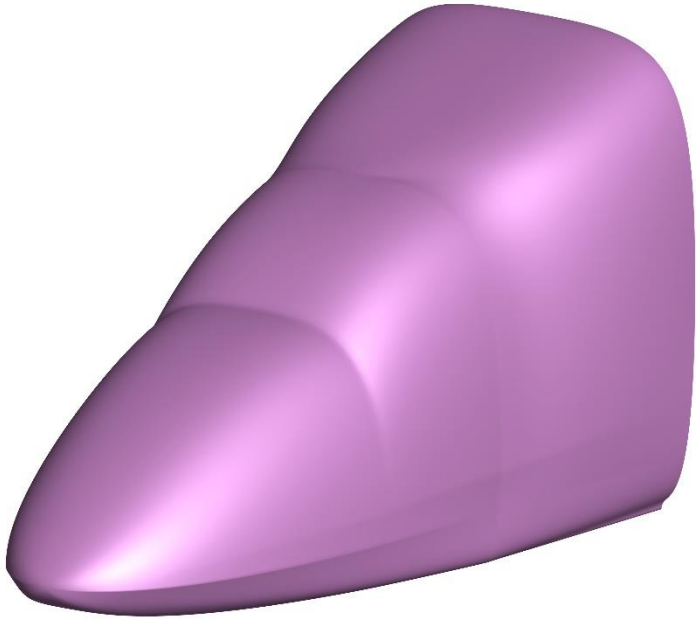




후처리 및 최적화

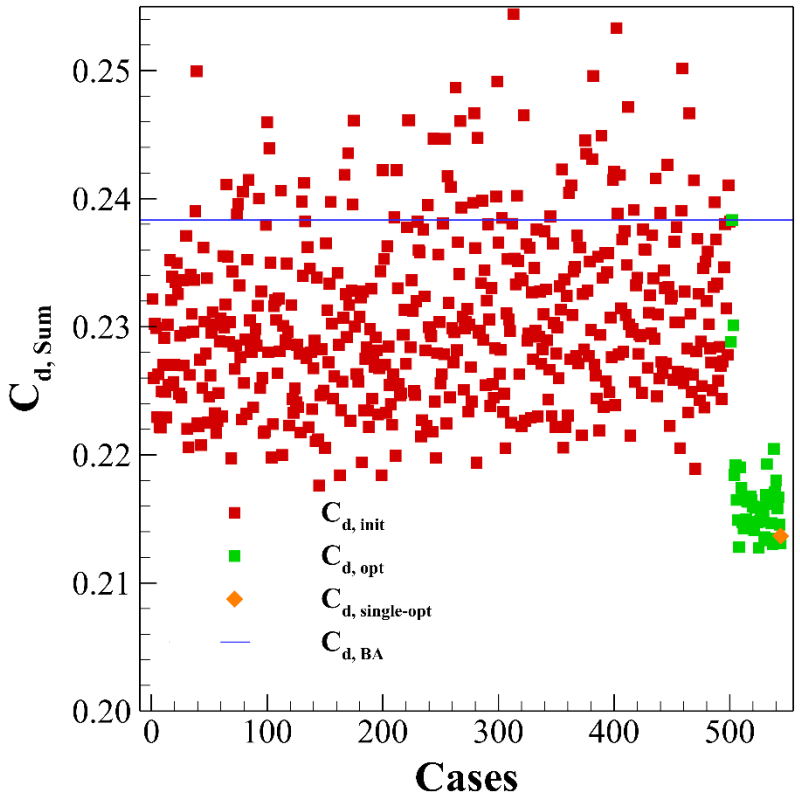
- 표본 학습 및 단일 목적 최적화 (NSGA-II) 결과
  - Cd, Tail (BA 0.0905, Single-opt 0.0676, 25.26% 저감

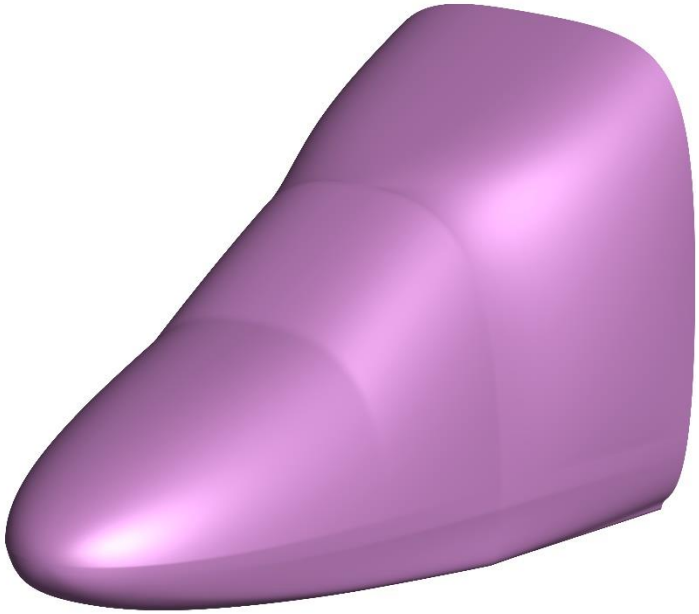




후처리 및 최적화

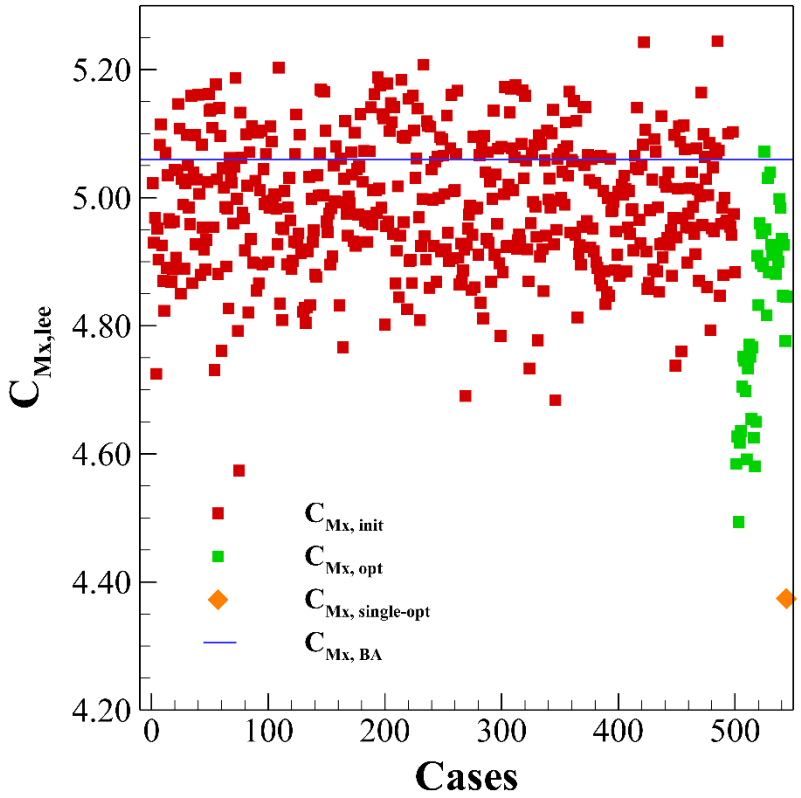
- 표본 학습 및 단일 목적 최적화 (NSGA-II) 결과
  - Cd, All (BA 0.2383, Single-opt 0.2137 10.35% 저감)

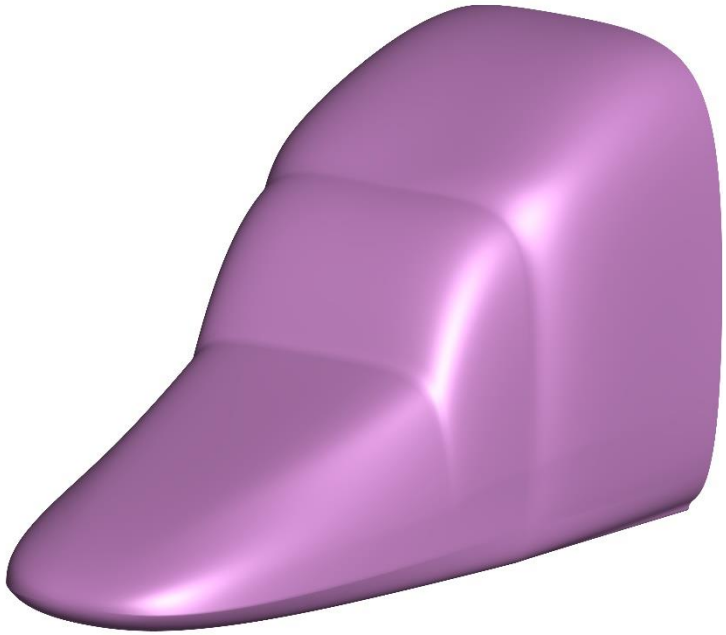




후처리 및 최적화

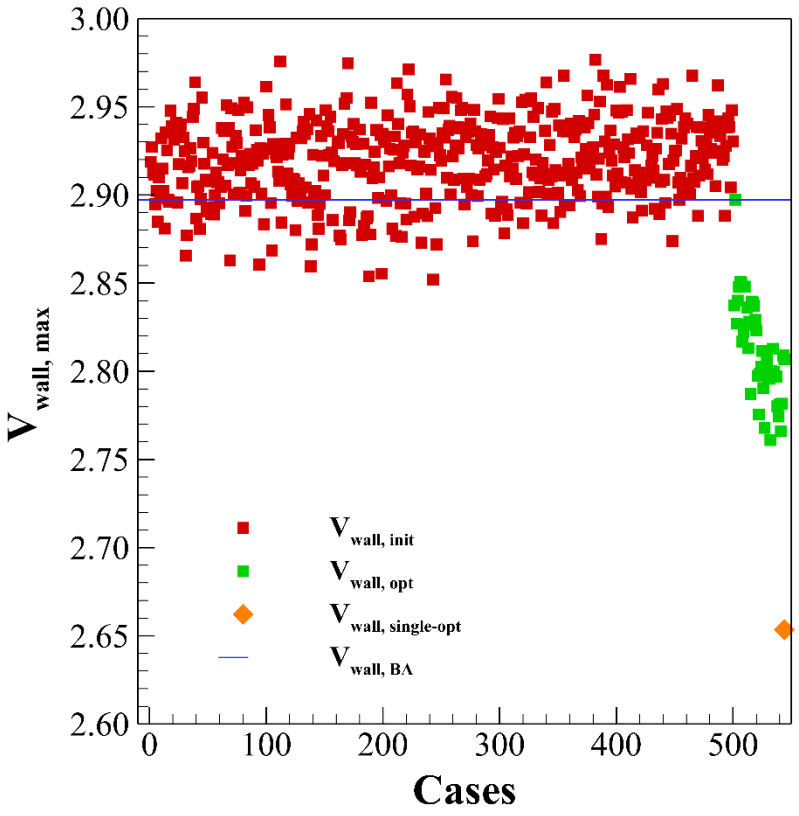
- 표본 학습 및 단일 목적 최적화 결과
  - CMx (BA 5.0597, Single-opt 4.3741 13.55% 저감)

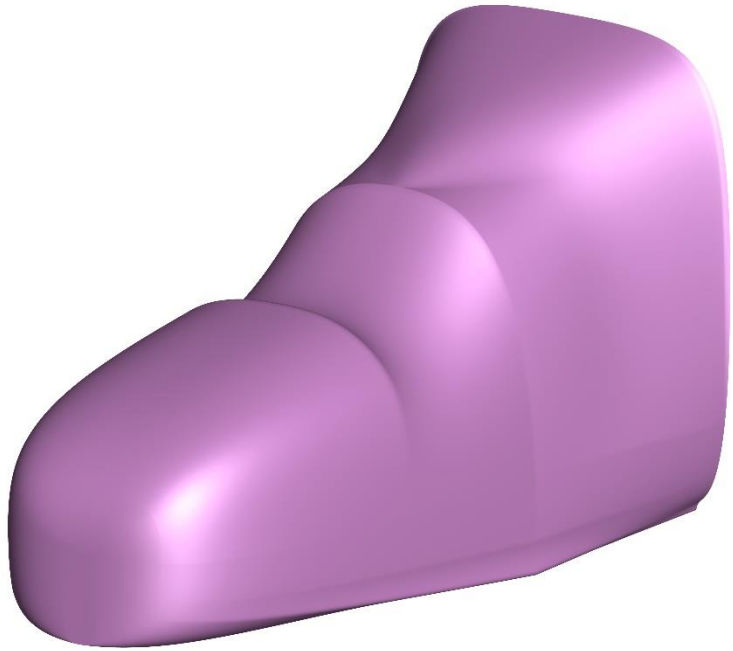




후처리 및 최적화

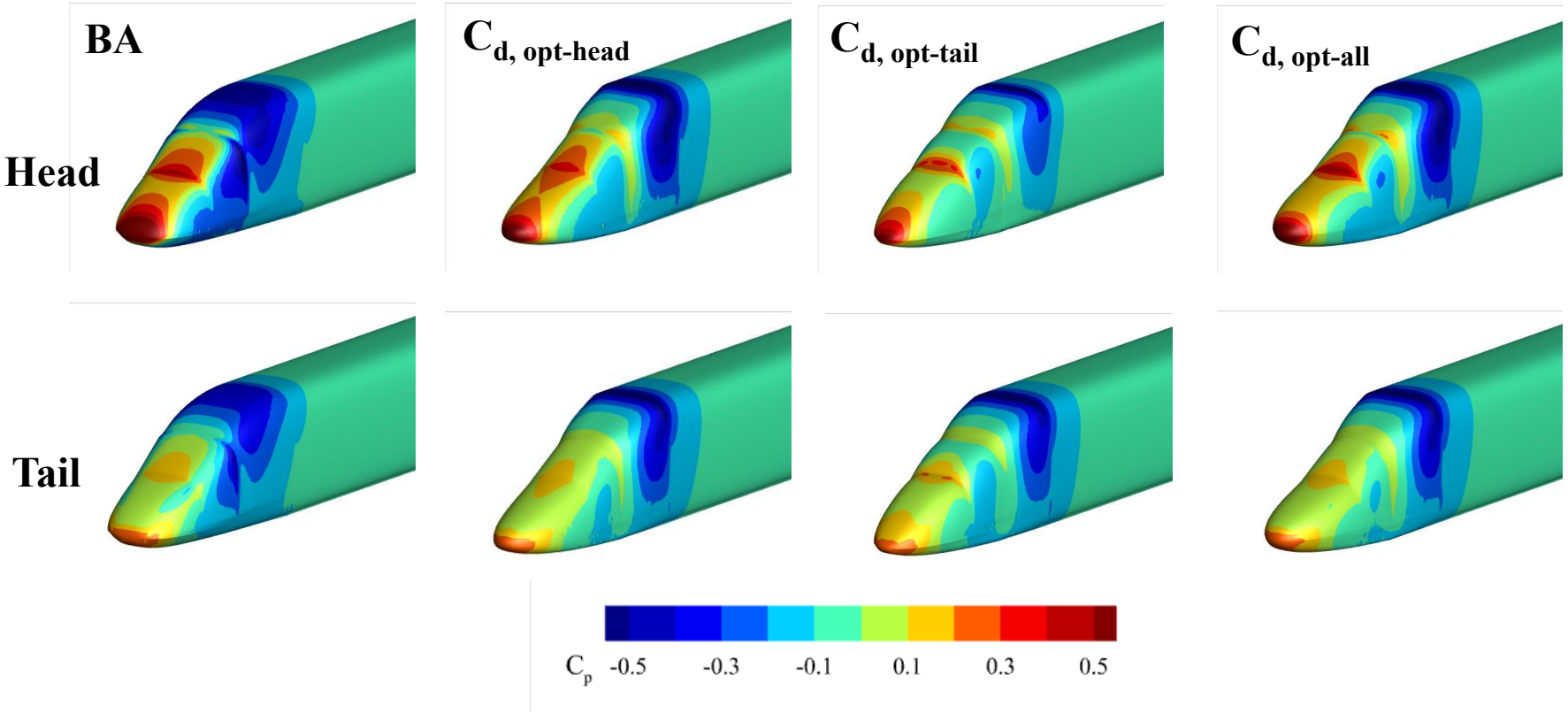
- 표본 학습 및 단일 목적 최적화 (NSGA-II) 결과
  - Vwall,max (BA 2.8973, single-opt 2.6534, 8.42% 저감)
  - 신칸센 E5계 전두부와 유사한 미기압파 분할 구조





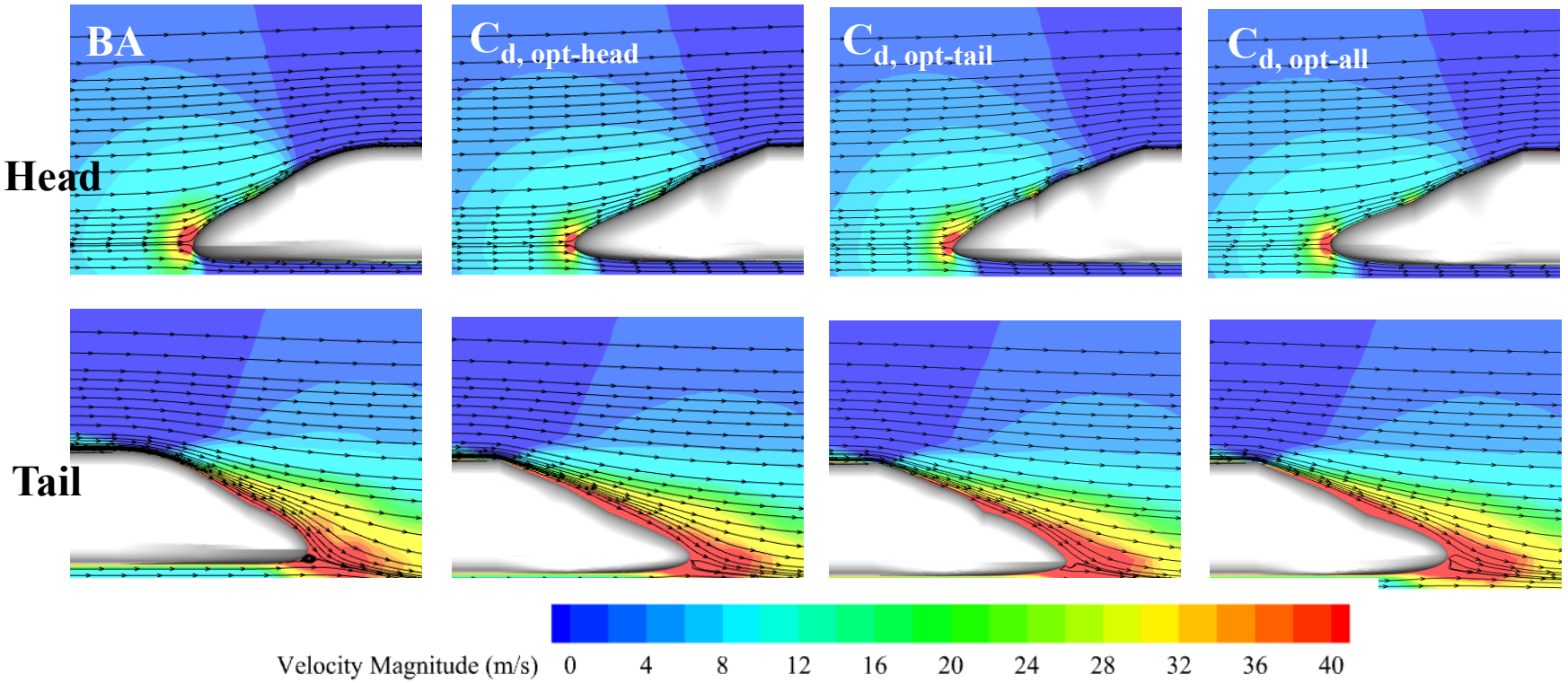
후처리 및 최적화

- Cd 최적화 케이스 분석
  - 표면압력 분석



후처리 및 최적화

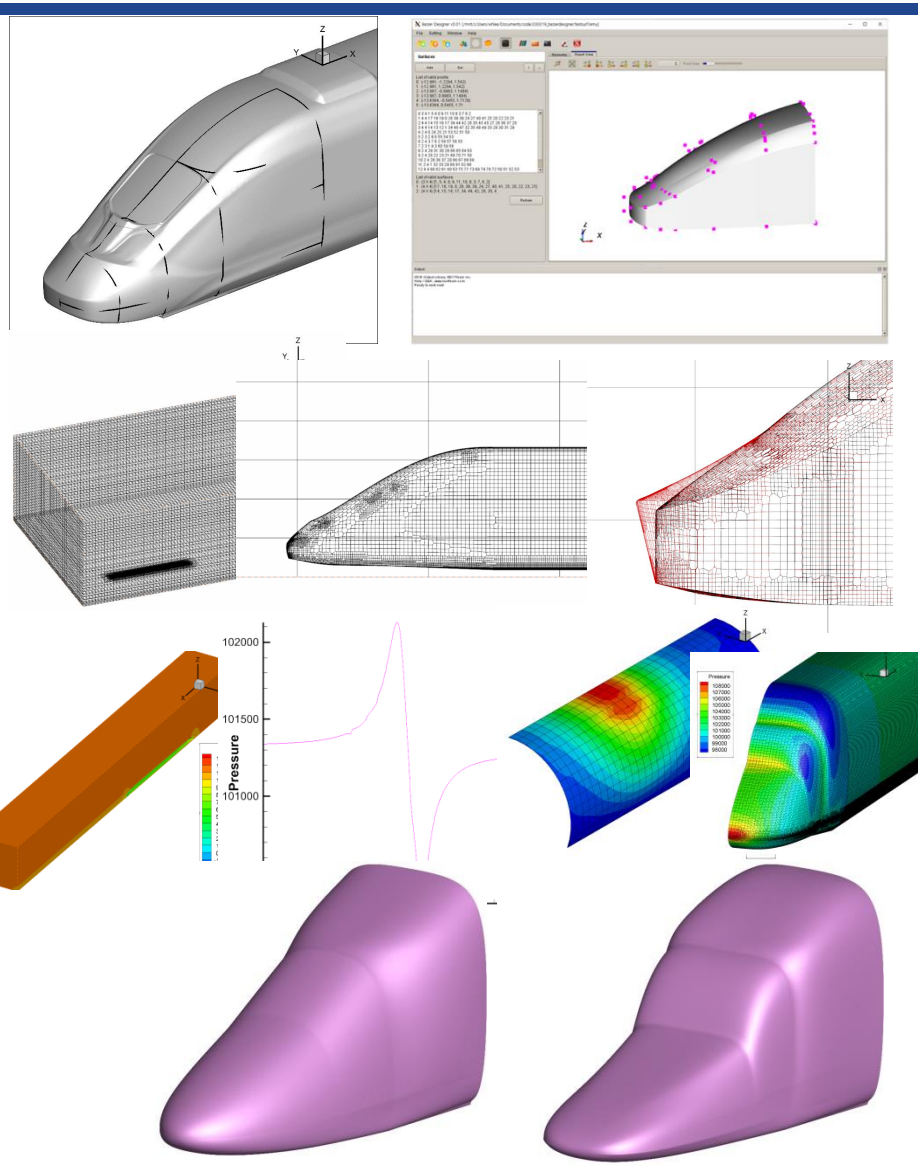
- Cd 최적화 케이스 분석
  - 표면압력 분석



과제 내용 요약

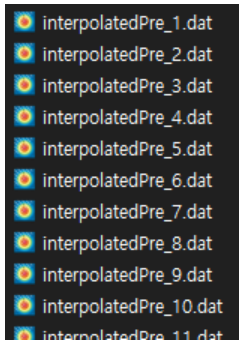
- 과제 기간 내 진행 사항
  - 전처리 자동화
    - Bezier surface를 이용한 전두부 형상 모사
    - CAD 생성 (별도 프로그램, TUI, GUI)
    - 공간격자 생성 (snappyHexMesh)
    - ~~격자 변형 (deformMesh)~~
  - 유동해석 자동화
    - 500개 표본 선정
    - OpenFOAM solver (buoyantSimpleNFoam)
  - 후처리 자동화
    - 공력계수
    - 전두부 주위 압력변화 데이터 (line extract)
    - V-wall 데이터
    - 표면압력분포
  - 최적설계
    - 단일 목적 & 다목적 최적화
    - 최적설계안 도출

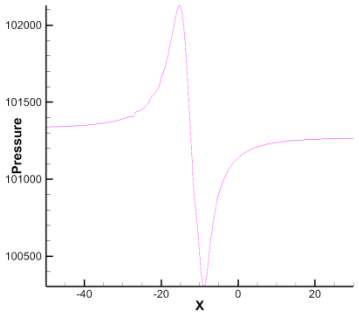
- 과제 종료 후
  - 적합직교분해 (POD) 활용 최적화 및 비교

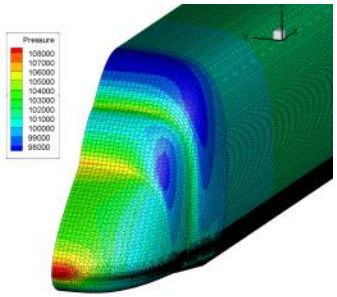


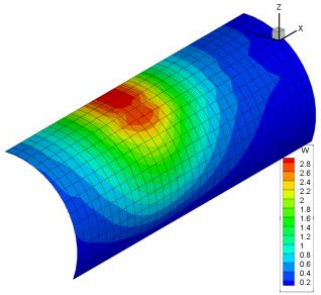
적합직교분해 (POD)

- 추가 연구 - 고속열차 해석 데이터에 적합직교분해 적용
  - 3차원 유동장 전체에 대한 기법 적용은 deformMesh 유틸리티 문제로 실패하였으나, 후처리 항목 중 1d / 2d 등 일정한 topology로 정리/출력되는 데이터에 적용 가능
    - 오히려 공력성능과 직접적으로 연관되는 데이터들





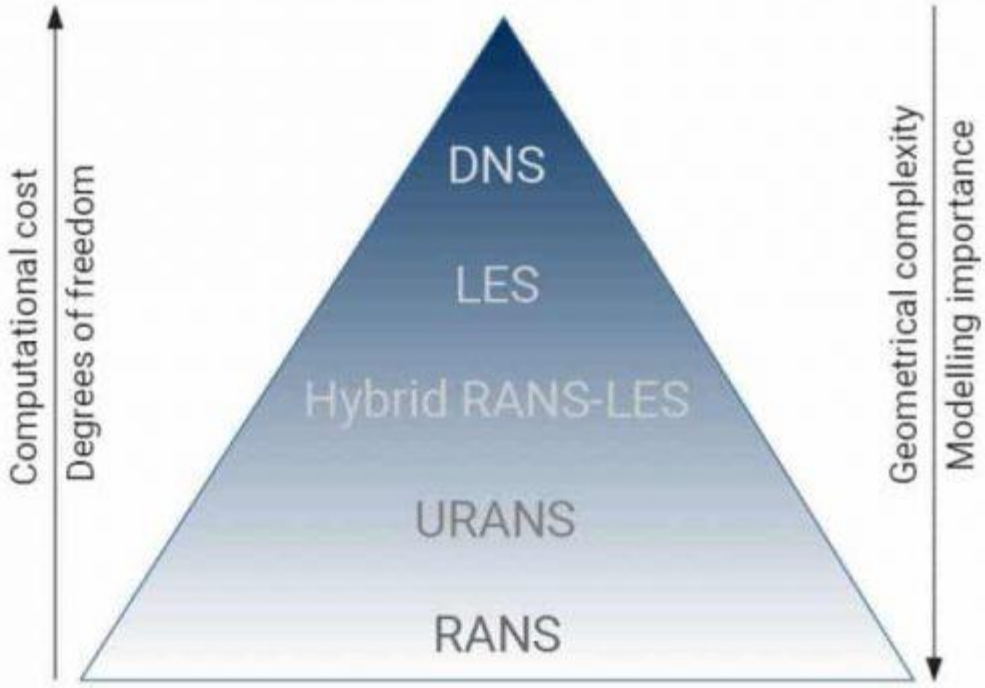




  - 적합직교분해 기반 형상최적설계 테스트 수행
    - 과제 수행 결과로 존재하는 500개 샘플 해석 데이터 활용
    - 1개씩 임의로 추출하면서 가상 데이터셋 구성 및 적합직교분해, 차수축소모델 생성
    - 모델 평가를 위해 10개 샘플을 추가 추출하여 CFD vs ROM 비교
    - 생성된 모델로부터 최적 형상 도출

적합직교분해 (POD)

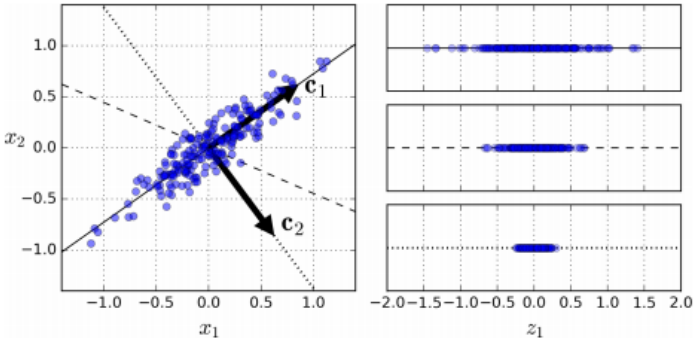
- 기법 개요
  - 전산유체역학 반복해석 수행 시
  - 기 존재하는 전산유체역학 해석 결과 사용, 최적 모드 및 계수 추출
  - 신규 조건에 대한 해석 결과를 예측
    - 예측된 해석 결과를 초기조건으로 부여하고 iteration을 돌려 수렴 가속 가능

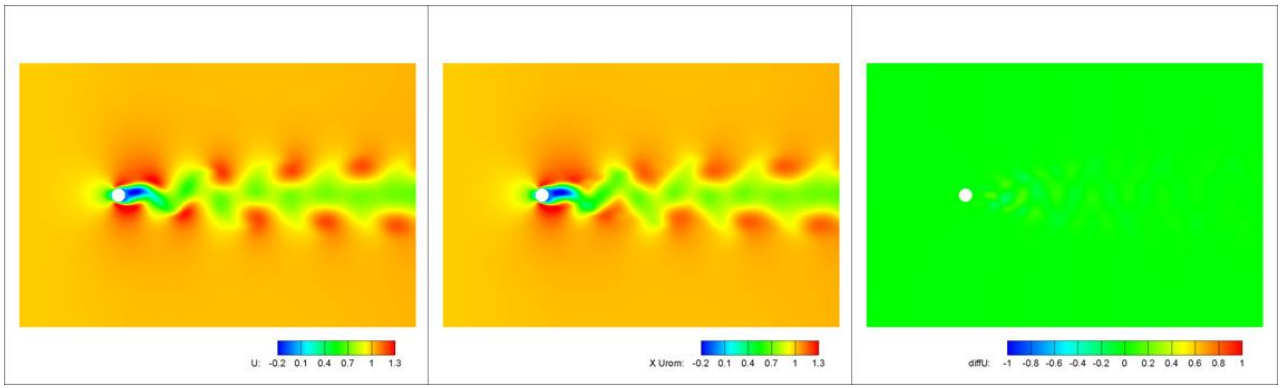


POD, Flow angle, Panel method, etc.

적합직교분해 (POD)

- 주성분 분석 (PCA) 기법 기반
- 특이치 분해 (SVD)
- 이미지 압축 등 광범위한 분야에 사용
- CFD 해석 결과에 적용 가능



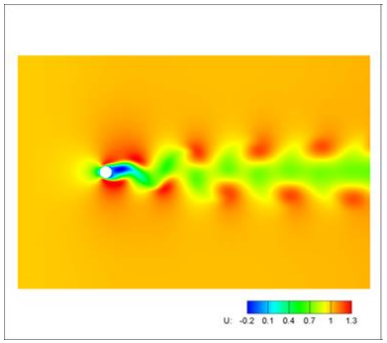


적합직교분해 (POD)

– 절차

- (1) 해석 결과 (snapshot) 확보

```
Documents/code/211215_pod_test$ ll

./
../
0 -> ../211214_stl_sonicfoam/motorBike_5/500/
10 -> ../211214_stl_sonicfoam/motorBike/500/
5 -> ../211214_stl_sonicfoam/motorBike_5/500/
8 -> ../211214_stl_sonicfoam/motorBike_8/500/
constant/
system/
```



- 입력 parameter 조건 별 해석 결과
  - AOA, BETA, Mach #, geometric parameter 등 종류 무관
  - 많을수록 정확도 ↑
- OpenFOAM 기반의 Open-Source POD 프로그램 사용 (AccelerateCFD)
- 단일 OpenFOAM 케이스 폴더로 취합
- N개 격자 * M개 시점 해석 결과가 취합된 N×M 행렬 Y

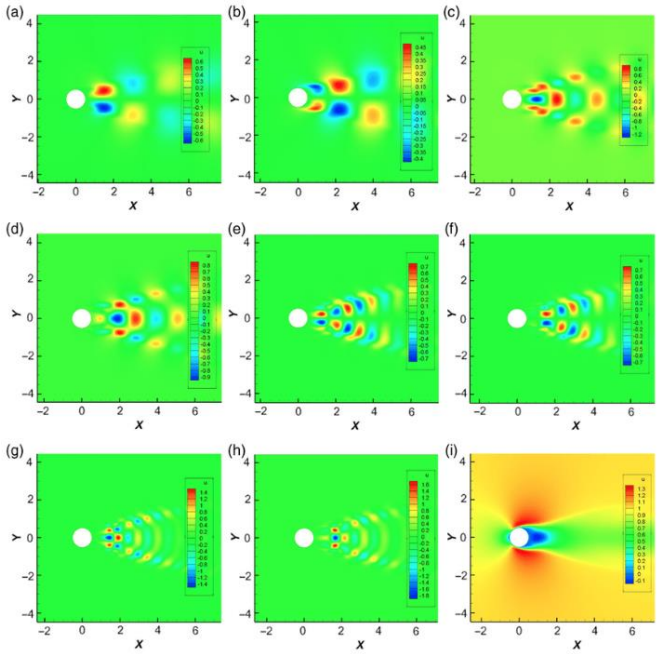
$$\overrightarrow{y_j} = \begin{bmatrix} y_{1j} \\ y_{2j} \\ \cdots \\ y_{Nj} \end{bmatrix} \qquad Y = \begin{bmatrix} \overrightarrow{y_1} & \overrightarrow{y_2} & \cdots & \overrightarrow{y_M} \end{bmatrix}$$

적합직교분해 (POD)

– 절차

- (2) 모드 (basis) 추출
  - 해석 결과의 기저 벡터
  - 선형결합으로 유동장 재현

02_SVD	• 공분산행렬(Covariance Martix) 생성 • Eigenvalue & Eigenvectors 추출 ✓ Left Eigenvector = POD mode (Basis function)	$R = X^T X$ $X = \begin{bmatrix} x_1(t_1) & \cdots & x_n(t_1) \\ \vdots & \ddots & \vdots \\ x_1(t_m) & \cdots & x_n(t_m) \end{bmatrix} \in R^{m \times n}$ $R\beta_m = \lambda_m \beta_m$



POD modes of the oscillating cylinder: (a) POD mode #1, (b) POD mode #2, (c) POD mode #3, (d) POD mode #4, (e) POD mode #5, (f) POD mode #6, (g) POD mode #7, (h) POD mode #8 and (i) average field.

$$Y = U\Sigma V^T$$

$$R = Y^T Y \quad R\vec{v_i} = \sigma_i^2 \vec{v_i}, \qquad \vec{u_i} = \frac{Y\vec{v_i}}{\sigma_i}$$

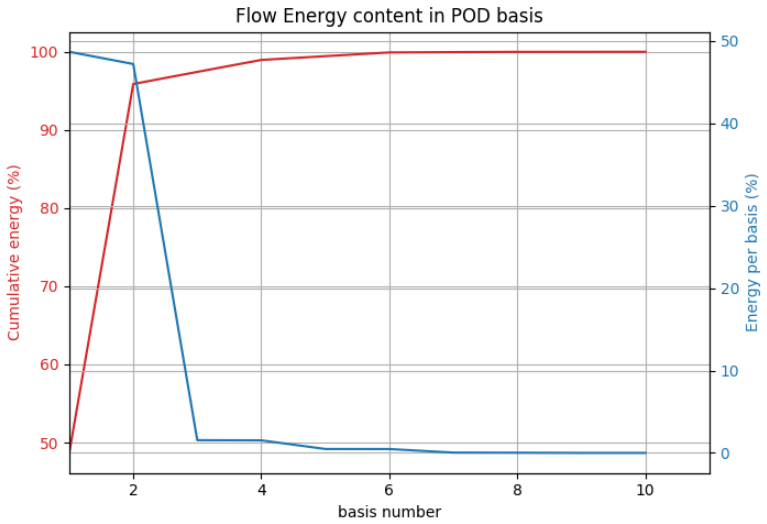
```
forAll(timeDirs, timei)
{
  n = 0;
  forAll(timeDirs, timej)
  {
     Cmn(m, n) = 0.0;
    // applying symmetry
    if (n < m)
    {
      Cmn(m, n) = Cmn(n, m);
      n++;
      continue;
    }

    volScalarField U1dotU2(generateCustomField(runTime,mesh,"U1dotU2"),
                           (vels[timei]&vels[timej])*cellVolume);

    Cmn(m,n) = Cmn(m,n) + gSum(U1dotU2);
    n++;
  }
  m++;
}

// Normalization of correlation matrix by dividing with total number of
Cmn = Cmn/nDim;

// Self Adjoint Eigen Solver is used here to solve for eigenvalue proble
Info<< "Solving eigenvalue problem" << nl;

Eigen::SelfAdjointEigenSolver<Eigen::MatrixXd> es(Cmn);
```

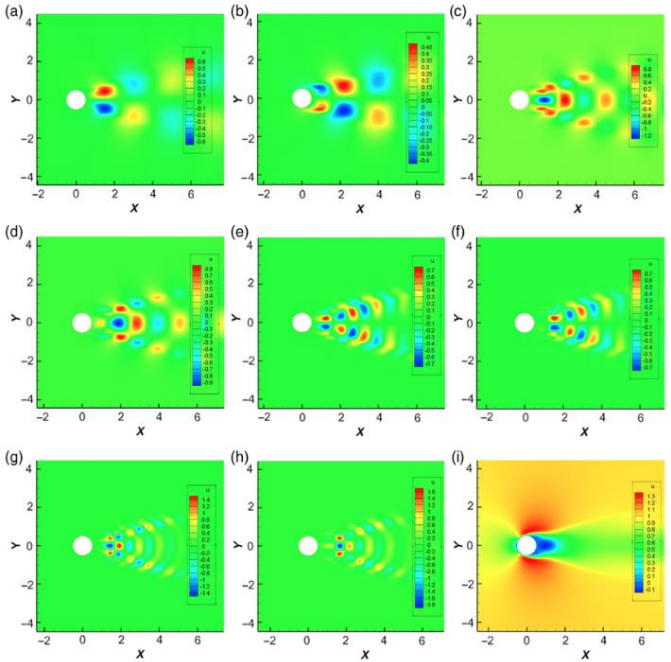
적합직교분해 (POD)

– 절차

- (2) 모드 (basis) 추출
  - 해석 결과의 기저 벡터
  - 선형결합으로 유동장 재현
  - 5~10개 모드에 99.9% 에너지 분포
  - 전체 해석 결과 요약/예측 가능

02_SVD	• 공분산행렬(Covariance Martix) 생성 • Eigenvalue & Eigenvectors 추출 ✓ Left Eigenvector = POD mode (Basis function)	$R = X^T X$ $X = \begin{bmatrix} x_1(t_1) & \cdots & x_n(t_1) \\ \vdots & \ddots & \vdots \\ x_1(t_m) & \cdots & x_n(t_m) \end{bmatrix} \in R^{m \times n}$ $R\beta_m = \lambda_m \beta_m$





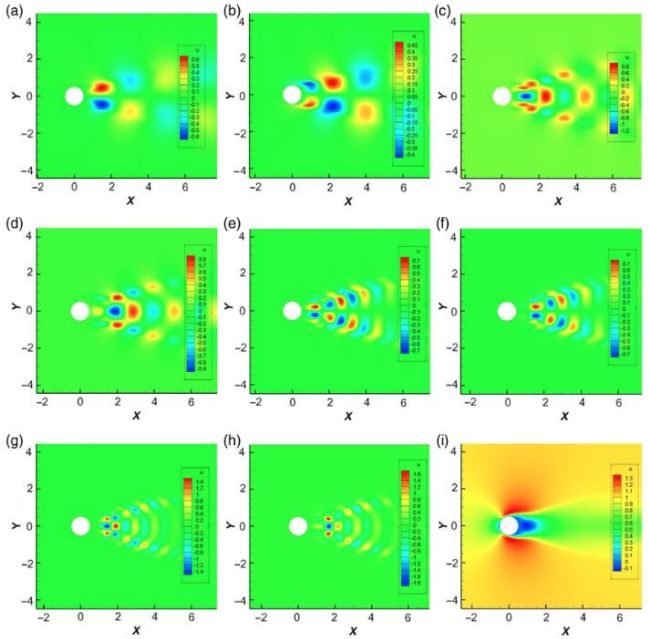
POD modes of the oscillating cylinder: (a) POD mode #1, (b) POD mode #2, (c) POD mode #3, (d) POD mode #4, (e) POD mode #5, (f) POD mode #6, (g) POD mode #7, (h) POD mode #8 and (i) average field.

적합직교분해 (POD)

– 절차

- (3) expansion coefficient 계산
  – 각 snapshot에 포함된 basis의 비중

03_POD	• Calculate the basis function(ϕ_m) • Normalize basis function($\bar{\phi}_m$) • Calculate POD's expansion coefficient(a_m)	$\phi_m = X^T \bullet \beta_m$ $\bar{\phi}_m = \phi_m / \sqrt{\phi^2}$ $a_m = X^T \bullet \bar{\phi}_m$



POD modes of the oscillating cylinder: (a) POD mode #1, (b) POD mode #2, (c) POD mode #3, (d) POD mode #4, (e) POD mode #5, (f) POD mode #6, (g) POD mode #7, (h) POD mode #8 and (i) average field.

```
for (int i=0; i<nDim; i++) {
  std::string uGradSigName;
  uGradSigName = "uGradSigName_" + std::to_string(i);
  volVectorField uGradSig(generateCustomField(runTime,mesh,uGradSigName),
                          UMean&gradSigs[i]);

  uGradSigs.push_back(uGradSig);

  std::string sigGradUName;
  sigGradUName = "sigGradUName_" + std::to_string(i);
  volVectorField sigGradU(generateCustomField(runTime,mesh,sigGradUName),
                          sigs[i]&gradU);
  sigGradUs.push_back(sigGradU);
}

for (int i=0; i<nDim; i++) {
  for (int j=0; j<nDim; j++) {
    std::string sigGradSigName;
    sigGradSigName = "sigGradSigName_" + std::to_string(i+nDim*j);
    volVectorField sigGradSig(generateCustomField(runTime,mesh,sigGradSigName),
                              sigs[i]&gradSigs[j]);
    sigGradSigs[i+nDim*j] = sigGradSig;
  }
}

std::vector<double> constant(nDim,0.0);
std::vector<std::vector<double>> linear(nDim, std::vector<double>(nDim,0.0));
std::vector<std::vector<std::vector<double>>> quadratic(nDim, std::vector<std::vector<double>>(nDim, std::vec

// this loop calculates the Galerkin System matrices Q L C for the ROM equation.
// constant term, linear term, and quadratic term
for (int k=0; k<nDim; k++) {
  constant[k] = -1*innerProductPOD(sigs[k],UgradU,cellVolume)+ (nu+nu_tilda)*innerProductPOD(sigs[k],laplUMea
  for (int m=0; m<nDim; m++) {
    linear[k][m] = -1*innerProductPOD(sigs[k],uGradSigs[m],cellVolume)
    - innerProductPOD(sigs[k],sigGradUs[m],cellVolume) + (nu+nu_tilda)*innerProductPOD(sigs[k],laplSigs[m],ce
    for (int n=0; n<nDim; n++) {
      quadratic[k][m][n] = -1*innerProductPOD(sigs[k],sigGradSigs[m+nDim*n],cellVolume);
    }
  }
}
```

적합직교분해 (POD)

– 절차

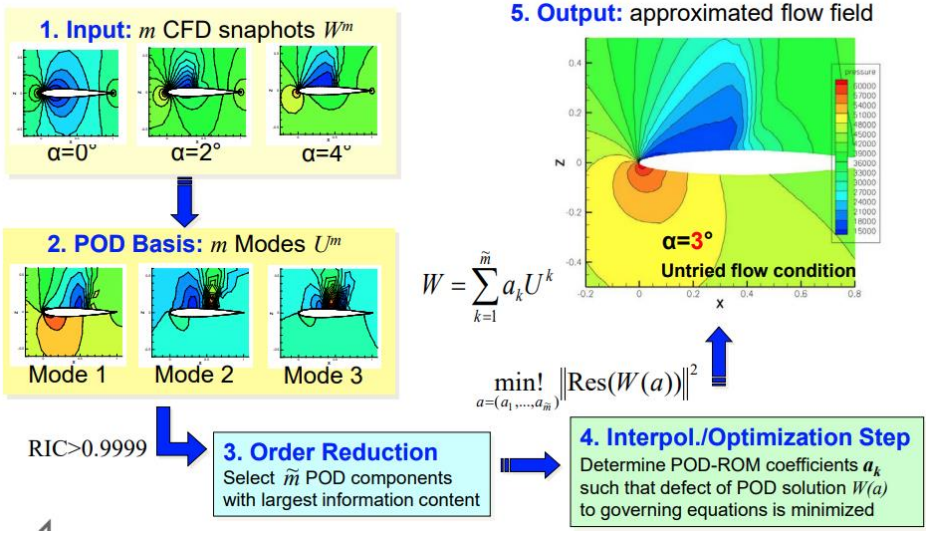
- (4) expansion coefficient 보간
  - 입력 매개변수 (AOA, etc.)
  - POD‘s expansion coefficient
  - Snapshot으로부터 보간 / 대체모델 생성

$$\vec{y}_j \approx \sum_{i=1}^{d} a_{ij} \vec{w}_i$$

04_Interpolation Coefficient	• POD's expansion coefficient(a_m)을 Design space의 변수에 따라 보간 또는 회귀모델 생성	• Linear interpolation using SciPy module (1D) • 또는 RBF 모델 사용(2D, Isight 사용)

podBasisCalc.C | podPrecompute.C | podROM.C | podFlowReconstruct.C | podPostProcess.C

```
202  int writeSteps = 0;
203
204  if(writeFreq == 0){
205    writeSteps = nSteps/numDirs;}
206  else{
207    writeSteps = writeFreq;}
208
209  std::ofstream afiles;
210  afiles.open("avals.csv");
211
212  for (int t=0; t<nSteps+1; t++){
213    cout << "t = " << timeElapsed << endl; // Case progress info in terminal
214    for (int i=0; i<nDim; i++) {
215      double da = constant[i];
216      for (int j=0; j<nDim; j++) {
217        da += linear[i+j*nDim]*prevAvals[j];
218        for (int k=0; k<nDim; k++) {
219          da += quadratic[i+j*nDim+k*nDim*nDim]*prevAvals[k]*prevAvals[j];
220        }
221      }
222      avals[i] = prevAvals[i] + da*dt;
223    }
224
225    for (int i=0; i<nDim; i++){
226      prevAvals[i] = avals[i]; // updating previous avals
227      avalues[t][i] = avals[i]; // storing a values for in 2D vector for lat
228    }
229
230    if(t%writeSteps == 0){
231      double tcol = startTime + dt*t;
232      afiles << tcol << ",";
233      for (int j=0; j<nDim; j++){
234        afiles << std::fixed << std::setprecision(16) << avalues[t][j] << ",
235      }
236      afiles << endl << std::flush;
237    }
238
239    timeElapsed += dt;
240  }
241  afiles.close();
```



적합직교분해 (POD)

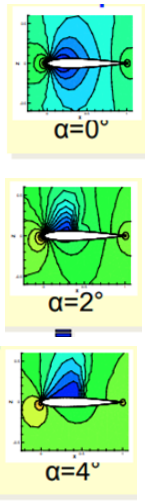
– 절차

- (4) expansion coefficient 보간
  - 입력 매개변수 (AOA, etc.)
  - POD‘s expansion coefficient
  - Snapshot으로부터 보간 / 대체모델 생성

$$\vec{y_j} \approx \sum_{i=1}^{d} a_{ij} \vec{w_i}$$

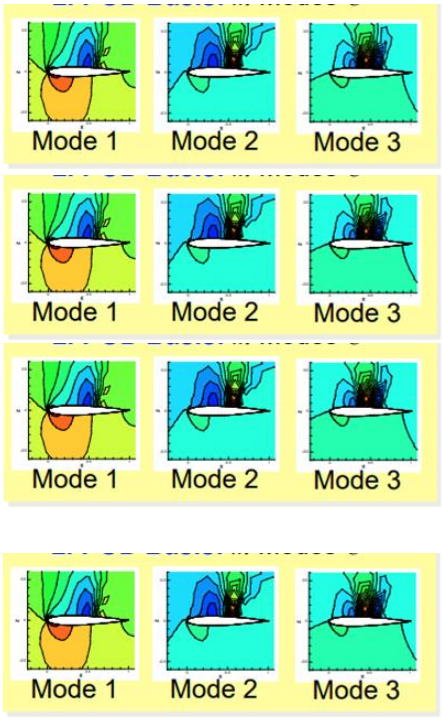
04_Interpolation Coefficient	• POD's expansion coefficient(a_m)을 Design space의 변수에 따라 보간 또는 회귀모델 생성	• Linear interpolation using SciPy module (1D) • 또는 RBF 모델 사용(2D, Isight 사용)

Snapshot (Known)



$$U(0) = a1(0)\,\sigma1 + a2(0)\,\sigma2 + a3(0)\,\sigma3$$
$$U(2) = a1(2)\,\sigma1 + a2(2)\,\sigma2 + a3(2)\,\sigma3$$
$$U(4) = a1(4)\,\sigma1 + a2(4)\,\sigma2 + a3(4)\,\sigma3$$
$$U(3) = a1(3)\,\sigma1 + a2(3)\,\sigma2 + a3(3)\,\sigma3$$

const.



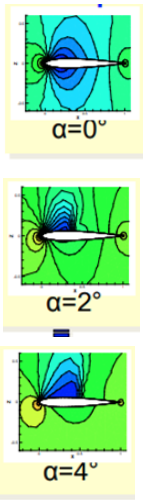
적합직교분해 (POD)

– 절차
- (5) 유동장 재생성
  – Basis × Expansion coefficient

$$\vec{y_j} \approx \sum_{i=1}^{d} a_{ij} \vec{w_i}$$

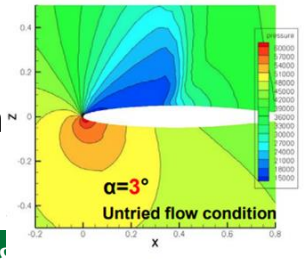
04_Interpolation Coefficient	• POD's expansion coefficient(a_m)을 Design space의 변수에 따라 보간 또는 회귀모델 생성	• Linear interpolation using SciPy module (1D) • 또는 RBF 모델 사용(2D, Isight 사용)

Snapshot (Known)

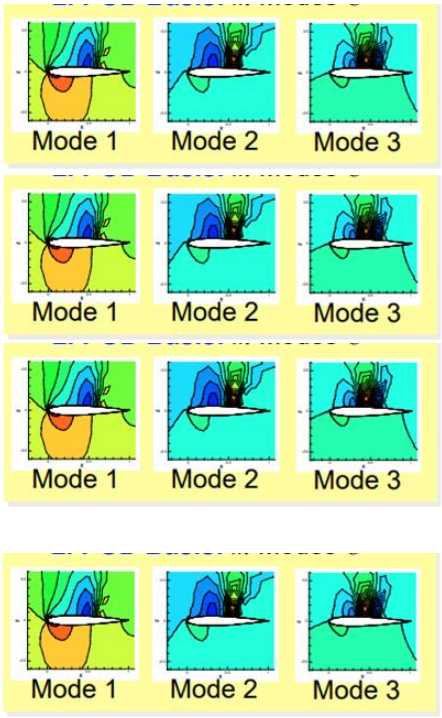


$$U(0) = a1(0)\,\sigma1 + a2(0)\,\sigma2 + a3(0)\,\sigma3 \quad \text{const.}$$
$$U(2) = a1(2)\,\sigma1 + a2(2)\,\sigma2 + a3(2)\,\sigma3$$
$$U(4) = a1(4)\,\sigma1 + a2(4)\,\sigma2 + a3(4)\,\sigma3$$

Prediction



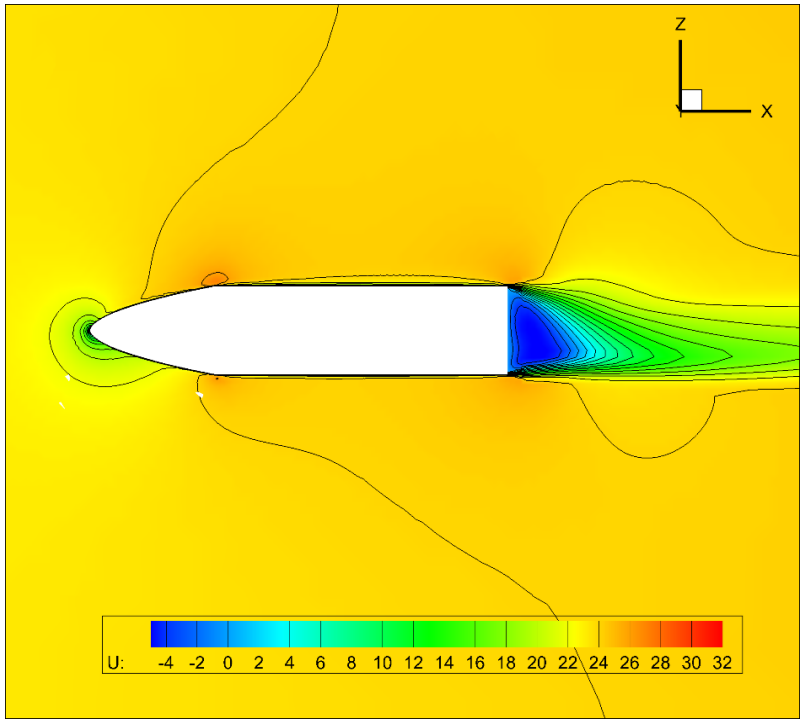
$$U(3) = a1(3)\,\sigma1 + a2(3)\,\sigma2 + a3(3)\,\sigma3$$

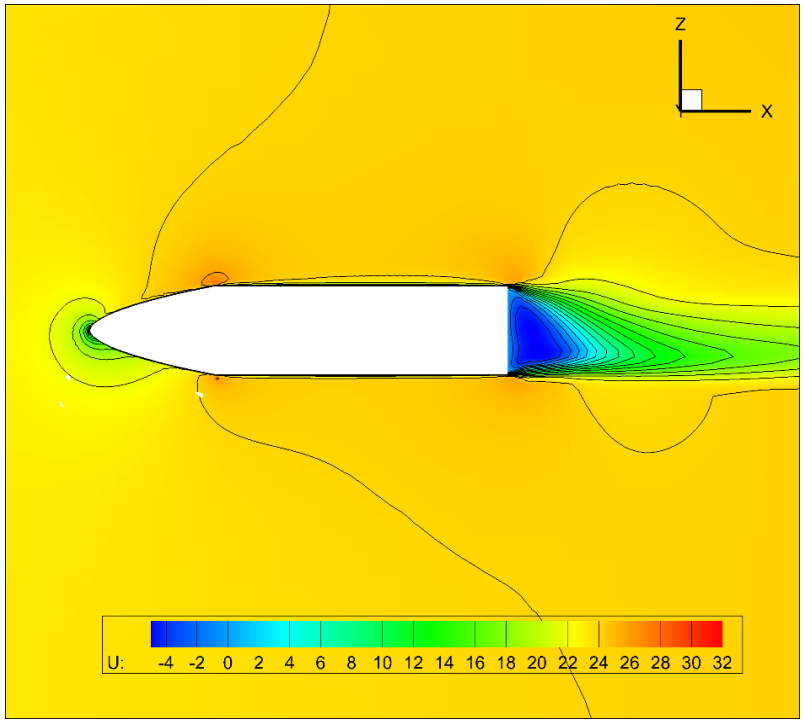


적합직교분해 (POD)

– 성능 및 정확도 예시

- 6개의 입력 매개변수를 갖는 3차원 공력 해석 케이스 (격자 개수 약 48만)
- 135개 스냅샷 데이터를 CFD 해석으로 획득
- POD ROM 구성 (1분 소요) + 유동장 재구성 (1초 미만 소요)
- 신규 해석조건에 대한 FOM 해석 (좌), POD ROM 예측 (우), 오차 1.09%
- CFD 해석 (30분 소요) 대비 소요 시간 대폭 저감
- 실시간 시뮬레이터 구현 가능성 확인





적합직교분해 (POD)

- 기법 적용 시 주요 고려 사항
  - 차수축소모델 (ROM) 구성에 사용할 snapshot 개수의 적절한 선정 필요
    - CFD를 많이 풀수록 ROM이 정확해지지만 계산 시간 증가
    - 고유값 기반으로 ROM 정확도를 추정하는 방법 활용
    - 통상적으로 Full-Order Model 대비 1/4 수준
  - 해석 조건의 적절한 범위 설정 필요
    - 목표하는 조건을 어느 정도 포함하도록 설정
      - » 예시 : airfoil 해석 시 stall 고려 등
  - 선형결합을 통한 재구성이기 때문에 충격파 등 불연속을 포착하는 것에는 취약

- 격자 요구조건
  - 수식 상 모든 결과데이터를 단일 직사각행렬로 취합 필요
  - 기본적으로는 Topology가 동일한 격자여야 함 (개수, index 등 모두 일치)
    - 일부 최신 기법 예외

- 형상최적설계 적용 시
  - 변동하는 모든 형상에 대해 topology가 동일한 격자를 생성하는 것이 일차적 목표
  - 최초 격자 1개 생성 후 deform utility 사용

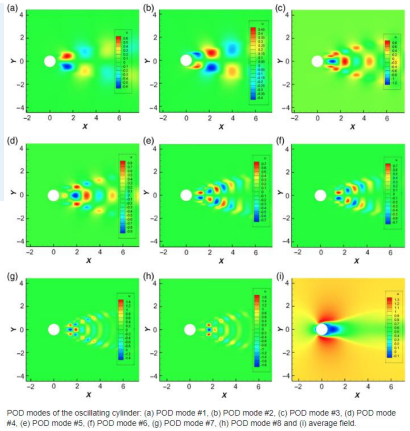
적합직교분해 (POD)

- Snapshot의 적절한 개수 설정 근거
  - 강형민, “적합직교분해 기법에서의 효율적인 스냅샷 선정을 위한 고유값 분석”, 2017

02_SVD	• 공분산행렬(Covariance Martix) 생성 • Eigenvalue & Eigenvectors 추출 ✓ Left Eigenvector = POD mode (Basis function)	$R = X^T X$ $X = \begin{bmatrix} x_1(t_1) & \cdots & x_n(t_1) \\ \vdots & \ddots & \vdots \\ x_1(t_m) & \cdots & x_n(t_m) \end{bmatrix} \in R^{m \times n}$ $R\beta_m = \lambda_m \beta_m$

$$Y = U\Sigma V^T$$

$$R = Y^T Y \quad R\vec{v_i} = \sigma_i^2 \vec{v_i}, \vec{u_i} = \frac{Y\vec{v_i}}{\sigma_i}$$



POD modes of the oscillating cylinder: (a) POD mode #1, (b) POD mode #2, (c) POD mode #3, (d) POD mode #4, (e) POD mode #5, (f) POD mode #6, (g) POD mode #7, (h) POD mode #8 and (i) average field.

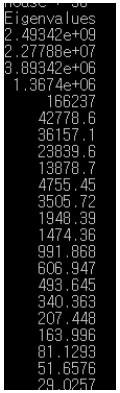


Table 2 Eigen values and $O(R)$ in steady cases

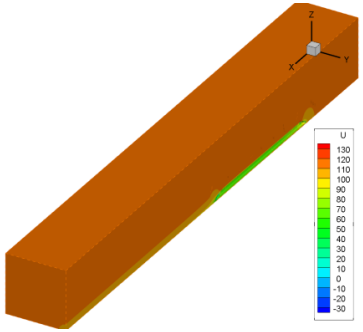
N_s	Eigen Value Pressure Max	Eigen Value Pressure Min	Eigen Value Pressure O(R)	L_2 Error	C_{l_Ori}	C_{l_POD}	Differ. (%)
3	1.23E+04	3.61E-03	3.41E+06	1.69E-06		0.7252	1.97
5	1.23E+04	4.94E-06	2.49E+09	7.84E-07		0.7343	0.74
9	1.23E+04	8.98E-08	1.37E+11	3.13E-07	0.7398	0.7357	0.55
17	1.23E+04	2.55E-09	4.82E+12	2.25E-07		0.7365	0.45
33	1.23E+04	2.30E-10	5.34E+13	1.60E-07		0.7374	0.32

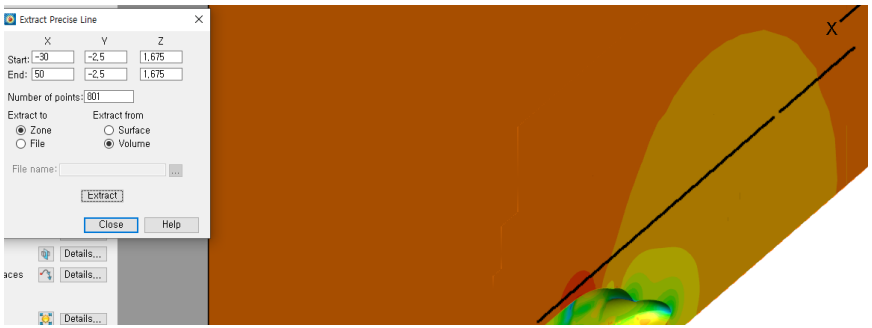
  - 공분산행렬 고유값의 최대/최소 비율
  - Snapshot 개수가 늘어남에 따라 해당 비율이 증가함을 관찰
  - Steady 문제의 경우 10^9, Unsteady 문제의 경우 10^11~12 수준이 되도록 Snapshot의 개수를 조절함이 적당

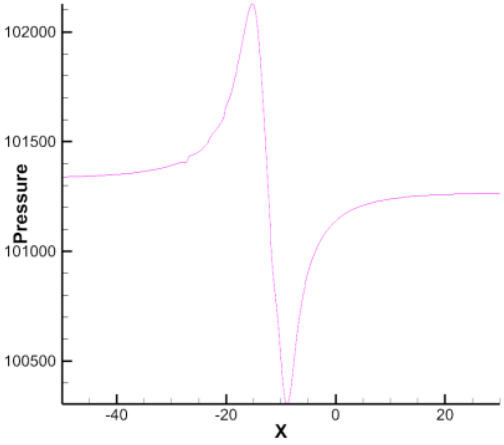
  - 고속열차의 27개 형상매개변수에 대한 500개 샘플 선정이 적절했는지 평가 가능

적합직교분해 (POD)

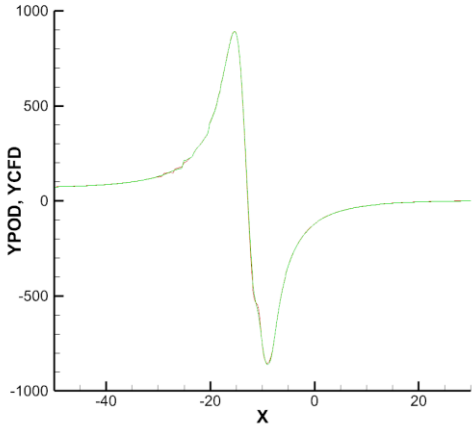
- 고속열차 해석 데이터에 적합직교분해 적용
  - 후처리 항목 중 전두부 주위 압력 변동 데이터 (1D)

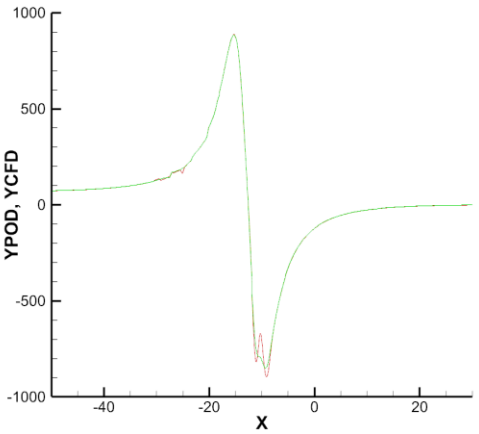


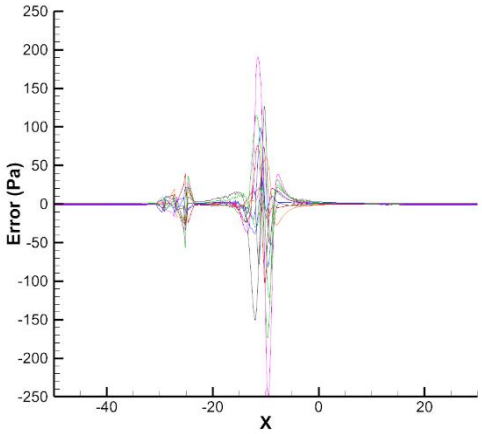




  - 고유값 비율 10^9 를 만족하는 스냅샷 개수 : 36개
    - 형상매개변수가 27개임을 감안할 때 충분하지 않은 개수로 판단됨
    - 평균 오차는 이미 0.5% 수준으로 수렴하였으나, 최대 오차가 10% 수준으로 과도함



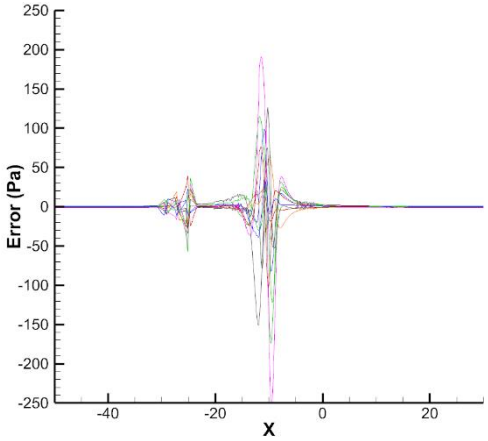


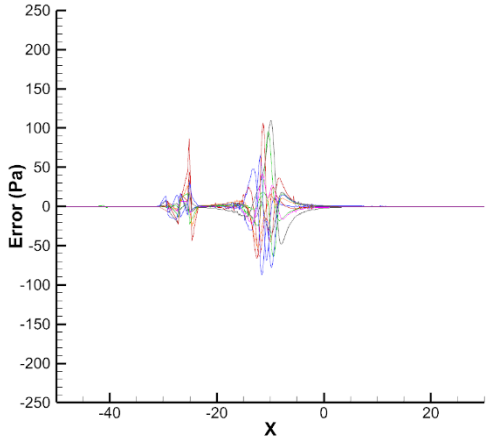


적합직교분해 (POD)

- 고속열차 해석 데이터에 적합직교분해 적용
  - 후처리 항목 중 전두부 주위 압력 변동 데이터 (1D)

  - 고유값 비율 10^9 (스냅샷 36개) -> 고유값 비율 10^13 (스냅샷 228개)





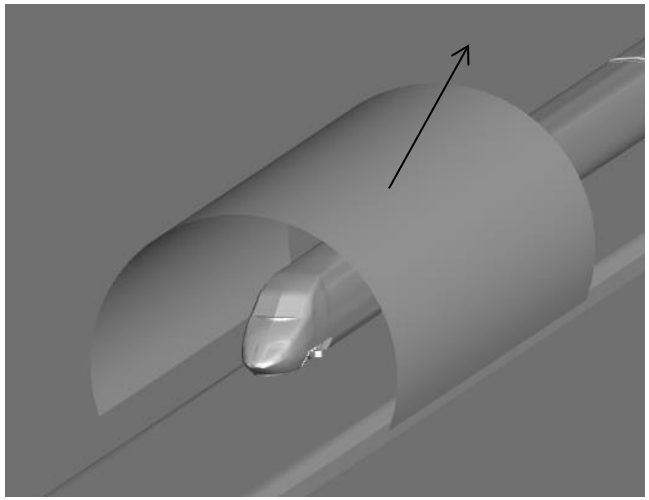
- 오차 개선 확인

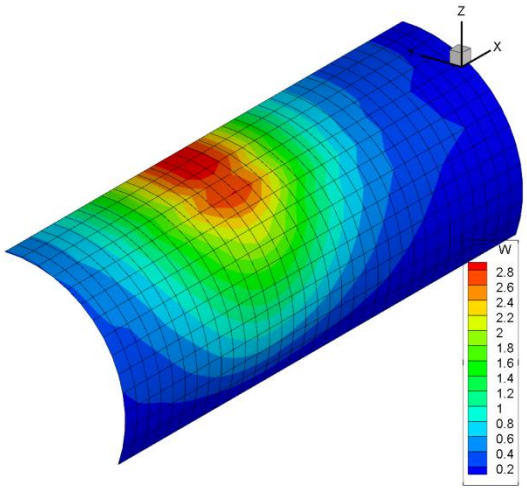
- 고정된 격자 상에서 압력구배의 위치가 변동하는 데이터 특성으로 인한 오차 발생 추정

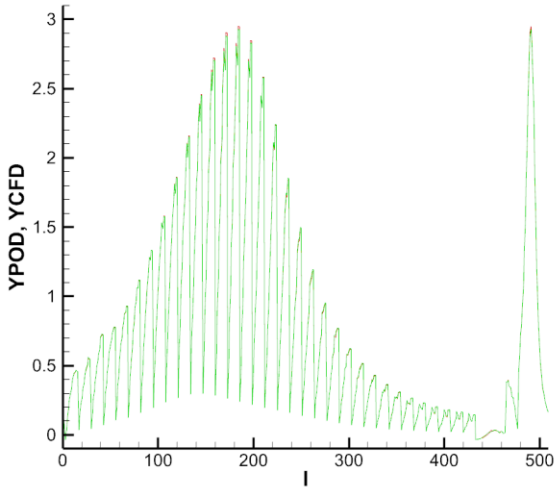
- 수렴 판정 기준 추가 연구 필요
  - 물리량 값의 기준치에 따라 고유값 비율이 변동하는 것으로 추측됨
  - Ex) 절대압력 / 계기압력 등

적합직교분해 (POD)

- 고속열차 해석 데이터에 적합직교분해 적용
  - 후처리 항목 중 V-wall 데이터



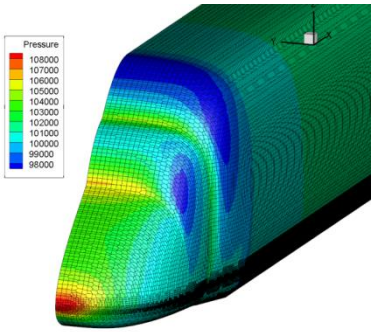




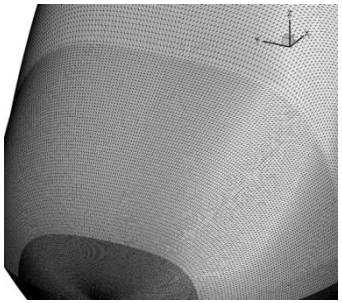
  - 스냅샷 36개 사용 시 고유값 비율 10^9, 평균 오차 0.2%, 최대 오차 1% 수준
  - 스냅샷 248개 사용시 고유값 비율 10^12, 평균 오차 0.15%, 최대 오차 1% 수준

적합직교분해 (POD)

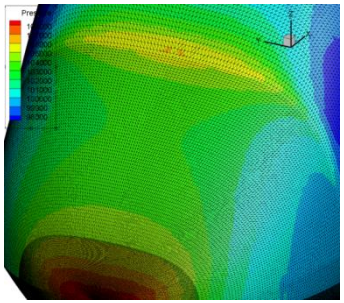
- 고속열차 해석 데이터에 적합직교분해 적용
  - 후처리 항목 중 표면압력분포 데이터 (보완 필요)
    - 기존 데이터 500개를 모두 사용했음에도 고유값 비율 10^9 기준을 충족하지 못함
    - 격자 topology를 통일하기 위해 데이터를 내삽하는 과정에서 큰 오차가 발생한 것으로 추정
    - 정성적인 경향 확인 / 공력계수 등 평균적 수치 계산에는 문제가 없으나, 모든 격자점에 대한 정밀한 결과데이터 추정은 불가능



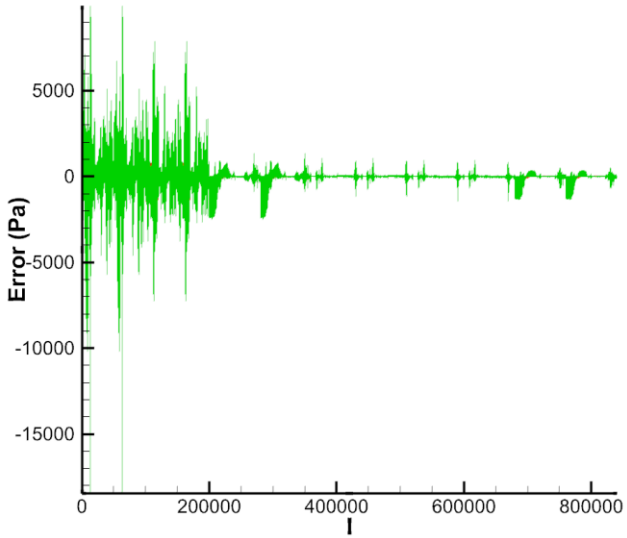
CFD 해석 결과
(임의 topology)



CAD 기반 격자
(통일된 topology)



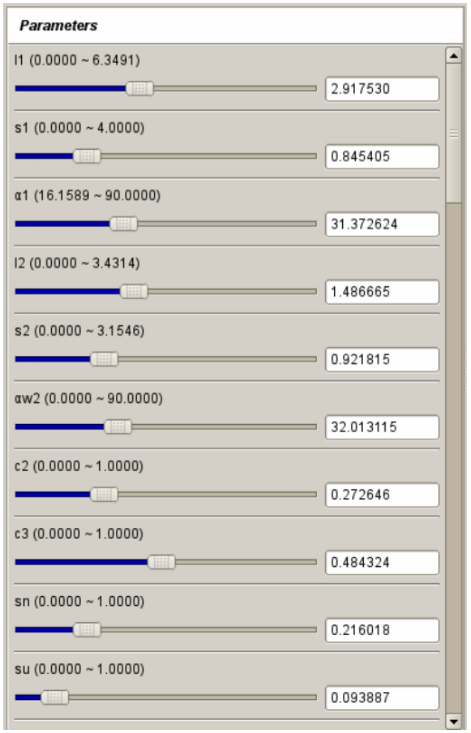
내삽된
표면압력분포

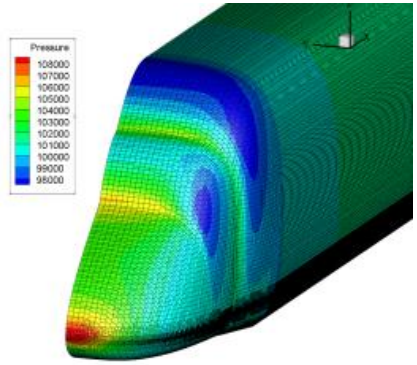


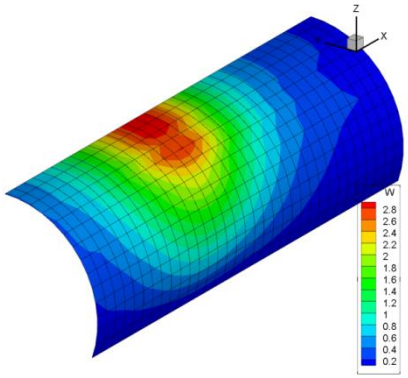
  - 적절한 격자 생성 및 사용이 PODROM 정확도에 매우 중요한 것으로 생각됨
    - deformMesh의 우선적 보완 검토

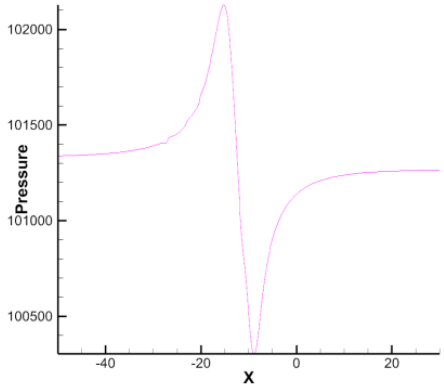
적합직교분해 (POD)

- 고속열차 해석 데이터에 적합직교분해 적용
  - 적합직교분해 기반 차수축소모델 활용 형상최적설계 테스트
    - (1) 고유값 비율 10^10 를 만족하는 차수축소모델 생성 (100개 케이스, 데이터 11GB)
      - 적합직교분해로 모델 생성 (4분)
      - 신규 형상매개변수를 입력하면 결과데이터 추정 (수 초)



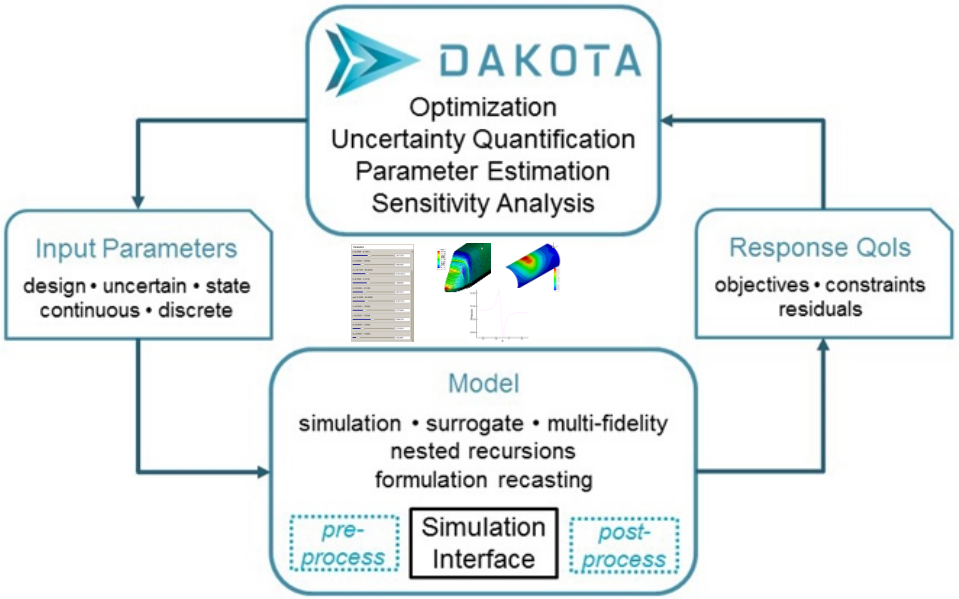


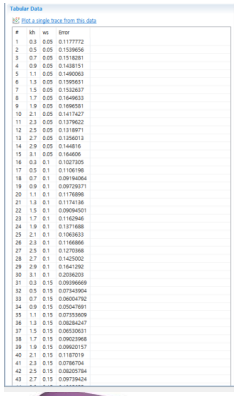


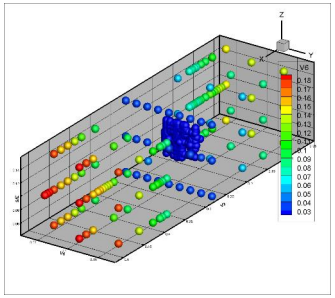


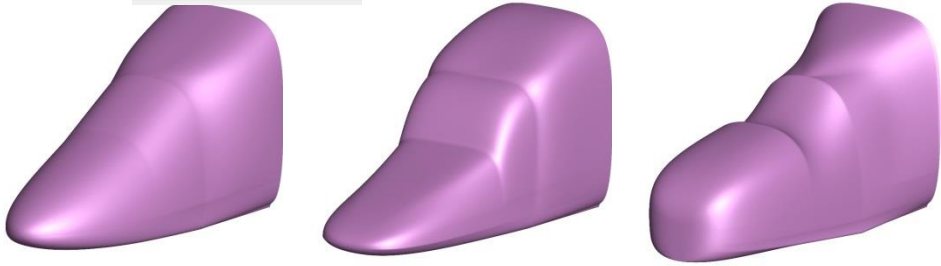
적합직교분해 (POD)

- 고속열차 해석 데이터에 적합직교분해 적용
  - 적합직교분해 기반 차수축소모델 활용 형상최적설계 테스트
    - (2) 형상매개변수 -> 공력성능 계산 프로그램 작성
      - 항력계수, 전도모멘트 : PODROM으로 추정된 표면압력분포를 면적분하여 계산
      - 미기압파 세기 : PODROM으로 추정된 V-wall 데이터를 면적분하여 최댓값을 계산
    - (3) 공력성능 최적화 문제를 DAKOTA로 자동 계산
    - 과제에서 수행한 NSGA-II 기반 형상최적설계와 동일한 결과 확인
      - 매개변수의 적절한 범위 설정, 최적 형상의 직관성에서 기인한 것으로 추측
        - » 100케이스 vs 500케이스



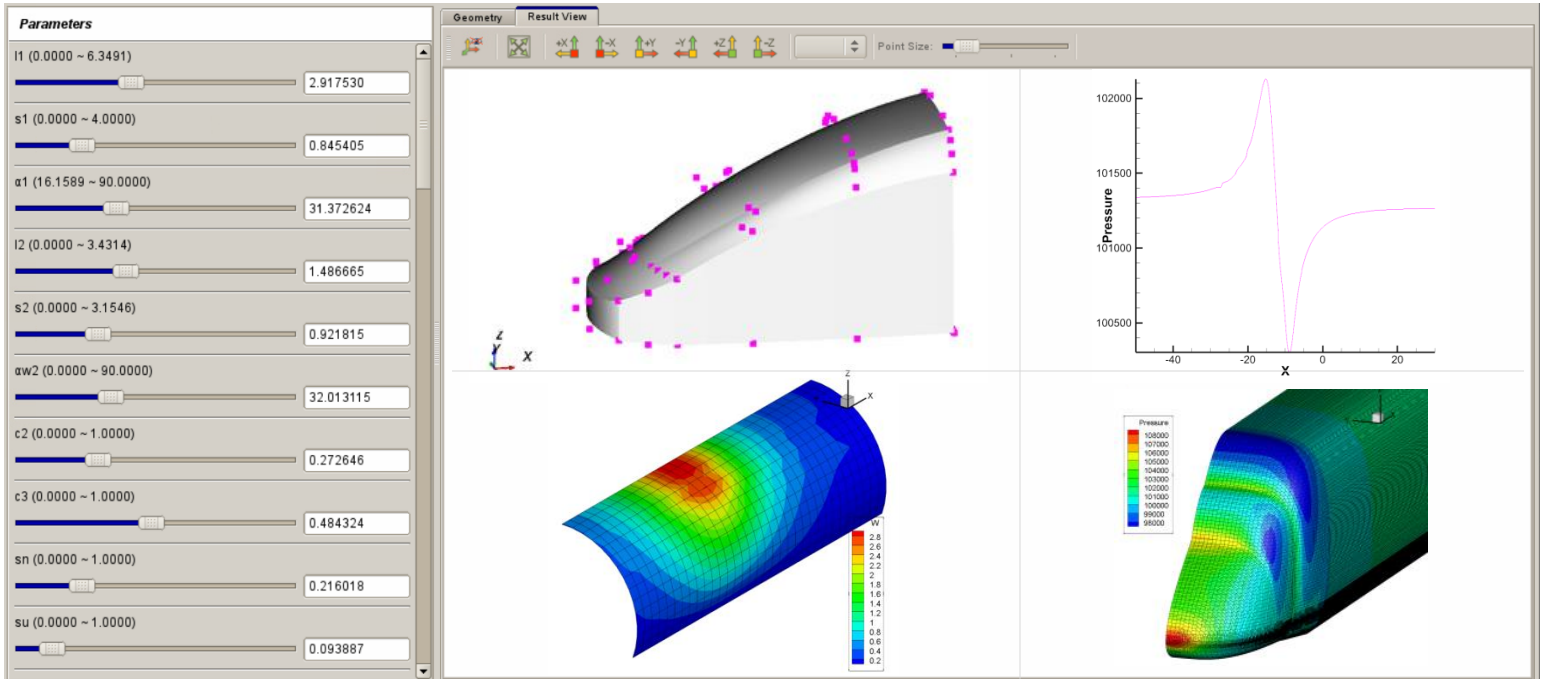






적합직교분해 (POD)

- 고속열차 해석 데이터에 적합직교분해 적용
  - (구상) 실시간 시뮬레이터가 내장된 형상 설계 도구 구현 방안



  - 매개변수 변동에 따른 형상, 공력성능 관련 지표 등을 수 초 이내에 출력
  - 현재까지 검증된 기반기술을 바탕으로 GUI 개발 및 활용이 가능한 상태

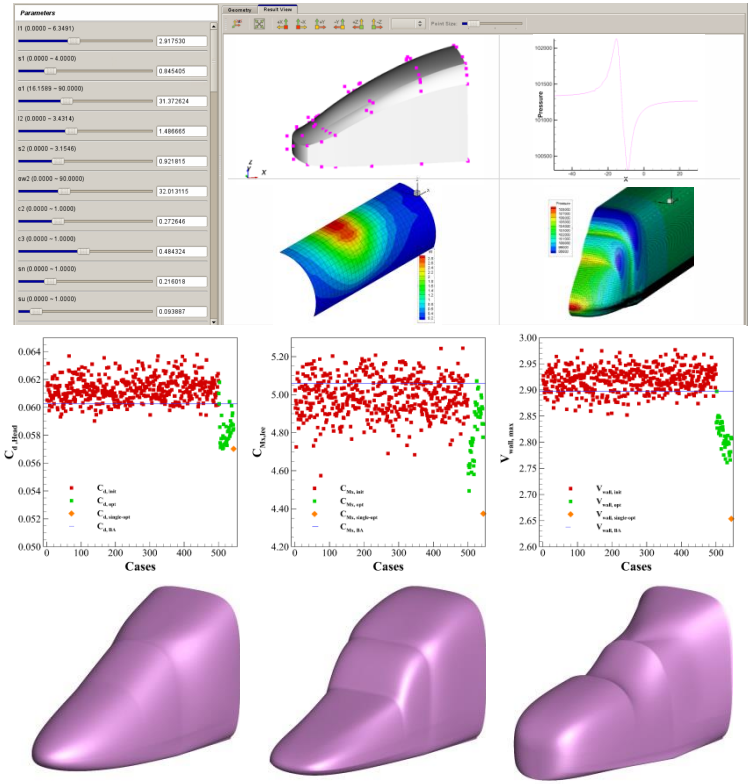
요약

- 고속열차 공력성능 향상을 위한 전두부 형상최적설계
  - CFD 반복해석 및 NSGA-II
  - 적합직교분해 (POD) 기반 차수축소모델 (ROM)

  - PODROM 기법 특성
    - 최적화에 필요한 해석 케이스 개수의 결정이 용이
      - 데이터셋 생성 도중 ROM 정확도 추정 가능
    - Full-Order Model CFD 해석 대비 소요시간 대폭 저감
      - 단순 행렬연산으로 ROM 구성 (분 단위), 신규 결과 추정 (초 단위)
    - Field 전체를 실시간으로 추정 가능
      - 추정 결과에 CFD iteration 수행 시 빠른 수렴으로 정확도 향상

    - 적절한 문제 선정, 입력매개변수 설정, 격자 생성 필요

  - 형상최적설계 문제의 효율적인 해결 및 실시간 시뮬레이터 구현 가능성 확인
  - 향후 신규 문제에 적용하여 검증 및 보완 목표



2024 11th OpenFOAM Korea User's Community Conference

2024.9.26~27

감사합니다