Project 1: Data Compression and Model Reduction

AE 523, Computational Fluid Dynamics, Fall 2018
Runda Ji

I. Introduction

In this project, we apply the singular value decomposition (SVD) to obtain the singular values and the corresponding basis vectors based on the 20 training snapshots. In order to compress the data, the test snapshots are projected on the basis vectors obtained. Linear and quadratic models are used to model the influence of parameters on the CFD solution, where least-squares regression is used to determine the coefficients within the models.

II. Methods

The quantities used in this project are listed in table 1.

Table 1. Nomenclature

Symbol	Description
ρ	Density
u	Velocity, x-component
v	Velocity, y-component
E	Total energy
p	Pressure
γ	Ratio of specific heats
a	Speed of sound
m	Mach number
S	State matrix
s_i	i^{th} column in S, state vector
$\tilde{s}_i$	Projected state vector
U	Left singular vector (matrix)
u_i	i^{th} column in U, i^{th} basis vector
Σ	Singular value matrix
σ_i	i^{th} singular value
V	Right singular vector (matrix)
v_i	i^{th} column in V, i^{th} basis vector
c_{ij}	Projection of state s_i on the i^{th} basis u_i
ϵ	L_2 error

A. Singular value decomposition

The method of singular value decomposition (SVD) provide us an approach to extract the characteristic from the given data. Specifically, the SVD for a given matrix S can be expressed as,

$$S = U\Sigma V^T. \tag{1}$$

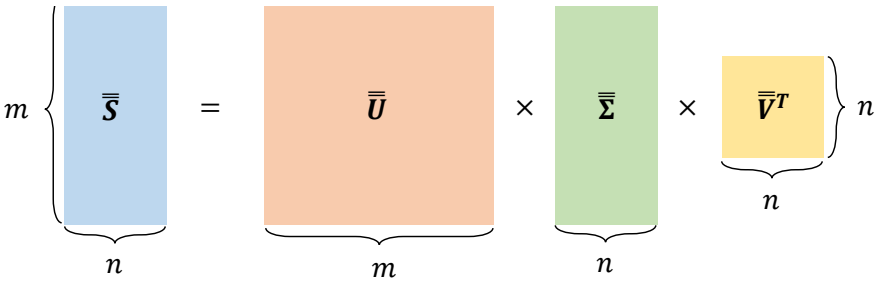

Figure 1. Matrix multiplication in singular values decomposition

Where U and V are the left and right singular vector respectively. Note that both U and V are unitary matrix, i.e. $U^TU = 1$ and $V^TV = 1$. The singular values σ_i $(i = 1 \cdots n)$ are stored in matrix Σ, which can be considered as the significance or importance of the corresponding basis.

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_n \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}_{m\times n}. \tag{2}$$

Transpose both side of equation (1), we have,

$$\begin{aligned} S^T &= (U\Sigma V^T)^T, \\ &= V\Sigma^T U^T. \end{aligned} \tag{3}$$

Hence,

$$\begin{aligned} S^TS &= V\Sigma^T U^T U\Sigma V^T, \\ S^TSV &= V\Sigma^T\Sigma V^T V, \\ S^TSV &= V\Sigma^T\Sigma, \\ S^TSV &= \Sigma^T\Sigma V. \end{aligned} \tag{4}$$

The last step hold true because $\Sigma^T\Sigma$ is a diagonal matrix,

$$\Sigma^T\Sigma = \begin{bmatrix} \sigma_1^2 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_n^2 \end{bmatrix}_{n\times n}, \tag{5}$$

where σ_i^2 can be considered as the eigenvalues of matrix S^TS. While the i^{th} column of matrix V is the corresponding eigenbasis.

Multiply matrix V on both side of equation (1) and note that V is unitary matrix,

$$\begin{aligned} SV &= U\Sigma V^T V, \\ SV &= U\Sigma. \end{aligned} \tag{6}$$

v_i and u_i are the i^{th} column of matrix U and V respectively. So equation (6) can be re-written as,

$$\begin{aligned} Sv_i &= u_i\sigma_i, \\ u_i &= \frac{Sv_i}{\sigma_i}. \end{aligned} \tag{7}$$

B. Least-squares regression

The basic idea of the least-squares regression is to minimize the error function S,

$$S = \sum r_i^2, \tag{8}$$

where r_i is the residual between the actual value of dependent variable and the approximated value.

$$r_i = c_i - f(\vec{X}_i, \vec{\beta}). \tag{9}$$

In equation (9), c_i is the actual value at $\vec{X}_i$, while $f(\vec{X}_i, \vec{\beta})$ is the approximation at $\vec{X}_i$.

To be more specific, in this project, when applying the linear model, $\vec{X}_i = (M_i, \alpha_i)$, $\vec{\beta} = (a_0, a_1, a_2)$.

$$f(\vec{X}_i, \vec{\beta}) = a_0 + a_1 M_i + a_2 \alpha_i. \tag{10}$$

While applying the quadratic model, $\vec{X}_i = (M_i, \alpha_i)$, $\vec{\beta} = (b_0, b_1, b_2, b_3, b_4, b_5)$.

$$f(\vec{X}_i, \vec{\beta}) = b_0 + b_1 M_i + b_2 \alpha_i + b_3 M_i^2 + b_4 M_i \alpha_i + b_5 \alpha_i^2. \tag{11}$$

III. Questions and Tasks

A. Snapshot visualization

In this section, we will visualize the given snapshots and plot the contour of local Mach number. The first step is to figure out the local sound speed a. From the given state files we can obtain,

$$u_j = [\rho, \rho u, \rho v, \rho E, \rho\tilde{\nu}]^T. \tag{12}$$

Using the provided terms, the pressure can be calculated by,

$$p = (\gamma - 1)[\rho E - \frac{1}{2}\rho(u^2 + v^2)]. \tag{13}$$

So the local speed of sound a can be written as,

$$a = \sqrt{\gamma \frac{p}{\rho}}. \tag{14}$$

Hence the Mach number can be obtained,

$$M = \frac{u^2 + v^2}{a}. \tag{15}$$

The contour of Mach number of training snapshot 1 and 20 are depicted in figure 2.

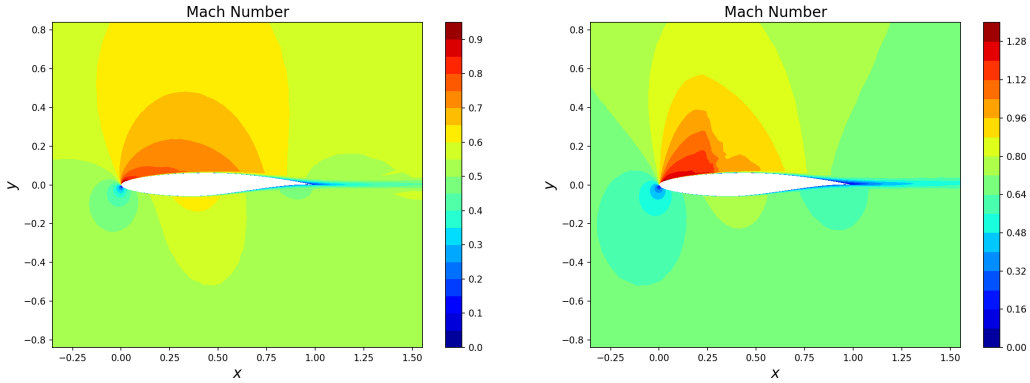

Figure 2. Contour for training solution 1 (left) and training solution 20 (right).

The contour of Mach number of test snapshot 55 and 90 are depicted in figure 3.

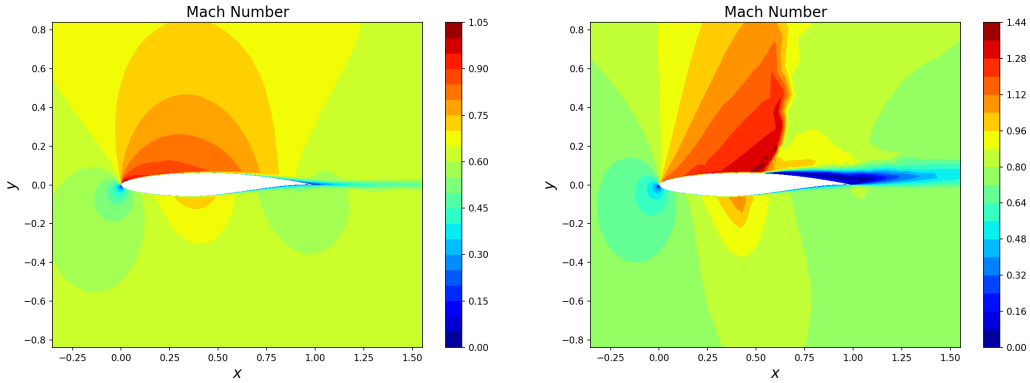

Figure 3. Contour for training solution 55 (left) and training solution 90 (right).

The contour of Mach number of the rest snapshots can be found in folder '.\figure\question_1'.

B. Proper orthogonal decomposition

Within this section, we will perform a decomposition of the 20 training snapshots. Specifically, to perform a singular-value decomposition (SVD) using Python, we may directly use the built-in function 'numpy.linalg.svd()' and obtain matrix U, Σ and V^T.

Note that performing a full SVD, which gives all columns (basis vectors) in matrix U, can be extremely time and memory consuming. Since we are only interested in the first 20 basis vectors that corresponding to the singular values, we merely need to perform a truncated SVD by simply adding 'full_matrices=False' as an argument of the built-in function. The singular values obtained from 'numpy.linalg.svd()' is depicted as follows,

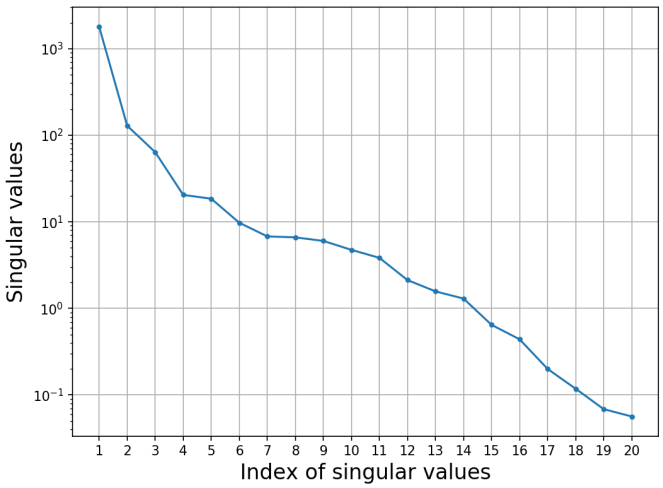

Figure 4. Singular values ranked form largest to smallest.

As we can see in figure 4, the singular values drop rapidly at first. The first singular value is nearly 10 times of the second one. The singular values keep decreasing relatively fast until the index reaches 5. The decreasing of singular values with index 6 through 10 are relatively slow, which indicates that the basis corresponding to these singular values are equally important. The singular values with index larger than 18 is less than 0.1, which means that the corresponding basis are not important and could be reasonably neglected.

The density and x-momentum components of the first four basis vectors are plotted in figure 5 and 6 respectively.

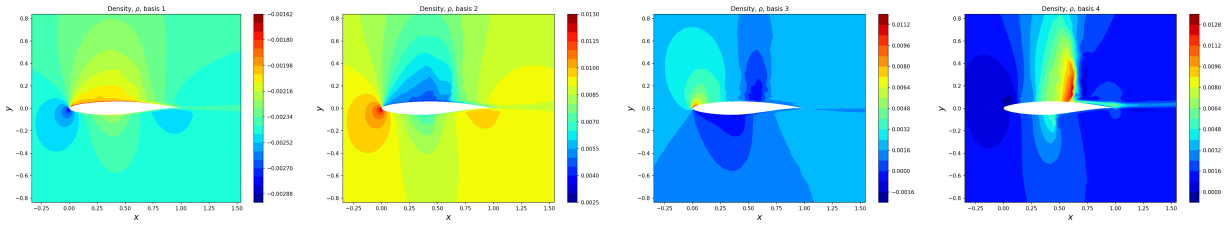

Figure 5. Density ρ of basis 1 through 4.

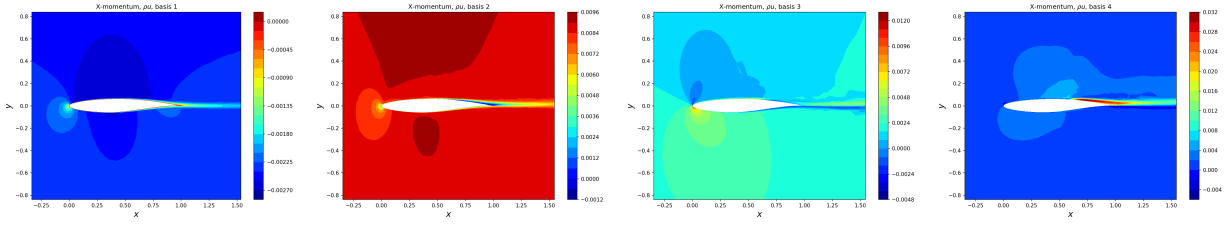

Figure 6. X-momentum ρu of basis 1 through 4.

In figure 5, the "density" of some basis are negative. Meanwhile, in figure 6, some basis have negative x-momentum, which is opposite with the main stream direction. Although it seems to be wrong, this is actually caused by the ambiguity of SVD. To be more specific, after adding negative signs in front of both U and V^T, equation (1) still hold true. Hence the negative density does NOT mean that the density is physically negative. Meanwhile, the negative x-momentum does NOT indicates that the x-momentum is toward $-x$ direction.

Alternatively, rather than directly using 'numpy.linalg.svd()', we could first find the eigenvalues and eigenvectors of matrix $S^T S$ and then compute the matrix U. Note that $S^T S$ is only a 20 by 20 matrix, so we could easily diagonalize it utilizing 'numpy.linalg.eig()'. Specifically, the eigenvalues e_i and eigenvectors V of $S^T S$ satisfy,

$$(S^T S)V = \begin{bmatrix} e_1 & 0 & 0 & \dots & 0 \\ 0 & e_2 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & e_{20} \end{bmatrix}_{20\times 20} V, \tag{16}$$

As shown in equation (1) - (5), the eigenvector matrix V of $S^T S$ is the right singular vector in equation (1). At the same time, e_i is the square of the singular values in matrix Σ, so $\sigma_i = \sqrt{e_i}$. The first 20 columns of matrix U can be easily obtained using equation (7).

Applying this alternative method, we can obtain the singular values σ_i, which are exactly the same as the singular values plotted in figure 4. However, some singular vectors obtained by this alternative method are different with those singular vectors in figure 5 and 6 by a negative sign. As we have mentioned, this negative sign does not matter since equation (1) always hold true after adding negative signs in front of both left singular vector U_i and right singular vector V_i.

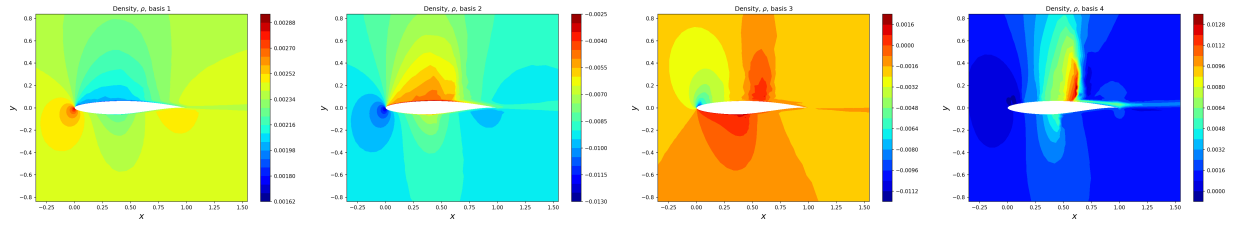

Figure 7. Density ρ of basis 1 through 4.

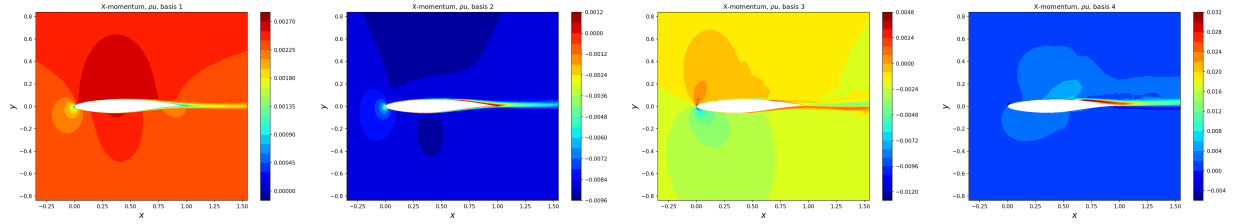

Figure 8. X-momentum ρu of basis 1 through 4.

C. Data compression

Utilizing the result obtained from the previous section, we are going to compress the test snapshots in this section. Use s_j to represent the state vector of a test snapshot, where $j = 1, 2, \ldots, 100$, the projection on basis u_i can be written as,

$$c_{ij} = |s_j| \cos\theta_{ij} = \frac{s_j \cdot u_i}{|u_i|}, \tag{17}$$

where θ_{ij} is the angle between state vector s_j and basis u_i. Note that we are only using n_b basis to represent the projected state vector $\tilde{s_j}$, i.e.

$$\tilde{s_j} = \sum_{i=1}^{n_b} c_{ij} u_i. \tag{18}$$

Hence the L_2 error between the exact state vector s_j and the projected state vector $\tilde{s_j}$ can be written as,

$$\epsilon = \sqrt{\frac{1}{N} \sum_{k=1}^{N} [(\tilde{s_j} - s_j)_k]^2}. \tag{19}$$

where $(\tilde{s_j} - s_j)_k$ is the k^{th} component (i.e. k^{th} row) of the vector. The total number of rows in the vector $N = 35225$.

As shown in figure 9, the discrete L_2 error between the original test snapshots and projected snapshots are plotted in (M, α) space. From figure 9 we can see that the L_2 error drops dramatically from using 2 basis to using 4 basis. This is in accordance with the singular values plotted in figure 4, where we may not be able to fully represent the state vector merely using the first two basis. Additionally, when we are using 16 basis to represent the state vector, the L_2 error is mostly zero within the low-speed, small-angle region.

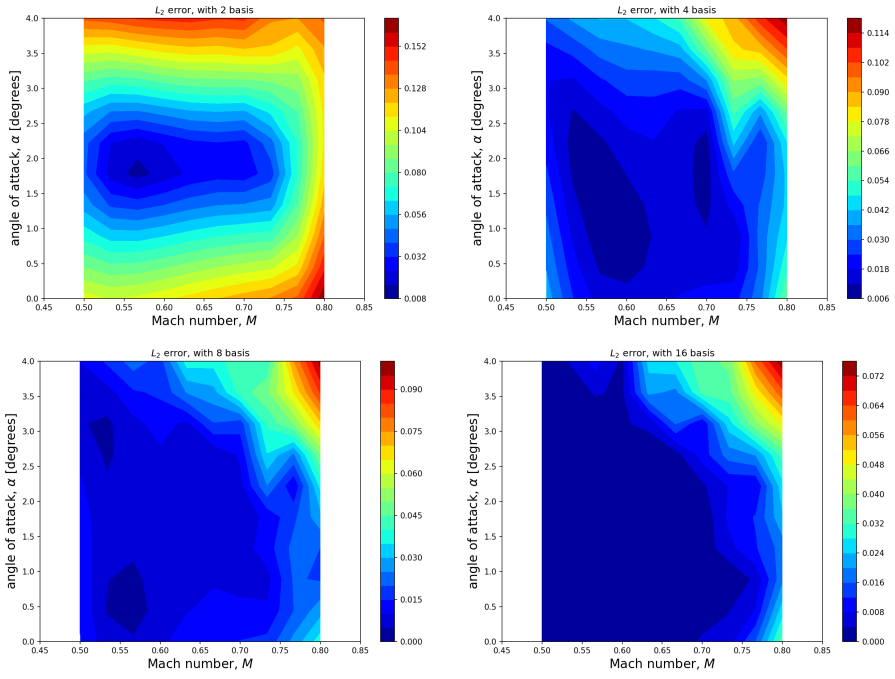

Figure 9. L_2 **error between the test snapshots and projected snapshots,** $n_b = 2, 4, 8, 16$

However, the error is quite large when the Mach number is high and the attack angle is large. So why there is such a significant error within the high-Mach-number, large-attack-angle region? From the right hand side of figure 3, which is the Mach number contour of test snapshot 90, we can see that there is a shock wave above the airfoil, which will cause discontinuity of $[\rho, \rho u, \rho v, \rho E, \rho\tilde{\nu}]$. The question is weather we can

represent these discontinuities correctly. Since within the training data set, there is no such cases that the Mach number is high while attack angle is large, our training set may be too limited to fully represent the test data set. Therefore the pattern (shock wave) exist in test snapshot 90 may not exist in those training snapshots. Although in figure 2, there is a shock wave above the airfoil of the train snapshot 20. The position and shape of that shock wave is quite different from the shock wave in test snapshot 90. Which indicates that the linear combination of the training snapshots may not be able to fully represent the test snapshots, and consequently introduce the error.

In order to directly compare the projected state vector with the original state vector, the Mach number contours of projected test snapshot 55 and 90 are depicted in figure 10.

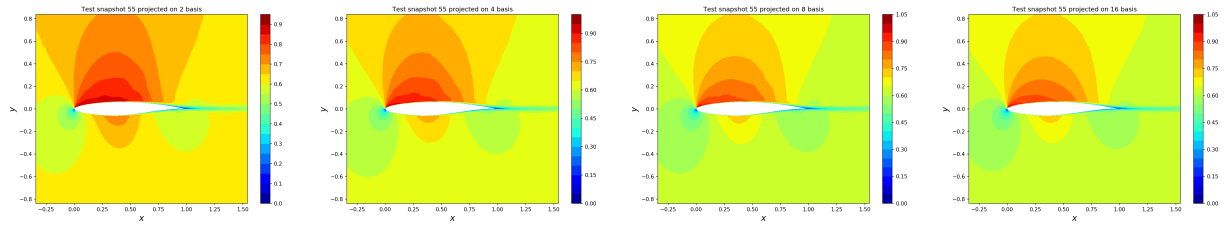

Figure 10. Mach number contours of projected test snapshot 55 on 2,4,8 and 16 basis

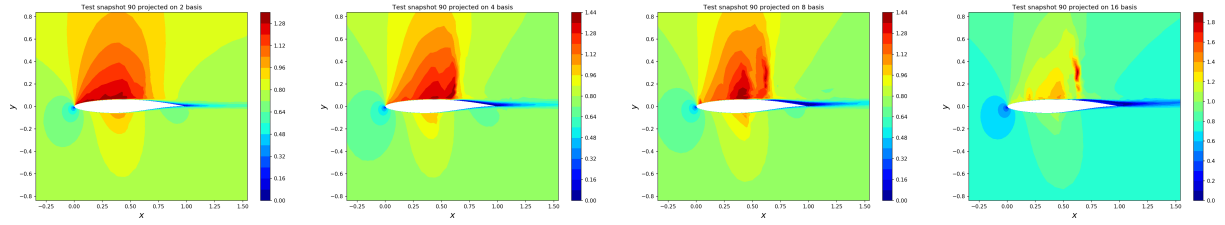

Figure 11. Mach number contours of projected test snapshot 90 on 2,4,8 and 16 basis

Comparing figure 10, figure 11 with figure 3, we can verify the conclusion that the accuracy of projection increases when using more basis vectors. As for test snapshot 55, the projection is already quite accurate when using 8 basis vectors. However, the accuracy of test snapshot 90 is relatively limited even if we are using 16 basis. This is in accordance with figure 9, since test snapshot 90 is in the region where both Mach number and attack angle are large, which corresponds to a significant L_2 error.

D. Parameter space modeling

In this section we will develop a linear and a quadratic model to quickly construct solution approximations at different (M, α). The key of least square regression is to minimize the error function. Specifically, we may use the built-in function 'scipy.optimize.leastsq' to obtain the coefficients. A loop over all 16 basis vectors is shown as follows, where c[i] are the coefficients in front of the i^{th} basis that we obtained from the training snapshots in question 3.

```
for i in range(0,16):
        c[i]=COEF_ALL[:,i];
        #coef in front basis 1 thru 16
        a[i],success=leastsq(func_error_linear,a_initial,args=(Malpha, c[i]));
        b[i],success=leastsq(func_error_quadratic,b_initial,args=(Malpha, c[i]));
```

Note that 'a_initial' and 'b_initial' are the initial guess of linear coefficients a_0, a_1, a_2 and quadratic coefficients $b_0, b_1, b_2, b_3, b_4, b_5$. 'Malpha' is a 20 by 2 matrix, the first column stores the Mach number of the 20 train snapshots, while the second column stores the attack angles of each snapshot. As for the error function, the error function corresponding to the linear model and quadratic model are shown as follows,

```
def func_error_linear(a,X,c):
    M=X[:,0];
    alpha=X[:,1];
    c_fitted=a[0]+a[1]*M+a[2]*alpha;
    error=c-c_fitted;
    return error;

def func_error_quadratic(b,X,c):
    M=X[:,0];
    alpha=X[:,1];
    c_fitted=b[0]+b[1]*M+b[2]*alpha+b[3]*M**2+b[4]*M*alpha+b[5]*alpha**2;
    error=c-c_fitted;
    return error;
```

When projected on 4 basis vectors, i.e. $n_b = 4$, the coefficient a_0, a_1, a_2 for linear model and $b_0, b_1, b_2, b_3, b_4, b_5$ for quadratic model are listed in table 2 and table 3 respectively.

Table 2. Coefficients of linear fitting model

Basis	a_0	a_1	a_2
1^{st} basis	-1165	1177	6
2^{nd} basis	-203	312	4
3^{rd} basis	4	29	-11
4^{th} basis	4	-4	0

Table 3. Coefficients of quadratic fitting model

Basis	b_0	b_1	b_2	b_3	b_4	b_5
1^{st} basis	-2166	4354	-1	-2453	4	0
2^{nd} basis	-442	1078	0	-595	4	0
3^{rd} basis	170	-494	-9	395	4	0
4^{th} basis	-287	908	1	-690	-8	0

The contour plot for discrete error L_2 in (M, α) space of $n_b = 2, 4, 6, 8$ is depicted in figure 12. As n_b increases from 2 to 4, the accuracy improves significantly. If we use more basis vectors, we will get more accurate result, however the improvement is quite limited. Generally speaking, the quadratic model has a better performance comparing with the linear model.

Specifically, when using the linear model, the L_2 error is quite large at the left and right boundaries, where the Mach number is small at left and large at right. Along the right and left boundaries, the L_2 error also increases when attack angle α increases. Additionally, within the region $0.6 < M < 0.7$ and $\alpha < 1$, the L_2 error is also non-negligible.

As for the quadratic model, the accuracy increases dramatically when using 4 basis vectors comparing with using 2 vectors. When using a quadratic model the L_2 error mainly located at the high-speed, large-angle region. It is worth noting that near the left boundary, which corresponds to low Mach number, the L_2 error is also non-neglectable.

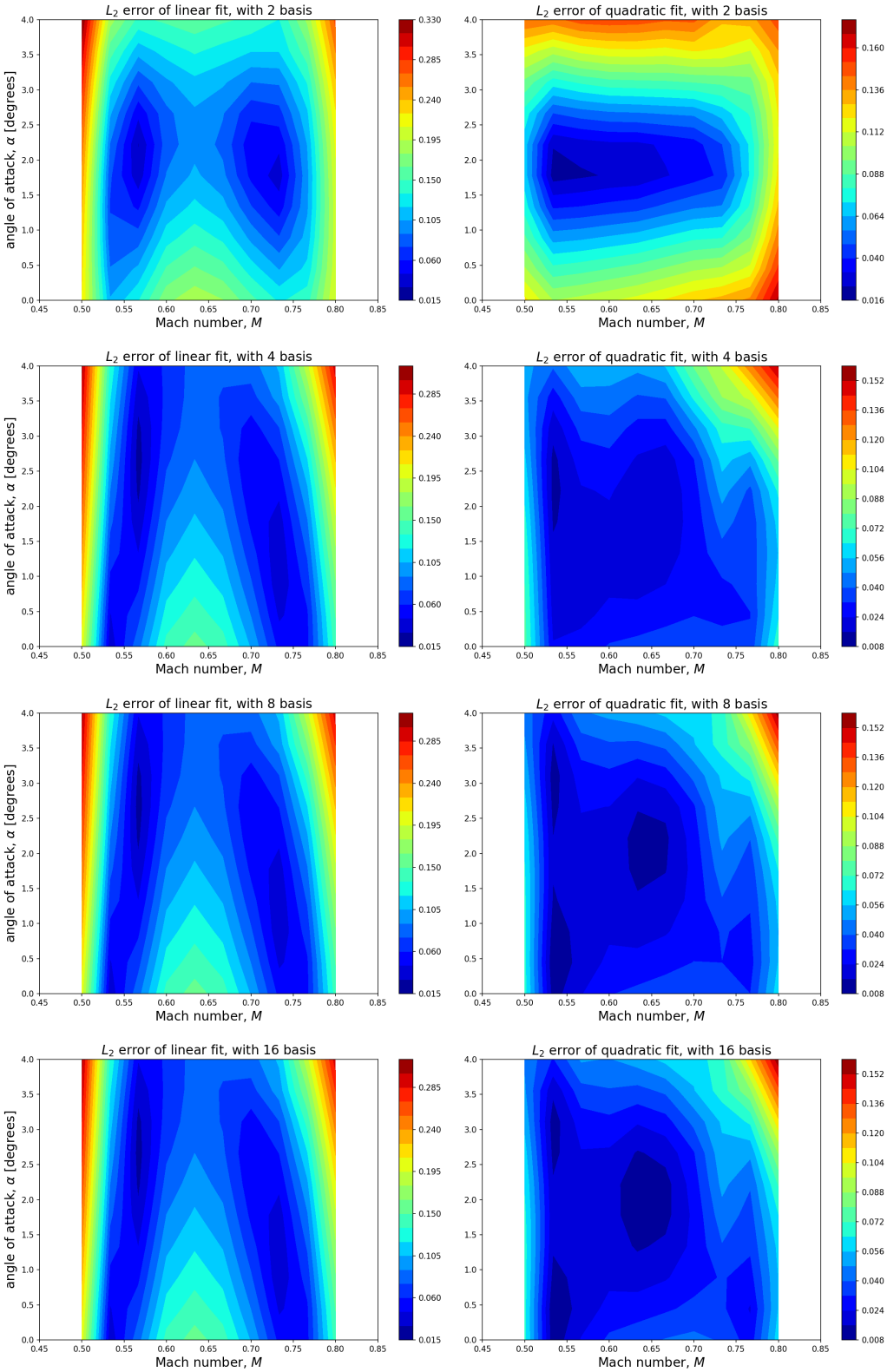

Figure 12. L_2 **error of linear model and quadratic model,** $n_b = 2, 4, 8, 16$**.**

For comparison, the Mach number contours of the linear and quadratic model-approximated states for snapshot 55 are depicted in figure 13. Comparing the first row in figure 13 with the left hand side of figure 3, we can see that the accuracy of linear approximation is quite limited even when using 16 basis. Fortunately, The quadratic fitting of test snapshot 55 has a satisfactory performance. As depicted in the second row of figure 13, when using 8 basis to represent the state vectors, the approximated Mach number contour is already quite similar with the exact one depicted in figure 3.

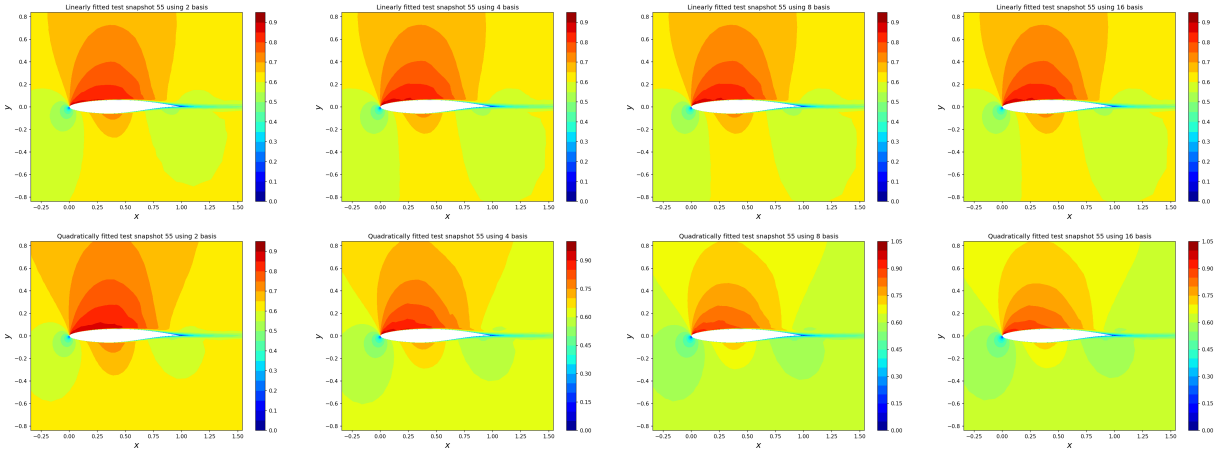

Figure 13. Mach number contours of linear approximation (1^{st} row) and quadratic approximation (2^{nd} row), test snapshot 55, $n_b = 2, 4, 8, 16$.

As for the approximation of test snapshot 90, the error is larger comparing with test snapshot 55. It is not surprising since test snapshot 90 is in the high Mach number, large attack angle region, where the L_2 error is relatively high. According to the Mach number contours of linear approximation shown in figure 14, the error mainly come from the over estimation of Mach number on the upper surface near the leading edge. Although the quadratic approximation for test snapshot 90 does provide a better result, the over estimation of Mach number on the upper surface of airfoil still exist.

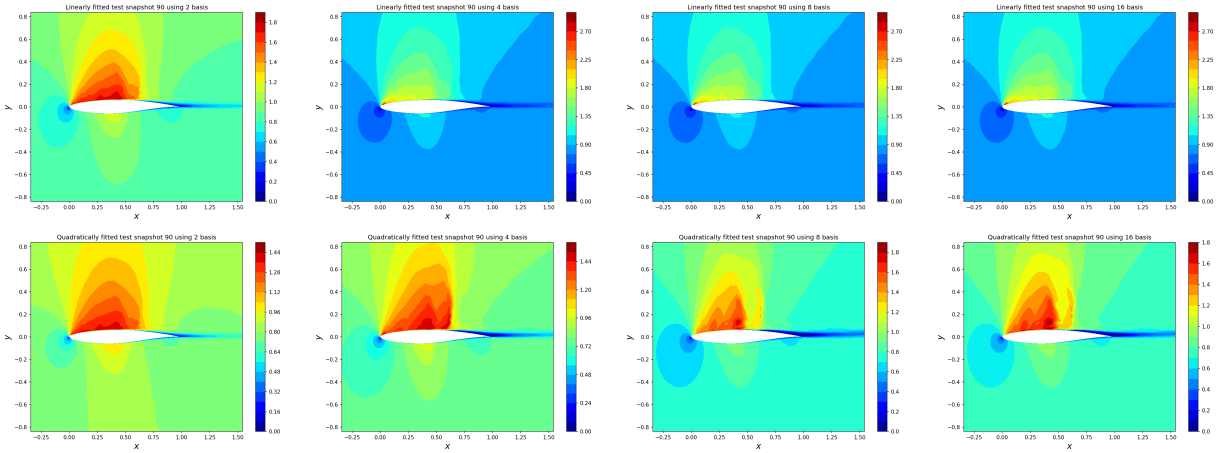

Figure 14. Mach number contours of linear approximation (1^{st} row) and quadratic approximation (2^{nd} row), test snapshot 90, $n_b = 2, 4, 8, 16$.