Nonlinear Reduced Order Modeling for Transonic Flows via Manifold Learning

Thomas Franz, Ralf Zimmermann, Stefan Görtz

DLR

Motivation

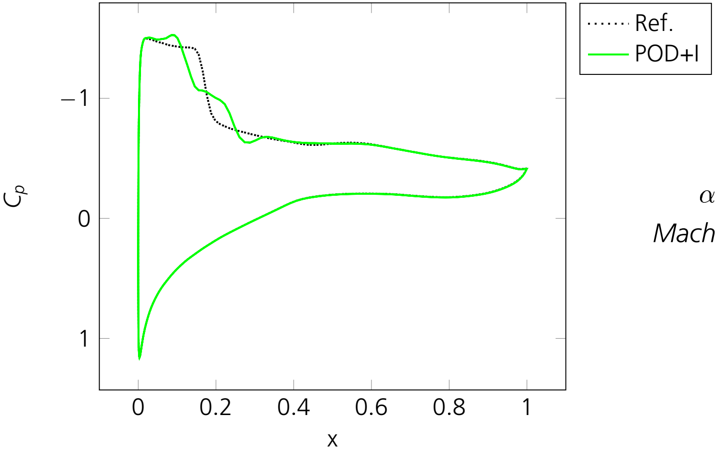
Objective: transonic flows

$\rightarrow$ Shocks ($\widehat{=}$ strong non-linearities) appear, which move along the flow domain as the parameters are varied

$\rightarrow$ Difficult to predict shocks by ROMs, because most ROMs assume some linear coherences or else require a large amount of full-order data input

How can we improve the prediction of shock dominated CFD solutions using ROMs?

Example - 2D NACA64A010 airfoil



$\alpha = 8.85$

$Mach = 0.79$

Outline

Introduction to Manifold Learning (ML)

- Given: $W = \{W^1, \ldots, W^m\} \subset \mathcal{W} \subset \mathbb{R}^n$ sampled from an unknown data manifold $\mathcal{W}$ with intrinsic dimensionality $\dim(\mathcal{W}) = d < n$
- Goal: find embedding mapping

$$h : W \subset \mathbb{R}^n \to Y = \{y^1, \ldots, y^m\} \subset \mathbb{R}^d,$$

while preserving the geometry of the data W as much as possible

$\Rightarrow$ The obtained embedding Y is a good representation for the high dimensional dataset W

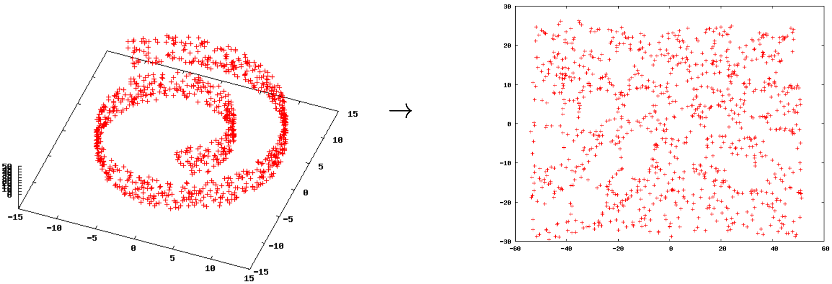
The main application of the established ML methods is data compression, image processing or data visualization.

Isomap

- Dimensionality Reduction / Manifold Learing method
- Based on Multidimensional Scaling (MDS)
- Attempts to preserve geodesic pairwise distances of input data

Isomap

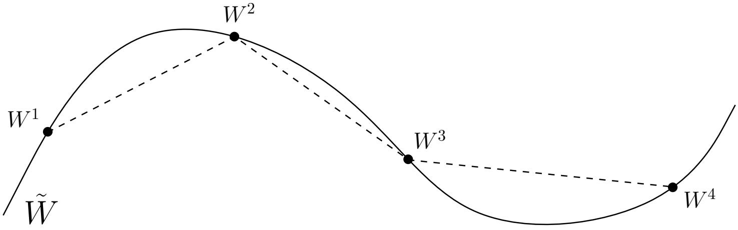
- Dimensionality Reduction / Manifold Learing method
- Based on Multidimensional Scaling (MDS)
- Attempts to preserve geodesic pairwise distances of input data



2d embedding of the Swiss roll with Isomap

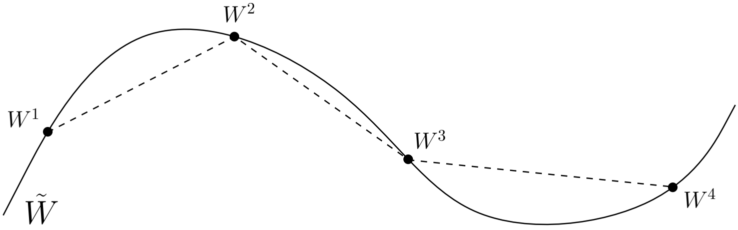
Approximating geodesic distances

- For close-by points: *Euclidean* distance ≈ geodesic distance
- For far away points: length of an *Euclidean* polygonal curve through close-by points



Approximating geodesic distances

- For close-by points: *Euclidean* distance ≈ geodesic distance
- For far away points: length of an *Euclidean* polygonal curve through close-by points



⇒ Graph-theoretical shortest paths problem

Metric Multidimensional Scaling (MDS)

- Maps high dimensional data $W = \{W^1, \ldots, W^m\} \subset \mathbb{R}^n$ to low dimensional representation $Y = \{y^1, \ldots, y^m\} \subset \mathbb{R}^d$ featuring *Euclidean* inter-point distances that (almost) equal the inter-point distances of the original data, i.e., $\mathrm{dist}(W^i, W^j) \simeq \|y^i, y^j\|_2$.
- Yields the best d-dimensional *Euclidean* embedding of the given data.
- Embedding is obtained by solving an Eigenvalue Decomposition.

Isomap in detail

Input: $W = \{W^1, \ldots, W^m\}, d, k$

1. construct weighted k-neighborhood graph (e.g. via k-d tree) to obtain euclidean distance matrix:

$$D_W(i,j) = \begin{cases} ||W^i - W^j||_2 & \text{if i, j are neighbors} \\ \infty & \text{else} \end{cases}$$

Isomap in detail

Input: $W = \{W^1, \dots, W^m\}, d, k$

1. construct weighted k-neighborhood graph (e.g. via k-d tree) to obtain euclidean distance matrix:

$$D_W(i,j) = \begin{cases} ||W^i - W^j||_2 & \text{if i, j are neighbors} \\ \infty & \text{else} \end{cases}$$

2. compute shortest paths based on D_W (e.g. via Floyd-Warshall) to obtain geodesic distance matrix D_G

Isomap in detail

Input: $W = \{W^1, \dots, W^m\}, d, k$

1. construct weighted k-neighborhood graph (e.g. via k-d tree) to obtain euclidean distance matrix:

$$D_W(i,j) = \begin{cases} ||W^i - W^j||_2 & \text{if i, j are neighbors} \\ \infty & \text{else} \end{cases}$$

2. compute shortest paths based on D_W (e.g. via Floyd-Warshall) to obtain geodesic distance matrix D_G
3. apply MDS to distance matrix D_G to obtain d-dimensional representation $Y = \{y^1, \dots, y^m\}$

$\Rightarrow$ Isometric embedding

Outline

Situation

- Given: Embedding $Y = \{y^1, \ldots, y^m\} \subset \mathbb{R}^d$ corresponding to a unknown data manifold $\mathcal{W}$ and new point $y^* \in \mathbb{R}^d$
- Goal: Find $W^* \subset \mathbb{R}^n$

Situation

- Given: Embedding $Y = \{y^1, \ldots, y^m\} \subset \mathbb{R}^d$ corresponding to a unknown data manifold $\mathcal{W}$ and new point $y^* \in \mathbb{R}^d$
- Goal: Find $W^* \subset \mathbb{R}^n$

Idea

- Nearest neighbors $\{y^j \mid j \in \mathcal{I}\}$ to y^* correspond isometrically to the nearest neighbors $\{W^j \mid j \in \mathcal{I}\}$ on the data manifold.

$\Rightarrow$ affine reconstruction of y^* by its N nearest neighbors should yield a good weighting to construct a linear combination of the corresponding high dimensional snapshots

Local approximate inverse mapping

Input: y^*, k

1. identify k nearest neighbors of y^* among the embedding $Y = \{y^1, \ldots, y^m\}$. Let N_0 denote the set of indices of the k nearest neighbors.

Local approximate inverse mapping

Input: y^*, k

1. identify k nearest neighbors of y^* among the embedding $Y = \{y^1, \ldots, y^m\}$. Let N_0 denote the set of indices of the k nearest neighbors.
2. compute $\min_{w \in \mathbb{R}^{|N_0|}} ||y^* - \sum_{j \in N_0} w_j y^j||$ s.t. $\sum_{j \in N_0} w_j = 1$

Local approximate inverse mapping

Input: y^*, k

1. identify k nearest neighbors of y^* among the embedding $Y = \{y^1, \dots, y^m\}$. Let N_0 denote the set of indices of the k nearest neighbors.
2. compute $\min_{w \in \mathbb{R}^{|N_0|}} ||y^* - \sum_{j \in N_0} w_j y^j||$ s.t. $\sum_{j \in N_0} w_j = 1$
3. compute $W^* := \sum_{j \in N_0} w_j W^j$ to get a corresponding high dimensional solution

DLR

Local approximate inverse mapping

Input: y^*, k

1. identify k nearest neighbors of y^* among the embedding $Y = \{y^1, \ldots, y^m\}$. Let N_0 denote the set of indices of the k nearest neighbors.
2. compute $\min_{w \in \mathbb{R}^{|N_0|}} ||y^* - \sum_{j \in N_0} w_j y^j||$ s.t. $\sum_{j \in N_0} w_j = 1$
3. compute $W^* := \sum_{j \in N_0} w_j W^j$ to get a corresponding high dimensional solution

Step 2 can be replaced by a linear system of equations, which appears by setting the gradient of the corresponding *Lagrange* function to zero.

Outline

Set up

Parameter configurations: $p^i \in \mathbb{R}^k,\ i = 1, \dots, m$

CFD solution snapshots: $W = \{W^1, \dots, W^m\}, W^i := W(p^i) \in \mathbb{R}^n$

Set up

Parameter configurations: $p^i \in \mathbb{R}^k,\ i = 1, \ldots, m$
CFD solution snapshots: $W = \{W^1, \ldots, W^m\}, W^i := W(p^i) \in \mathbb{R}^n$

ROM via Isomap + Interpolation

1. Apply Isomap to the data set W to obtain the embedding $Y = \{y^1, \ldots, y^m\} \subset \mathbb{R}^d$ (offline)

Remark: Process chain is similar to POD + Interpolation.

Set up

Parameter configurations: $p^i \in \mathbb{R}^k,\ i = 1, \dots, m$
CFD solution snapshots: $W = \{W^1, \dots, W^m\}, W^i := W(p^i) \in \mathbb{R}^n$

ROM via Isomap + Interpolation

1. Apply Isomap to the data set W to obtain the embedding $Y = \{y^1, \dots, y^m\} \subset \mathbb{R}^d$ (offline)
2. Interpolate representative $y^* \in \mathbb{R}^d$ for new parameter configuration $p^* \in \mathbb{R}^k$, where the interpolation set is given by $\{(p^i, y^i)\}_{i=1}^m$ (online)

Remark: Process chain is similar to POD + Interpolation.

Set up

Parameter configurations: $p^i \in \mathbb{R}^k,\ i = 1, \ldots, m$

CFD solution snapshots: $W = \{W^1, \ldots, W^m\}, W^i := W(p^i) \in \mathbb{R}^n$

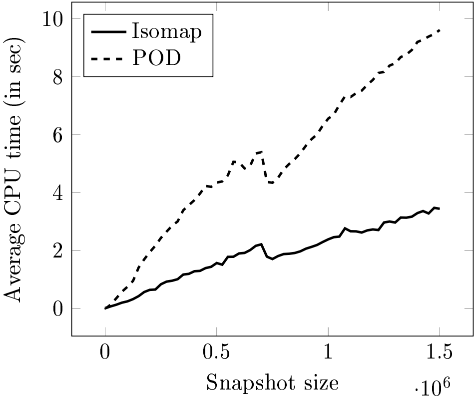
ROM via Isomap + Interpolation

1. Apply Isomap to the data set W to obtain the embedding $Y = \{y^1, \ldots, y^m\} \subset \mathbb{R}^d$ (offline)
2. Interpolate representative $y^* \in \mathbb{R}^d$ for new parameter configuration $p^* \in \mathbb{R}^k$, where the interpolation set is given by $\{(p^i, y^i)\}_{i=1}^m$ (online)
3. Apply back-mapping to y^* to obtain a prediction of the CFD solution $W^* = W(p^*)$ (online)

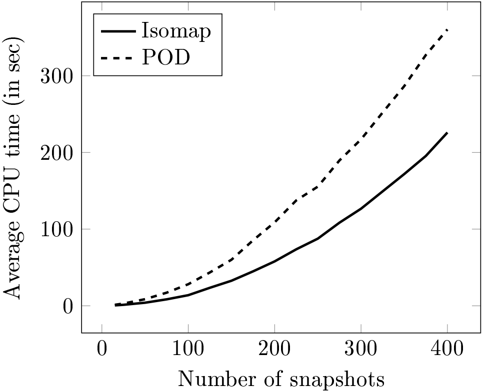
Remark: Process chain is similar to POD + Interpolation.

Outline

DLR



Offline costs depending on the **snapshot size** using 30 snapshots (without building a interpolation model).



Offline costs depending on the **number of snapshots** with a fixed snapshot size of 10^6 (without building a interpolation model).

Computational complexity

online stage

- $\mathcal{O}(dm)$ for RBF interpolation
- $\mathcal{O}(N_{rec} \log m)$ for finding the N_{rec} nearest neighbors
- $\mathcal{O}(N_{rec}^3)$ to calculate the weights for the back-mapping
- $\mathcal{O}(N_{rec} n)$ to map the reduced-ordner coordinates onto the manifold

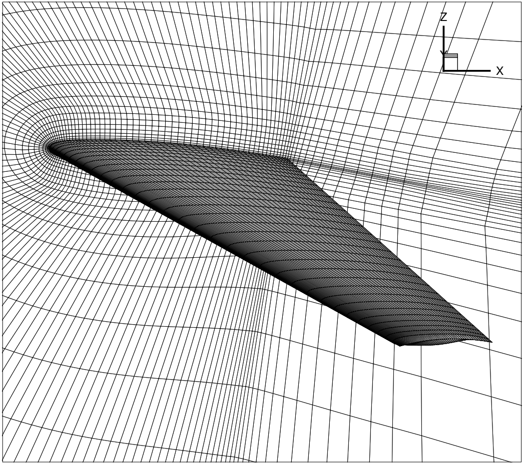
- Prediciton of full-order solutions scales **linearly** in n
- Prediction of reduced-order coordinates is **independet** of n
  $\Rightarrow$ Qualifies as a real-time method

DLR

Outline

DLR

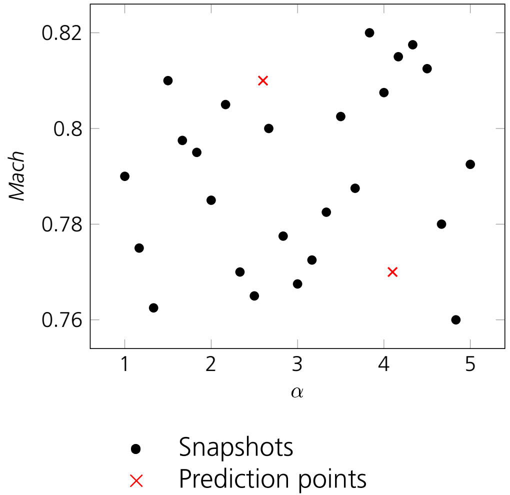
LANN



LANN wing

Grid size: 237,373

LANN



Latin hypercube sampling
(25 C_p snapshots $\in \mathbb{R}^{237373}$)

CFD Code: TAU Euler

CPU times:

Isomap+TPS: 0.72s
Prediction: $\leq$ 0.07s

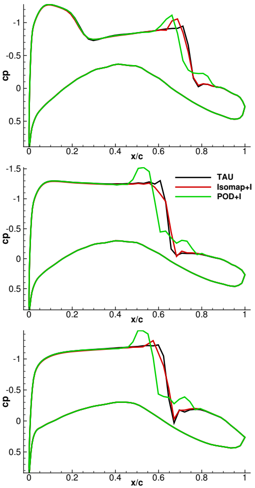
POD+TPS: 1.62s
Prediction: $\sim$ 0.03s

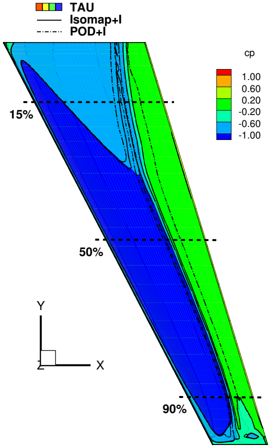
TAU: $\sim$ 450s

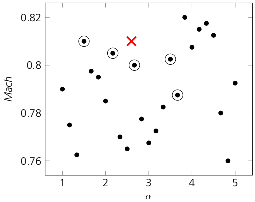
Isomap parameter

- Dimension of embedding space: 2
- Number of nearest neighbors for detecting the manifold: 7
- Number of nearest neighbors for the back-mapping: 5

LANN



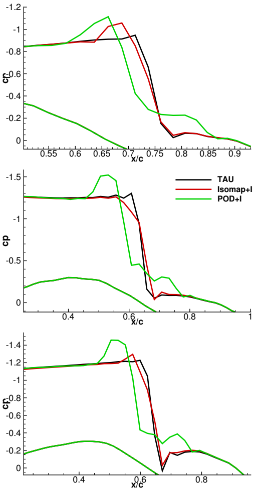


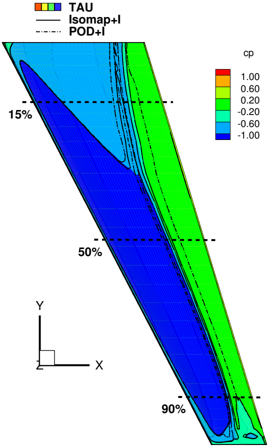


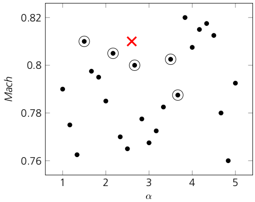
$$\alpha = 2.6$$

$$Mach = 0.81$$

LANN



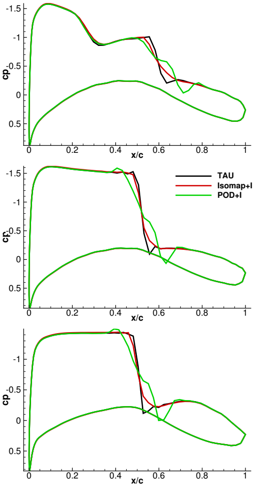


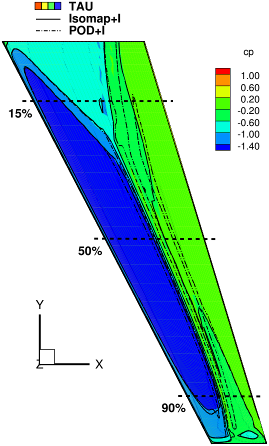


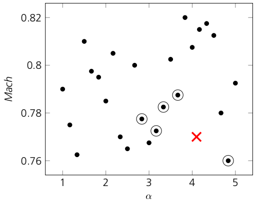
$\alpha = 2.6$

$Mach = 0.81$

LANN



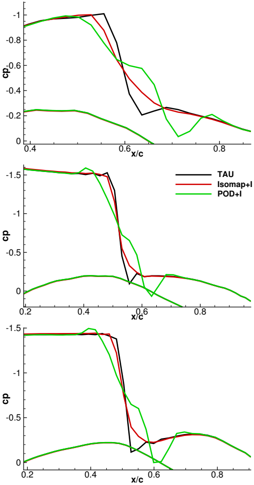


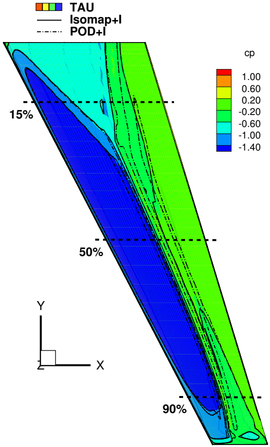


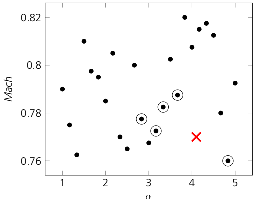
$$\alpha = 4.1$$

$$Mach = 0.77$$

LANN







$$\alpha = 4.1$$

$$Mach = 0.77$$

DLR

Outline

Outlook

- Integration of a residual based optimization method for the Isomap coefficients (e.g. LeastSquare ROM) to improve the solutions considering flow physics

- “Manifold filling” adaptive sampling strategy

Appendix

Computational complexity

Offline stage

Dominated by the terms

- $\mathcal{O}(nm \log m)$ for constructing the kd-tree
- $\mathcal{O}(m^3)$ for finding all shortest pathes (Ford-Warshall)

online stage

- $\mathcal{O}(dm)$ for RBF interpolation
- $\mathcal{O}(N_{rec} \log m)$ for finding the N_{rec} nearest neighbors
- $\mathcal{O}(N_{rec}^3)$ to calculate the weights for the back-mapping
- $\mathcal{O}(N_{rec} n)$ to map the reduced-ordner coordinates onto the manifold

$\Rightarrow$ Both stages scale *linearly* in n

Proper Orthogonal Decomposition

Model parameters: $p^i \in \mathbb{R}^d, i = 1, \ldots, m$
CFD solution snapshots: $W^i := W(p^i) \in \mathbb{R}^n, i = 1, \ldots, m$
Snapshot matrix: $Y := (W^1, \ldots, W^m) \in \mathbb{R}^{n \times m}$

$\rightarrow$ Compute $m \times m$ eigenvalue decomposition

$$Y^T Y V^j = \lambda_j V^j, \quad j = 1, \ldots, m$$

$\Rightarrow$ $span\{U^1, \ldots, U^m\} = span\{W^1, \ldots, W^m\}$,
where $U^j = \frac{1}{\sqrt{\lambda_j}} Y V^j \in \mathbb{R}^n$ with $\langle V^i, V^j \rangle = \delta_{ij}$ and $\lambda_1 \geq \lambda_2 \geq \cdots > 0$

Radial Basis Function interpolation 1/2

- Sample points: $X = \{x^1, \ldots, x^m\} \subset \mathbb{R}^k$,
- Responses: $Y = \{y^1, \ldots, y^m\} \subset \mathbb{R}$,

obtained by evaluating a function $f(x^i) = y^i$, $i = 1, \ldots, m$.

Simplest Radial Basis Function model:

$$\hat{f}(x) = w^T \boldsymbol{\psi} = \sum_{i=1}^{m} w_i \psi(\|x - x^i\|),$$

where $\boldsymbol{\psi} = (\psi(\|x - x^1\|), \ldots, \psi(\|x - x^m\|)^T \in \mathbb{R}^m$ and $\psi : r \mapsto \psi(r)$ is a radial basis function (RBF).

Radial Basis Function interpolation 2/2

The weights $w = (w_1, \dots, w_m)$ are determined by the interpolation conditions

$$\hat{f}(x^j) = y^j = \sum_{i=1}^{m} w_i \psi(\|x^j - x^i\|) = y^j, \quad j = 1, \dots, m,$$

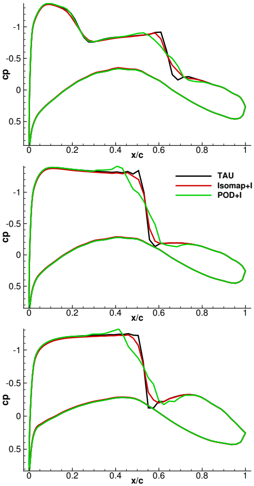
which gives the linear equation system

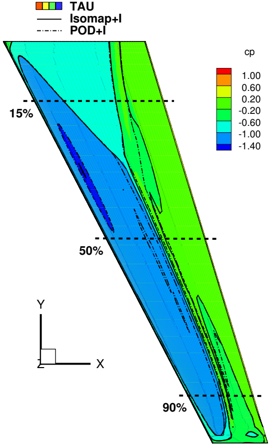
$$\Psi w = y, \tag{1}$$

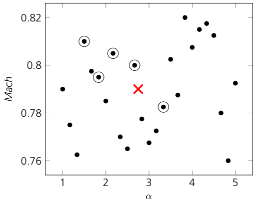
where Ψ is the so called *Gram matrix* with entries $\Psi_{i,j} = \psi(\|x^i - x^j\|)$, $i, j = 1, \dots, m$. If the matrix Ψ is regular, then the model becomes

$$\hat{f}(x) = y^T \Psi^{-1} \psi.$$

LANN



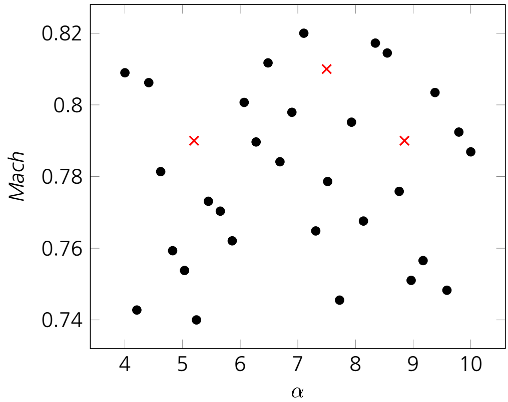




$$\alpha = 2.75$$

$$Mach = 0.79$$

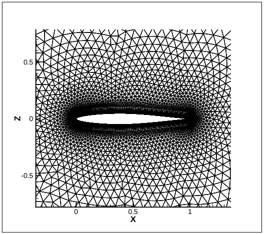
NACA64A010



● Snapshots
× Prediction points

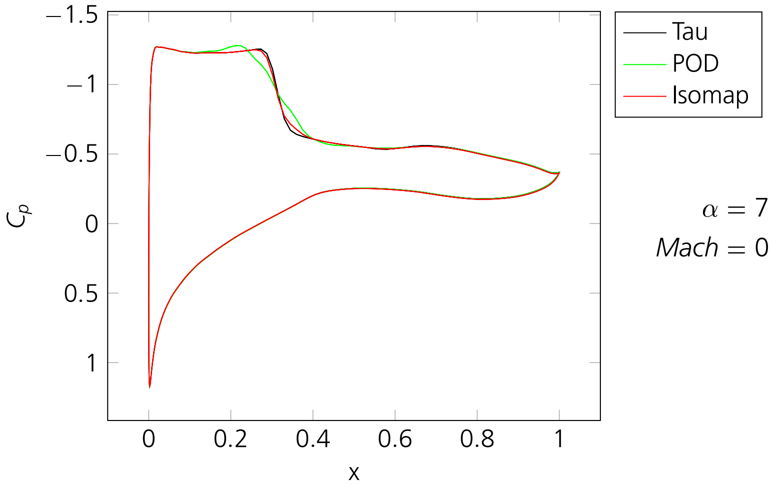
Latin hypercube sampling

(30 C_p surface snapshots $\in \mathbb{R}^{400}$)



NACA64A010 airfoil

NACA64A010

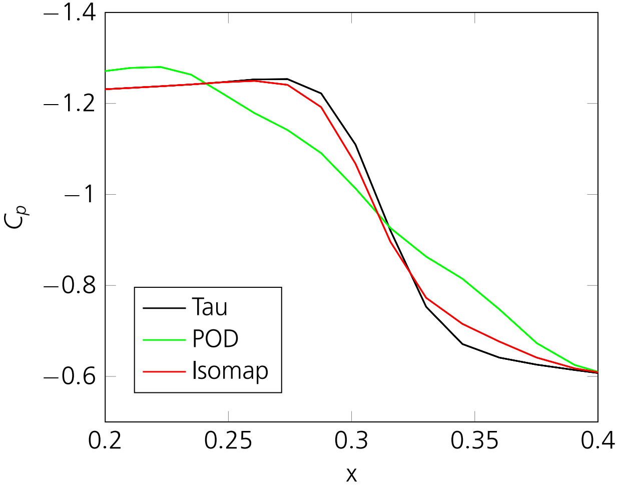


$\alpha = 7.5$

$Mach = 0.81$

DLR

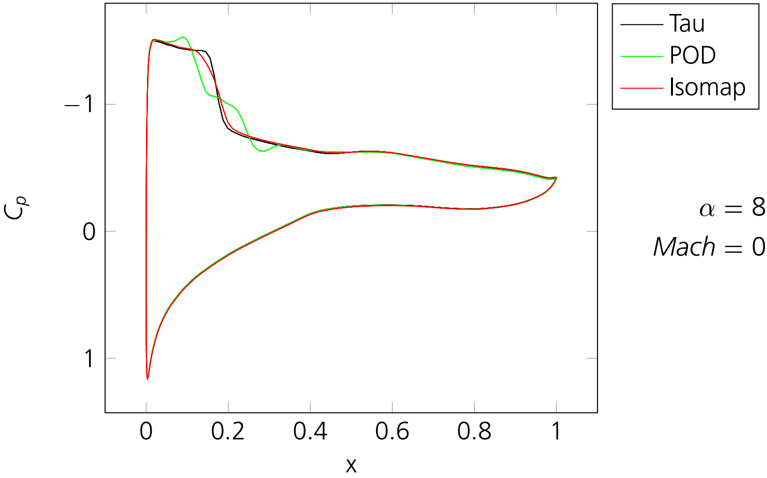
NACA64A010



$\alpha = 7.5$

$Mach = 0.81$

NACA64A010

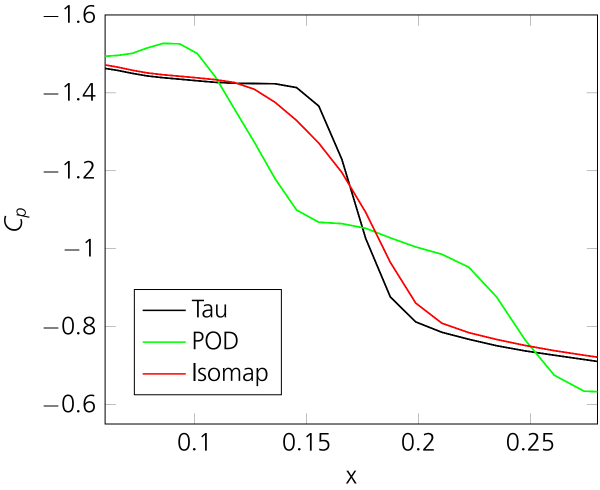


$\alpha = 8.85$

$Mach = 0.79$

DLR

NACA64A010



$\alpha = 8.85$

$Mach = 0.79$