차수축소모델을 이용한 형상최적설계를 위한 형상 및 격자 변형 방법

한국전산유체공학회 2023 춘계학술대회

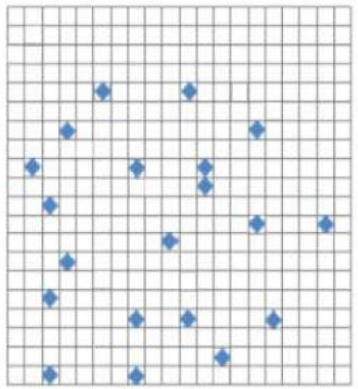
2023.5.11.

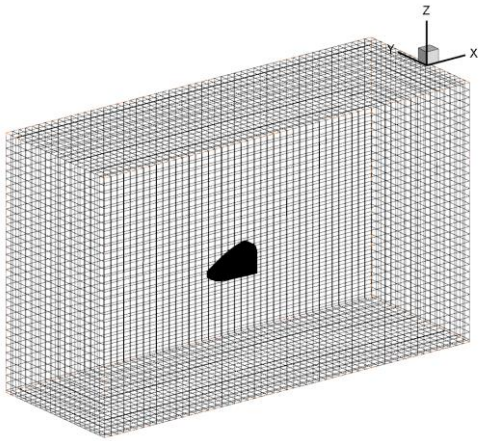
(주) 넥스트폼 이웅현

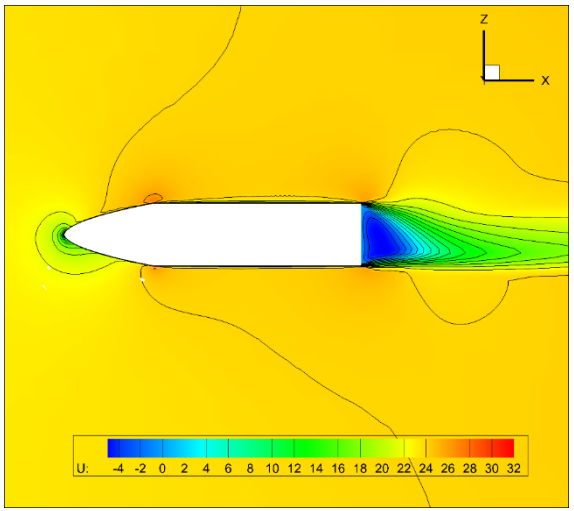
NEXTfoam
open source CFD consulting

개요

- 형상최적설계 시 반복적인 격자 생성 및 유동 해석의 부담





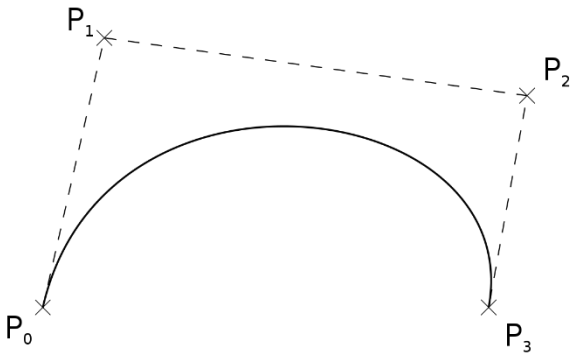


- 격자/해석 단계에서의 계산 부하 및 소요 시간을 저감하기 위한 방법 제시
  - 매개변수를 통한 3차원 형상 정의
  - RBF / IDW 기반의 격자 변형 유틸리티 (deformMesh)
  - 적합직교분해 / 인공신경망 기반의 차수축소모델

매개변수를 통한 3차원 형상 정의

- 베지에 (Bezier) 곡선
  - n개의 control point 및 0≤t≤1로 정의

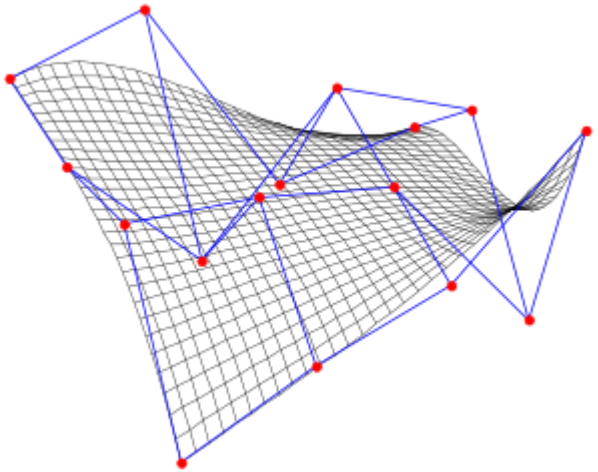
$$\mathbf{B}(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i \mathbf{P}_i$$



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

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

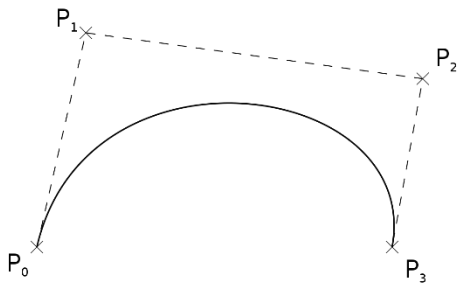
$$B_i^n(u) = \frac{n!}{i!(n-i)!} u^i (1-u)^{n-i}$$

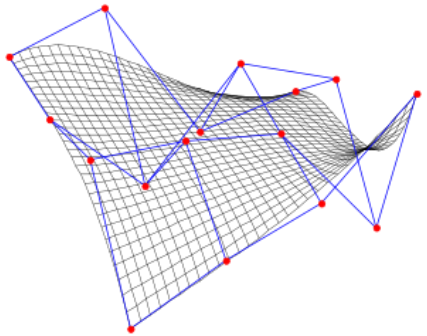


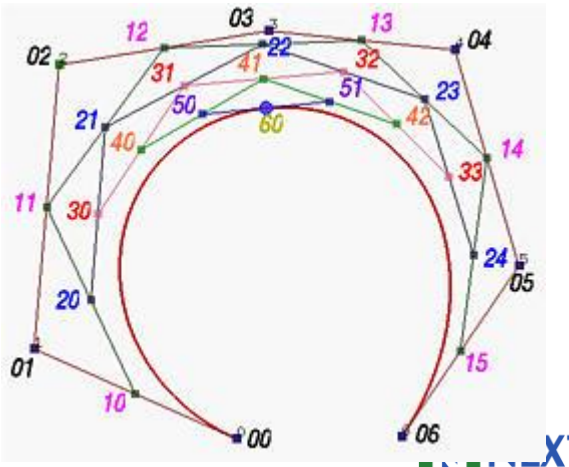
- 일반화 시 B-spline, NURBS 등 사용 가능

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

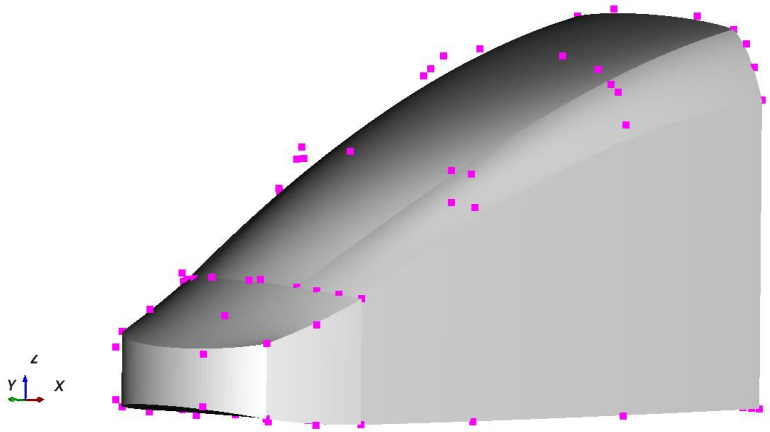


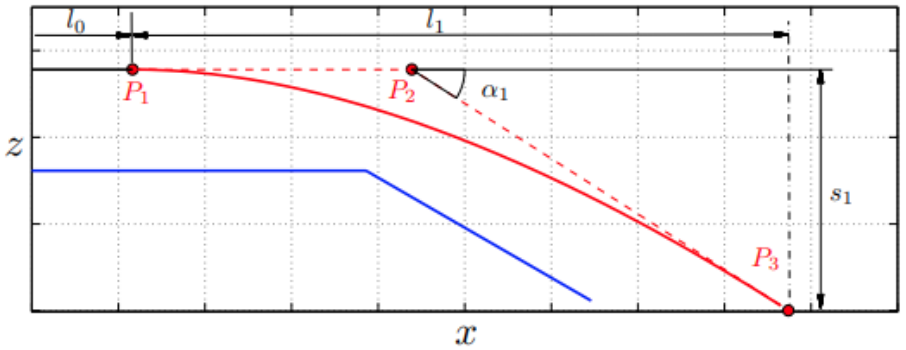




매개변수를 통한 3차원 형상 정의

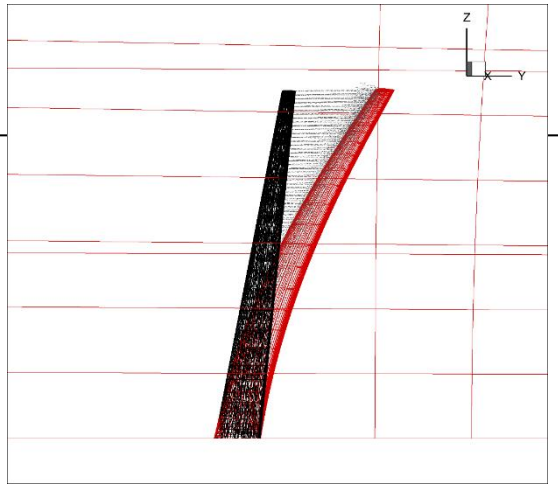
- 베지에 곡선/곡면 사용 예시
  - 철도차량 전두부 형상 매개변수화
  - Control point 80개
    - 대칭 고려 시 40개, 좌표 매개변수 120개
    - 기하학적 구속조건을 통한 매개변수 개수 저감 시 25~40개 수준



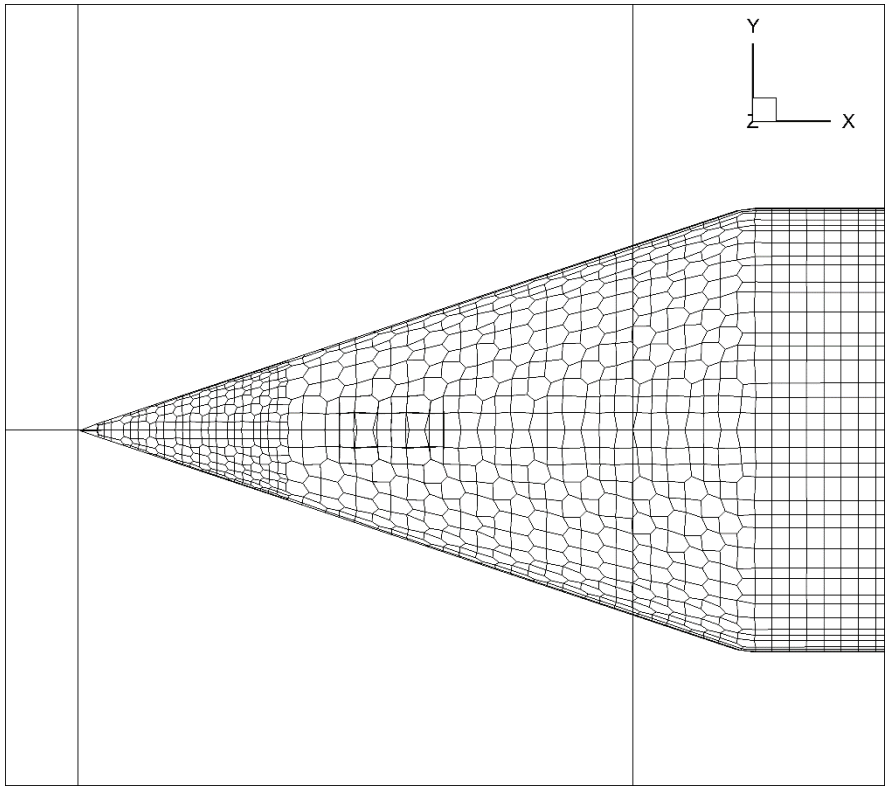


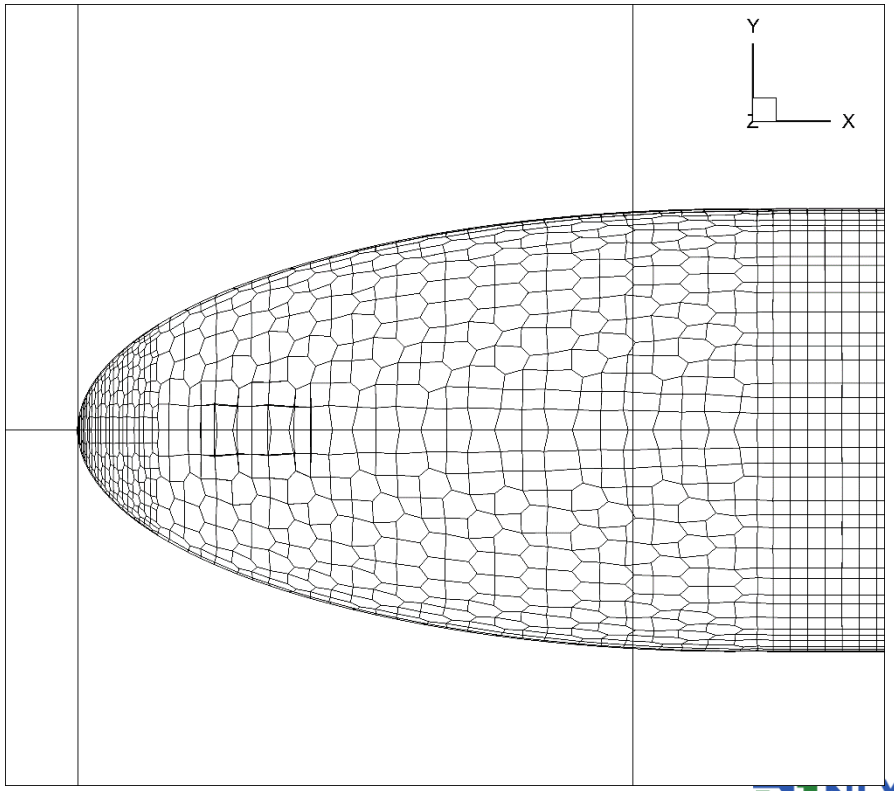
J. Munoz, Aerodynamic Optimization of the Nose Shape of a High-speed Train, 2014

격자 변형



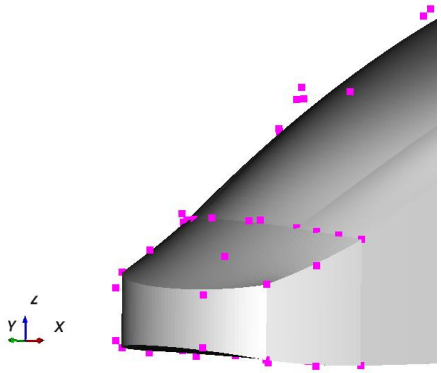
- deformMesh
  - OpenFOAM Utility (자체 개발)
  - control point 좌표 및 변위 벡터를 입력하여 변형
  - RBF, IDW, RBFIDW 방식 지원
  - 격자 품질 유지



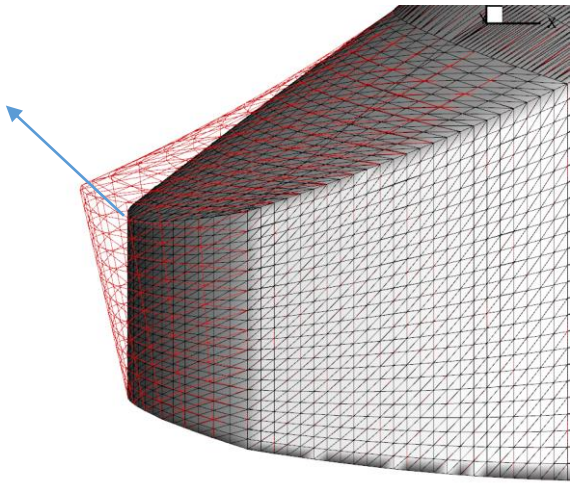


격자 변형

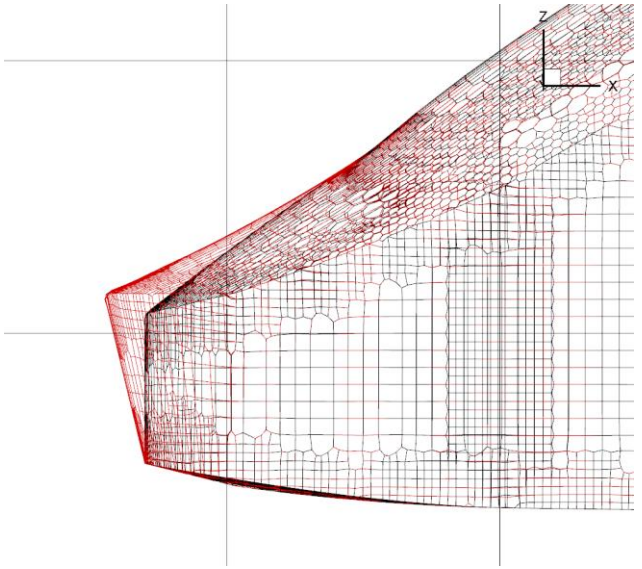
- deformMesh
  - 형상 매개변수 변동에 따른 베지에 곡면 변화 추적
  - 변동 전후의 베지에 곡면 상에서 동일 매개변수값 (u,v)에 해당하는 점들의 변위로부터 격자 변형에 필요한 input 계산
  - 최초 생성한 격자계를 변형하여 반복 해석에 사용 가능
  - 격자의 topology 및 index가 유지되므로 적합직교분해 (POD) 기반 차수 축소모델 구성 가능



매개변수화된 형상



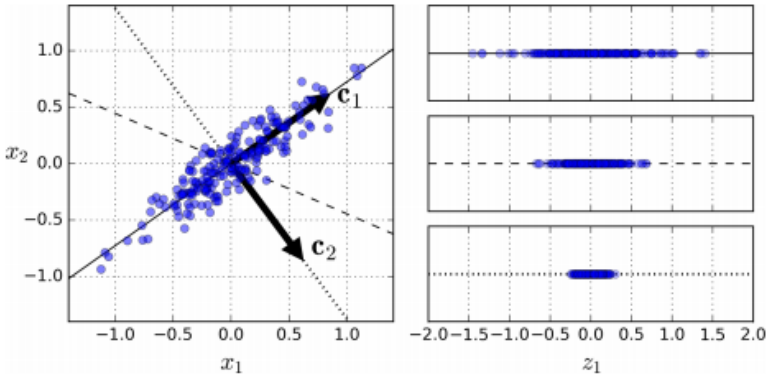
매개변수 변동에 따른
베지에 곡면 형상 변화

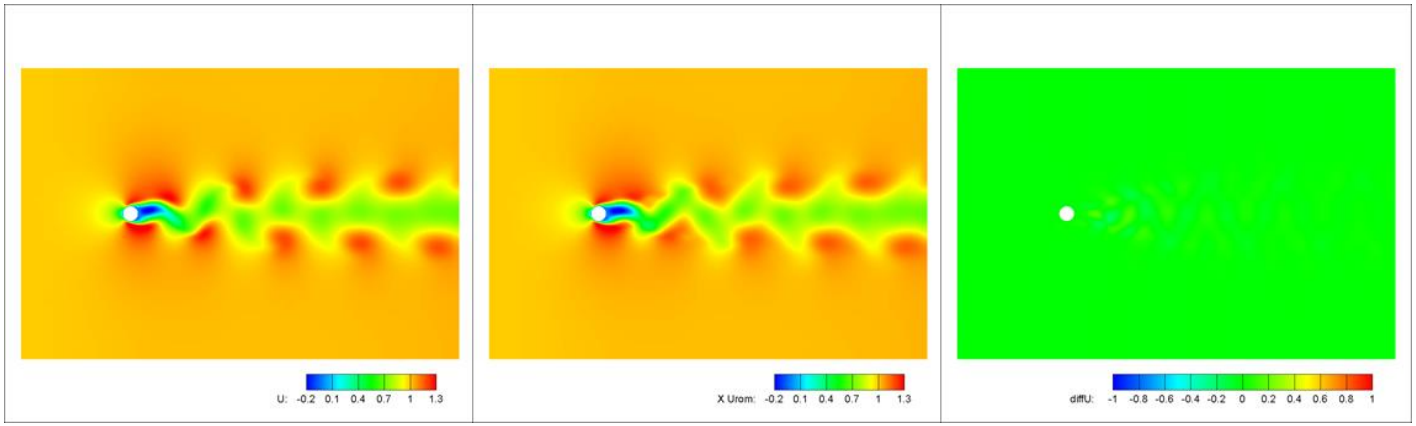


실제 해석 격자 변형 결과

적합직교분해 (POD)

- 주성분 분석 (PCA) 기법 기반
  - 특이치 분해 (SVD)
  - 이미지 압축 등 광범위한 분야에 사용
  - CFD 해석 결과에 적용 가능
  - (해석격자 topology 및 index가 일치해야 함)

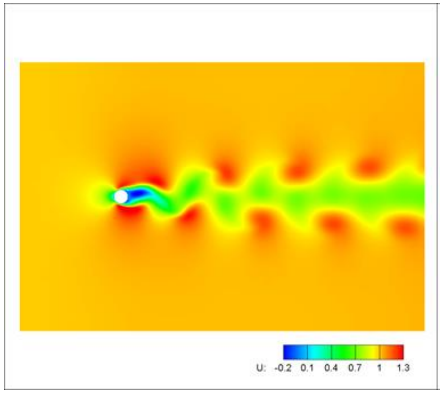




적합직교분해 (POD)

- 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/
```

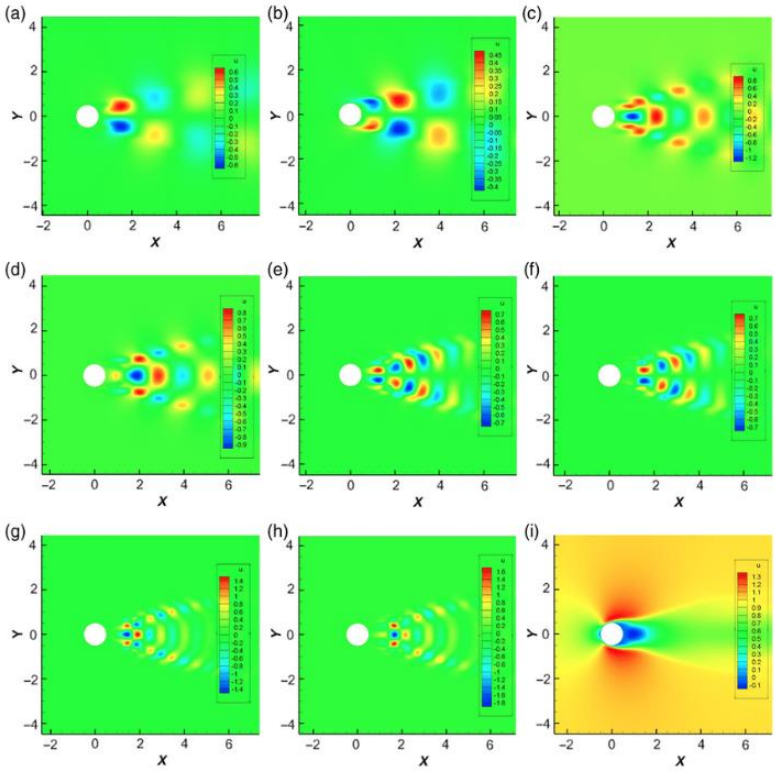


  - 입력 parameter 조건 별 해석 결과
    - AOA, BETA, Mach #, geometric parameter 등 종류 무관
    - 많을수록 정확도 ↑
  - AccelerateCFD 구동을 위해 단일 OpenFOAM 케이스 폴더로 취합
  - N개 격자 * M개 시점 해석 결과가 취합된 N×M 행렬 Y

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

적합직교분해 (POD)

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



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.

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

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

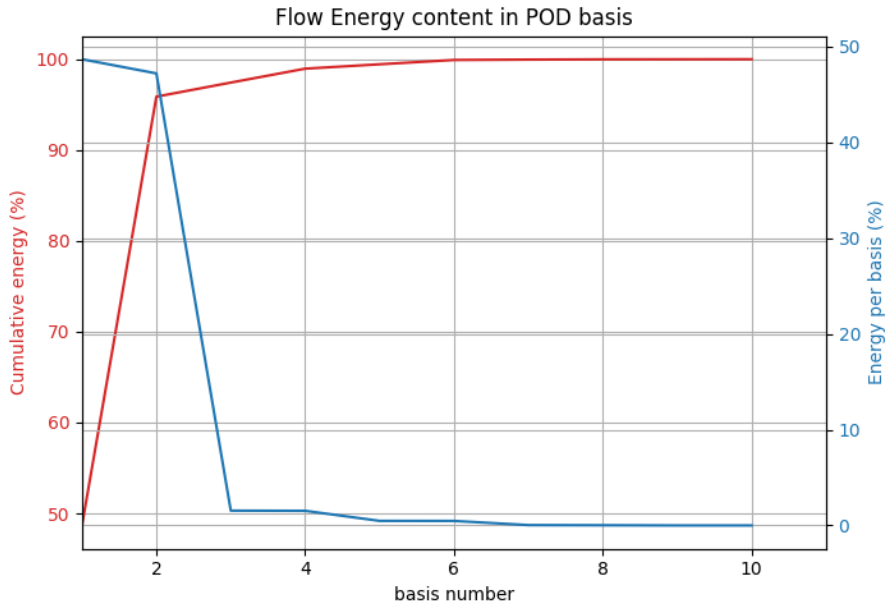
$$R = Y^T Y \quad R\vec{v}_i = \sigma_i^2 \vec{v}_i, \qquad \vec{u}_i = \frac{Y\vec{v}_i}{\sigma_i}$$

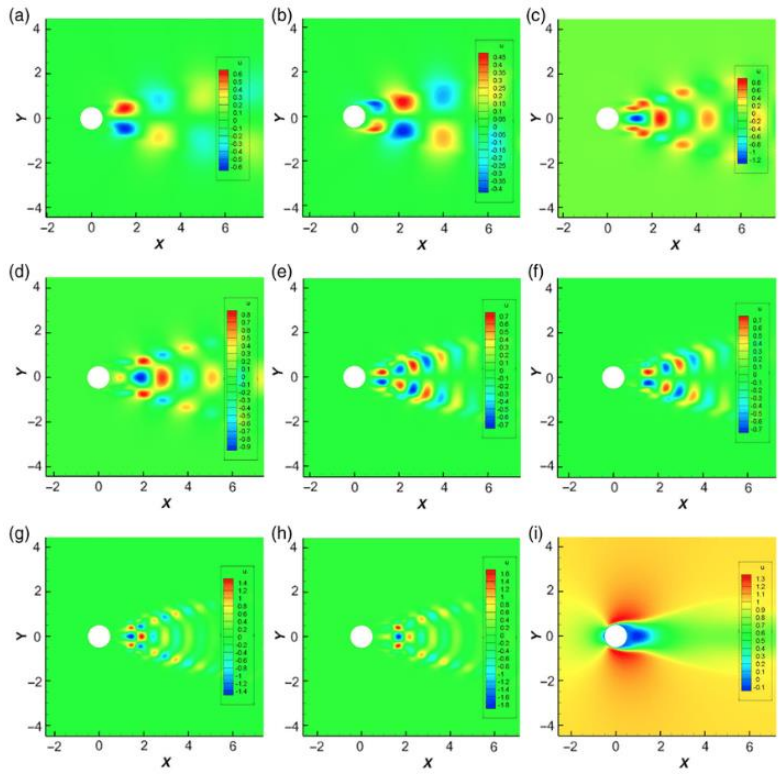
```
forAll(timeDirs, timei)
{
  n = 0;
  forAll(timeDirs, timej)
  {
    Cmn(m, n) = 0.0;
    // applying symmetry
    if (n < m)
    {
      Cmn(m, n) = Cmn(n, m);
      n++;
      continue;
    }

    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)	$R = X^T X$ $X = \begin{bmatrix} x_1(t_1) & \cdots & x_n(t_1) \\ \vdots & \ddots & \vdots \\ x_1(t_m) & \cdots & x_n(t_m) \end{bmatrix} \in R^{m \times n}$ $R\beta_m = \lambda_m \beta_m$



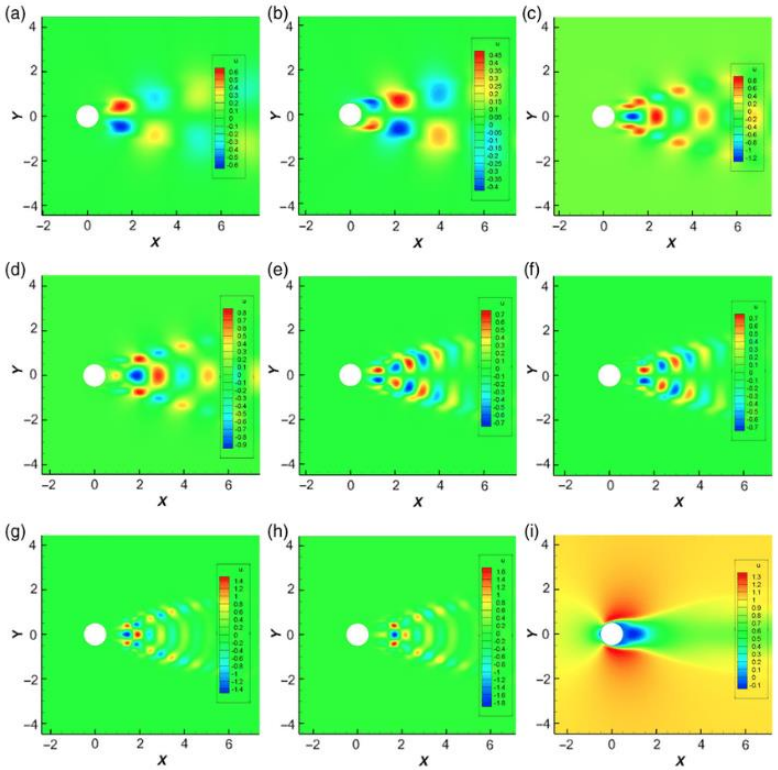


POD modes of the oscillating cylinder: (a) POD mode #1, (b) POD mode #2, (c) POD mode #3, (d) POD mode #4, (e) POD mode #5, (f) POD mode #6, (g) POD mode #7, (h) POD mode #8 and (i) average field.

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.

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

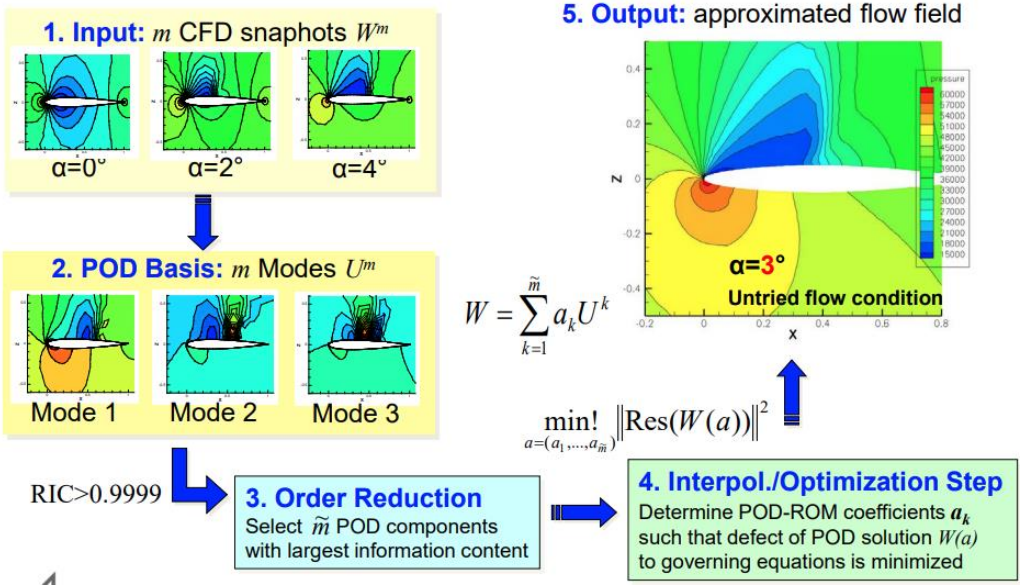
```
271  for (int i=0; i<nDim; i++) {
272    std::string uGradSigName;
273    uGradSigName = "uGradSigName_" + std::to_string(i);
274    volVectorField uGradSig(generateCustomField(runTime,mesh,uGradSigName),
275                            UMean&gradSigs[i]);
276
277    uGradSigs.push_back(uGradSig);
278
279    std::string sigGradUName;
280    sigGradUName = "sigGradUName_" + std::to_string(i);
281    volVectorField sigGradU(generateCustomField(runTime,mesh,sigGradUName),
282                            sigs[i]&gradU);
283    sigGradUs.push_back(sigGradU);
284  }
285
286  for (int i=0; i<nDim; i++) {
287    for (int j=0; j<nDim; j++) {
288      std::string sigGradSigName;
289      sigGradSigName = "sigGradSigName_" + std::to_string(i+nDim*j);
290      volVectorField sigGradSig(generateCustomField(runTime,mesh,sigGradSigName),
291                                sigs[i]&gradSigs[j]);
292      sigGradSigs[i+nDim*j] = sigGradSig;
293    }
294  }
295
296  std::vector<double> constant(nDim,0.0);
297  std::vector<std::vector<double>> linear(nDim, std::vector<double>(nDim,0.0));
298  std::vector<std::vector<std::vector<double>>> quadratic(nDim, std::vector<std::vector<double>>(nDim, std::vec
299
300  // this loop calculates the Galerkin System matrices Q L C for the ROM equation.
301  // constant term, linear term, and quadratic term
302  for (int k=0; k<nDim; k++) {
303    constant[k] = -1*innerProductPOD(sigs[k],UgradU,cellVolume)+ (nu+nu_tilda)*innerProductPOD(sigs[k],laplUMea
304    for (int m=0; m<nDim; m++) {
305      linear[k][m] = -1*innerProductPOD(sigs[k],uGradSigs[m],cellVolume)
306      - innerProductPOD(sigs[k],sigGradUs[m],cellVolume) + (nu+nu_tilda)*innerProductPOD(sigs[k],laplSigs[m],ce
307      for (int n=0; n<nDim; n++) {
308        quadratic[k][m][n] = -1*innerProductPOD(sigs[k],sigGradSigs[m+nDim*n],cellVolume);
309      }
310    }
311  }
```

적합직교분해 (POD)

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

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

04_Interpolation Coefficient	• POD's expansion coefficient(a_m)을 **Design space**의 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();
```

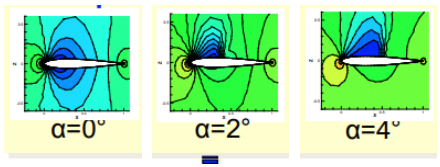
적합직교분해 (POD)

- 절차
  - (4) expansion coefficient 보간
    - 입력 매개변수 (AOA, etc.)
    - POD's expansion coefficient
    - Snapshot으로부터 보간 / 대체모델 생성
    - Kriging, 인공신경망

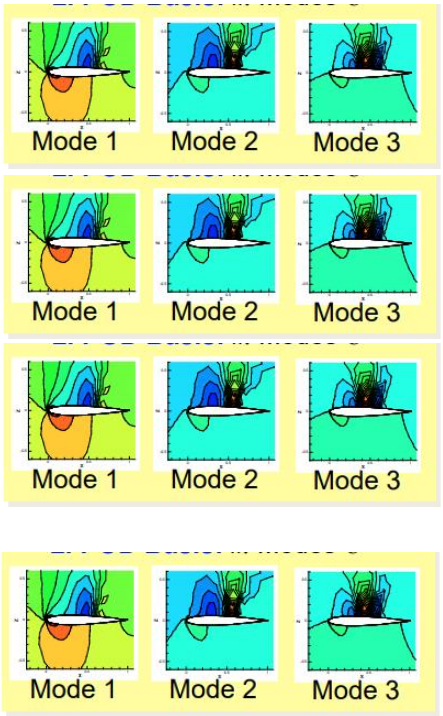
$$\vec{y_j} \approx \sum_{i=1}^{d} a_{ij} \vec{w_i}$$

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)



$$U(0) = a1(0)\,\sigma1 + a2(0)\,\sigma2 + a3(0)\,\sigma3 + \text{const.}$$
$$U(2) = a1(2)\,\sigma1 + a2(2)\,\sigma2 + a3(2)\,\sigma3$$
$$U(4) = a1(4)\,\sigma1 + a2(4)\,\sigma2 + a3(4)\,\sigma3$$
$$U(3) = a1(3)\,\sigma1 + a2(3)\,\sigma2 + a3(3)\,\sigma3$$

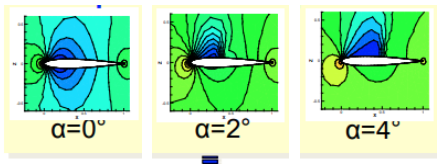


적합직교분해 (POD)

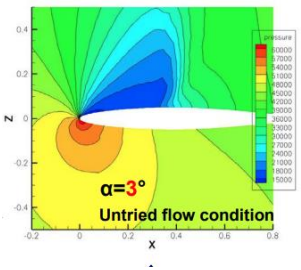
- 절차
  - (5) 유동장 재생성
    - Basis × Expansion coefficient

05_Reconstruction	• 보간/회귀모델로 생성된 POD's expansion coefficient과 basis function을 이용하여 Flow filed를 재생성	$X^{new} = a_m \bullet \bar{\phi}_m$

Snapshot
(Known)

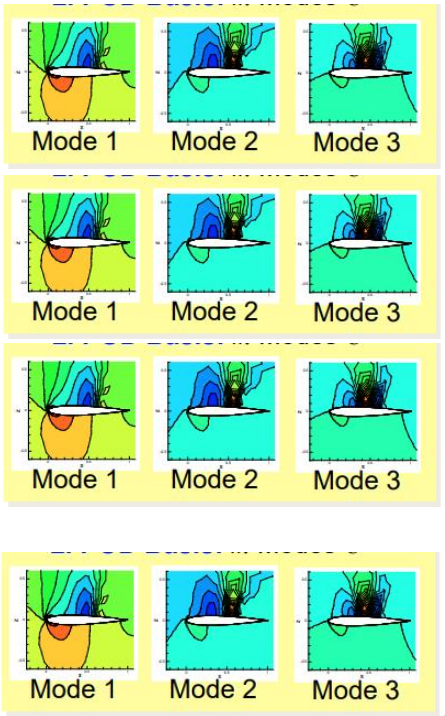


Prediction



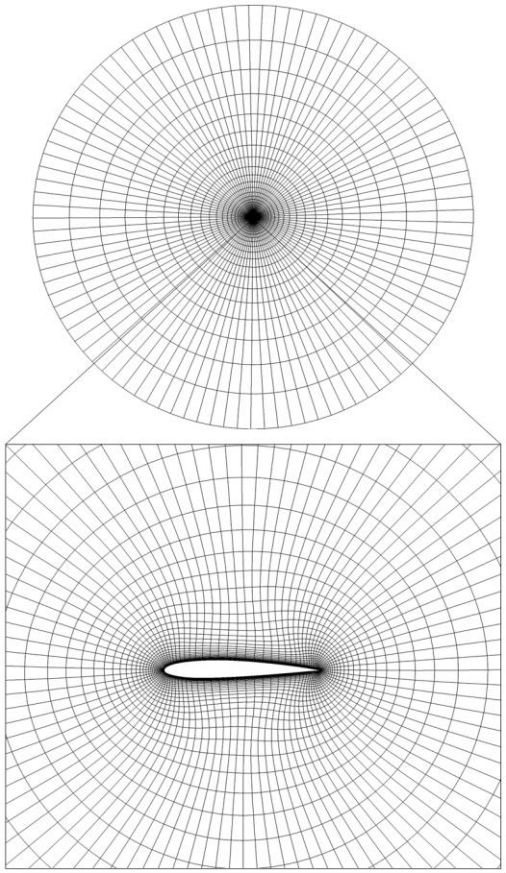
$$U(0) = a1(0)\,\sigma1 + a2(0)\,\sigma2 + a3(0)\,\sigma3$$
$$U(2) = a1(2)\,\sigma1 + a2(2)\,\sigma2 + a3(2)\,\sigma3$$
$$U(4) = a1(4)\,\sigma1 + a2(4)\,\sigma2 + a3(4)\,\sigma3$$
$$U(3) = a1(3)\,\sigma1 + a2(3)\,\sigma2 + a3(3)\,\sigma3$$

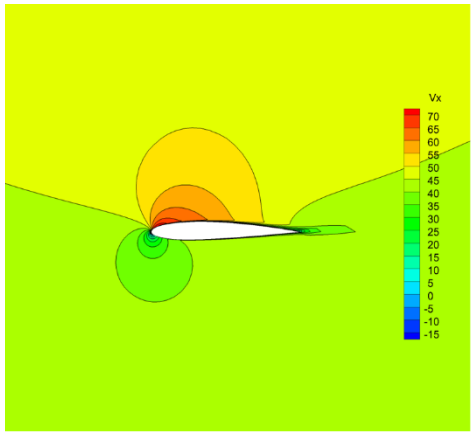
const.

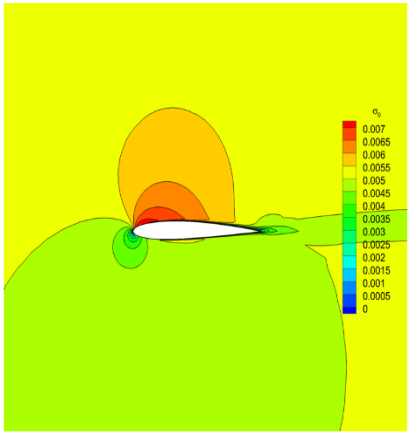


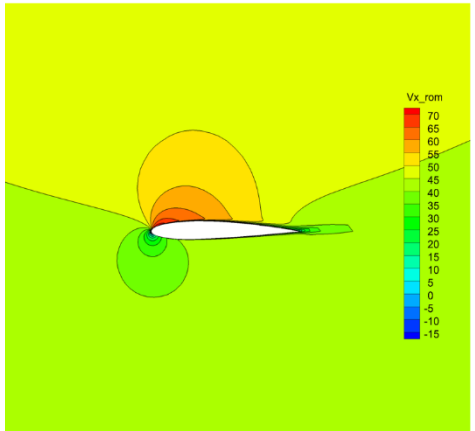
POD+ANN 차수축소모델 검증

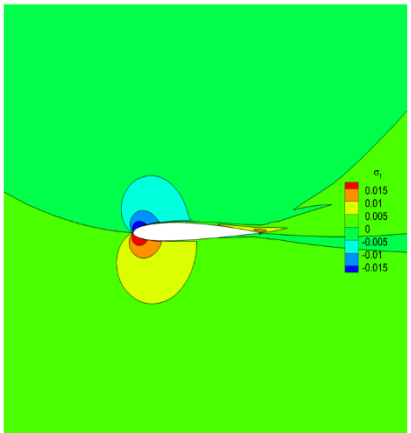
- 정확도 검증
  - NACA2412 익형 아음속 정상 유동 해석
    - 입력 변수 : 자유류 받음각, 속도
    - 5개 스냅샷 데이터로부터 신규 조건에 대한 해석 결과 추정
    - 해석 격자, CFD / POD+ANN 비교, POD 모드 (2개 모드에 에너지 99.999% 분포)











POD+ANN 차수축소모델 검증

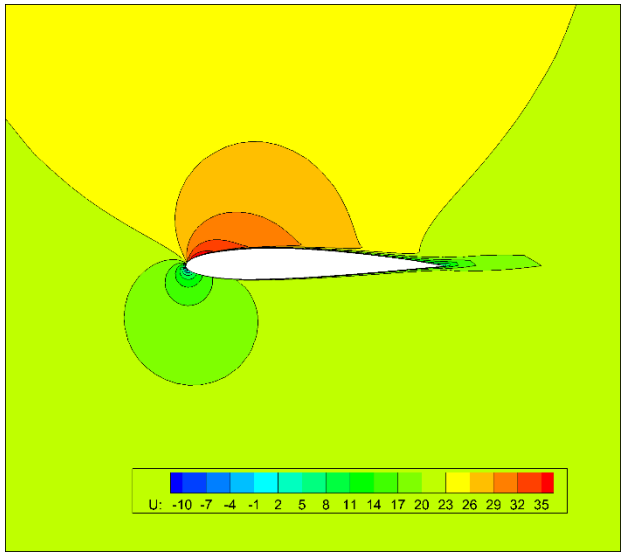
- 정확도 검증
  - NACA2412 익형 아음속 정상 유동 해석
    - POD 수행 과정 포함 시 소요 시간 5% 이내
    - 기 존재하는 POD ROM으로부터 보간만 수행할 경우 전체 유동장 즉시 획득 가능

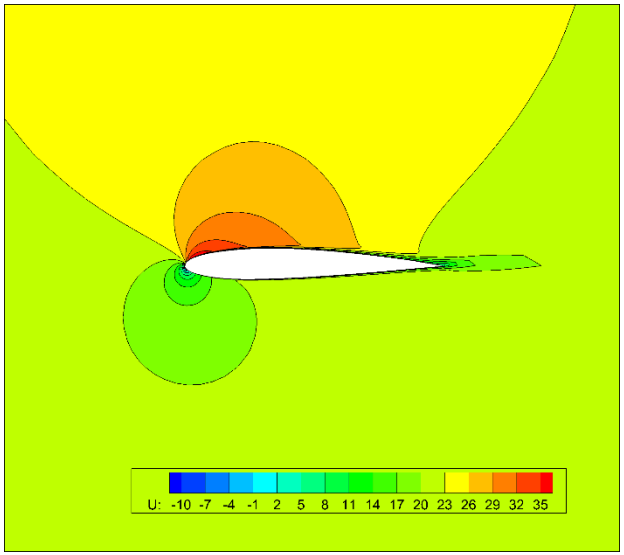
Case No.	Method	V_∞ (m/s)	Angle of attack (deg)	Expansion coefficient		Calculation Time (sec)	Error (%)
1	CFD	30	5	5864.55	-20.0794	5.60	-
2	CFD	40	0	7792.01	654.545		
3	CFD	40	4	7818.70	109.646		
4	CFD	40	10	7787.36	-707.902		
5	CFD	50	5	9774.26	-33.4612		
6	POD + ANN	45	7	8791.20	-302.488	0.27	4.98

  - 적은 개수 (5개) 의 스냅샷으로 인한 큰 오차 (4.98%) => 스냅샷 추가

POD+ANN 차수축소모델 검증

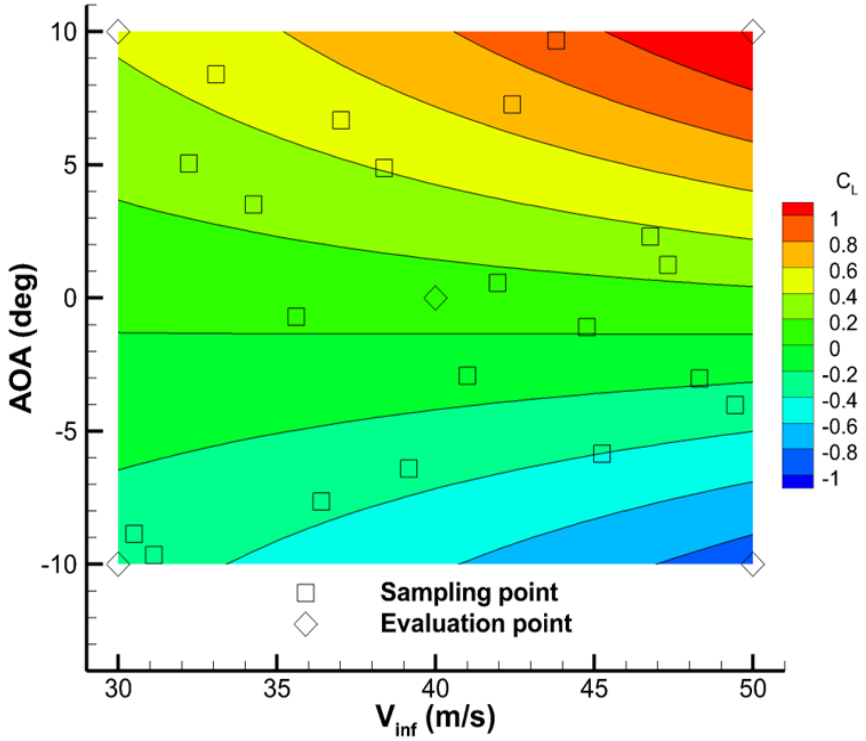
- 정확도 검증
  - NACA2412 익형 아음속 정상 유동 해석 – 스냅샷 개수 추가
    - 스냅샷 데이터 30개
      - 자유류 속도 30, 34, 38, 42, 46, 50 m/s
      - 받음각 0.0, 2.5, 5.0, 7.5, 10.0 도
    - CFD (좌), POD ROM (우)
    - 개선된 오차 0.148%
    - 적절한 스냅샷 개수 선정 필요
      - 강형민, “적합직교분해 기법에서의 효율적인 스냅샷 선정을 위한 고유값 분석”, 2017





POD+ANN 차수축소모델 검증

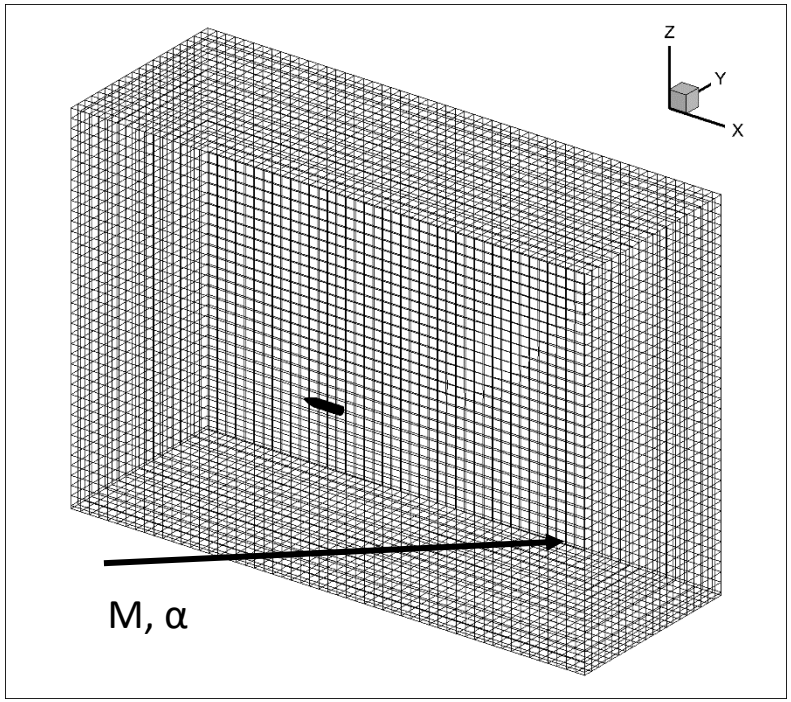
- 데이터베이스 생성 성능 검증
  - NACA2412 익형 아음속 정상 유동 해석
    - 자유류 속도 30m/s ~ 50m/s
    - 받음각 -10º ~ 10º
    - 20개의 입력 매개 변수 표본 사용
    - 5개 해석 결과 누적 시 POD 수행, 이후 해석 결과 초기 조건 추정
    - 소요 시간 : 1개 케이스 해석 소요 시간의 6.55배

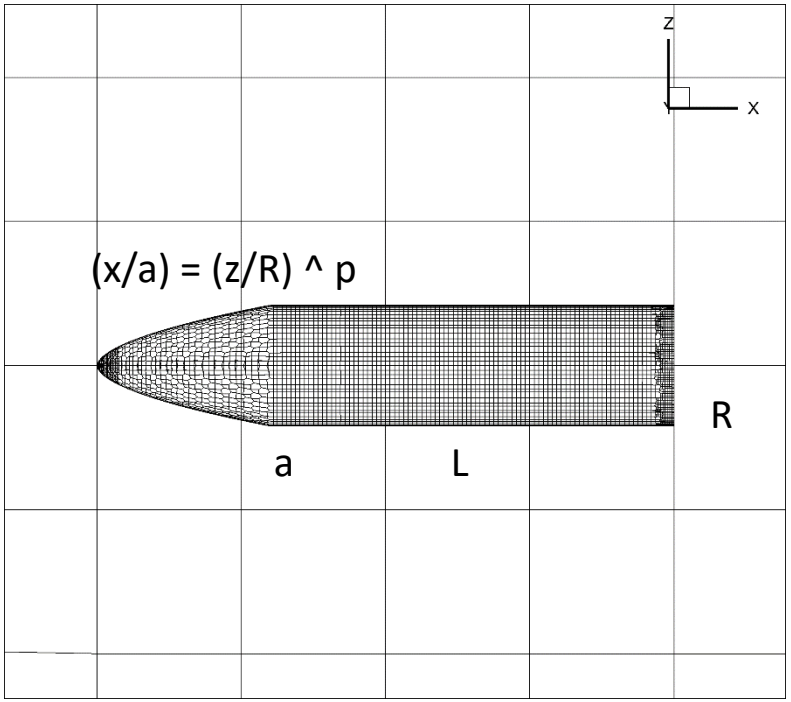


Case No.	V_∞ (m/s)	AOA (deg)	Lift coefficient		Error (%)
			CFD	ANN	
1	30	-10	-0.321	-0.321	0.005
2	30	10	0.433	0.433	0.008
3	40	0	0.0965	0.0965	0.022
4	50	-10	-0.900	-0.908	0.878
5	50	10	1.220	1.211	0.790

형상변형 / 차수축소모델 적용 예시

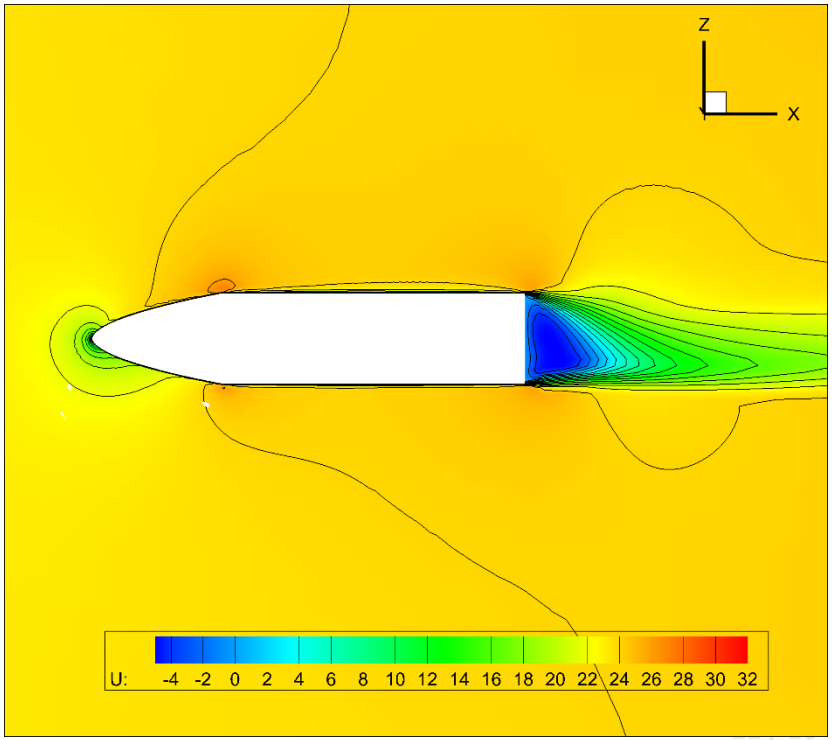
- 매개변수화된 3차원 비행체 외부 공력 해석
  - 전두부 형상 곡선 차수 (p), 전두부 길이 (a), 전체 길이 (L), 반지름 (R)
  - 자유류 받음각 (α), 마하수 (M)
  - 총 6개 입력 매개 변수, 격자 개수 약 48만

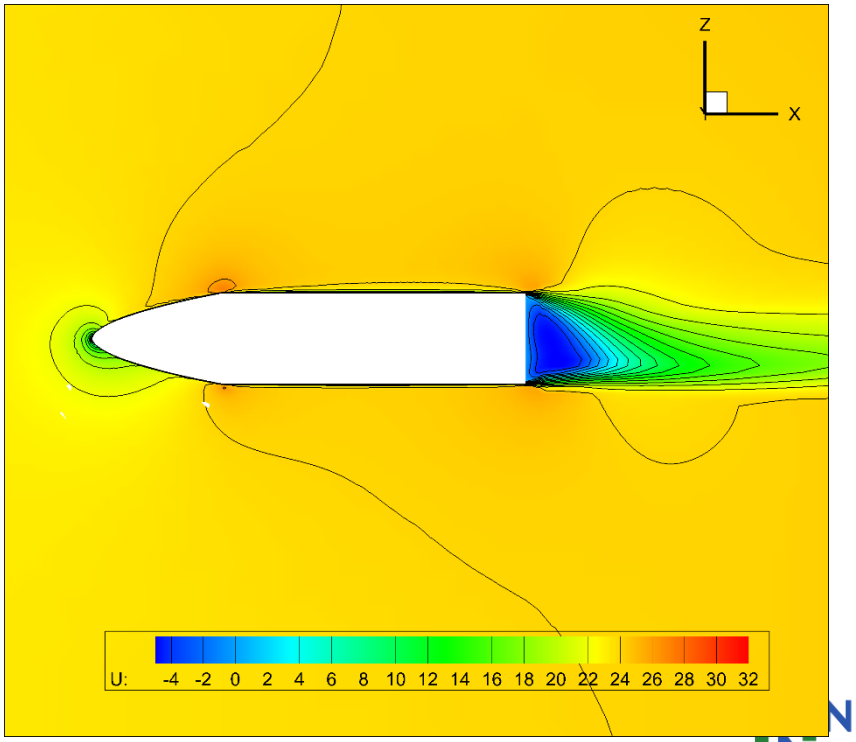




형상변형 / 차수축소모델 적용 예시

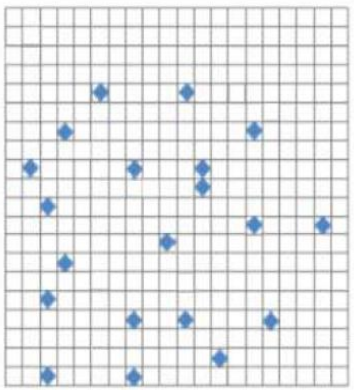
- 매개변수화된 3차원 비행체 외부 공력 해석
  - 135개 스냅샷 데이터를 CFD 해석으로 획득
  - POD ROM 구성 (1분 소요) + 유동장 재구성 (1초 미만 소요)
  - 신규 형상/유동 조건에 대한 해석 결과를 예측하여 비교
  - CFD 해석 (30분 소요) 대비 소요 시간 대폭 저감, 오차 1.09%
  - 형상최적설계 문제에 응용 가능한 기법 제시

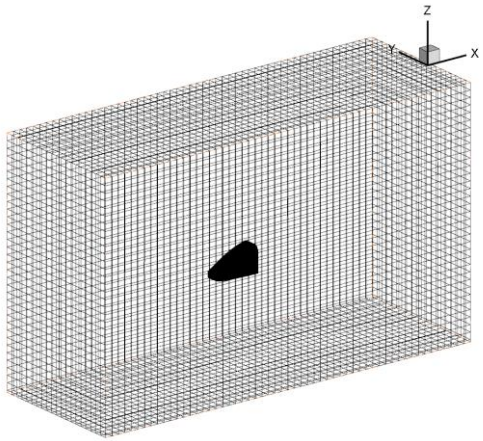


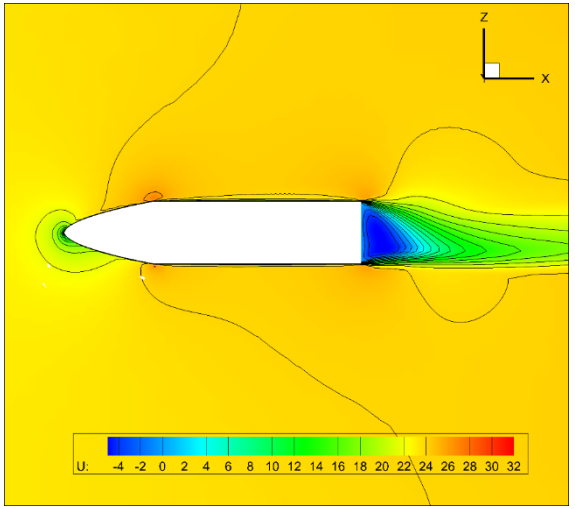


요약

- 연구 성과
  - 형상 매개변수화, 격자 변형, 차수축소모델 구성을 통한 형상최적설계 방법 제시
  - 정확도 및 성능 검증 완료
  - 반복적인 격자생성 / 유동해석의 부하를 저감할 수 있음을 확인
  - 실제 형상최적설계 문제에 응용 및 지속적 보완 예정







감사합니다