SANDIA REPORT

SAND2013-7051
Unlimited Release
Printed August 2013

Construction of Reduced Order Models for the Non-Linear Navier-Stokes Equations using the Proper Orthogonal Decomposition (POD)/Galerkin Method

Jeffrey A. Fike

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

SAND2013-7051
Unlimited Release
Printed August 2013

Construction of Reduced Order Models for the Non-Linear Navier-Stokes Equations using the Proper Orthogonal Decomposition (POD)/Galerkin Method

Jeffrey A. Fike

Abstract

The construction of stable reduced order models using Galerkin projection for the Euler or Navier-Stokes equations requires a suitable choice for the inner product. The standard L2 inner product is expected to produce unstable ROMs. For the non-linear Navier-Stokes equations this means the use of an energy inner product. In this report, Galerkin projection for the non-linear Navier-Stokes equations using the L2 inner product is implemented as a first step toward constructing stable ROMs for this set of physics.

Acknowledgment

This work was funded by the U.S. Department of Energy's Predictive Science Academic Alliance Program (PSAAP) Center at Stanford University.

This work was performed under the supervision and guidance of Irina Kalashnikova.

Contents

1 Introduction **9**

2 Galerkin Projection for the Non-Linear Navier-Stokes Equations **11**

3 Implementation in SPIRIT and MATLAB **15**

Integration Using Quadrature 15

Specifying Zero Base Flow 16

Galerkin Projection onto the Non-Linear Navier-Stokes Equations 16

Time Integration of the ROM 17

4 Discussion of Results **19**

Description of 2DViscPulseBox Problem 19

Results 20

5 Conclusions and Future Work **23**

References **24**

List of Figures

4.1 A comparison of the ROM coefficients with the projections of the snapshots onto the POD basis. 20

4.2 u-velocity at time $t = 1.5 \times 10^{-3}$ seconds 21

4.3 u-velocity at time $t = 9.2 \times 10^{-3}$ seconds 21

Chapter 1

Introduction

The proper orthogonal decomposition (POD) can be used to construct reduced order models (ROMs) for the Euler and Navier-Stokes equations. Discussed in detail in Lumley [7] and Holmes *et al.* [4], POD is a mathematical procedure that, given an ensemble of data and an inner product, constructs a basis for that ensemble that is optimal in the sense that it describes more energy (on average) of the ensemble in the chosen inner product than any other linear basis of the same dimension M. Stable ROMs based on the Galerkin projection of the fluid equations onto the POD modes requires specific definitions of the inner product [1, 8]. ROMs based on the simple L2 inner product can be unstable [1]. For the non-linear Navier-Stokes equations an energy inner product is required for stability [5]. Although the L2 inner product is expected to produce an unstable ROM, it is a logical first step toward developing a stable ROM for the non-linear Navier-Stokes equations.

The procedure for constructing a reduced order model using the POD method/Galerkin projection is as follows:

1. Collect snapshots of the state of the flow field from unsteady simulations of the problem of interest.
2. Perform singular value decomposition on these snapshots. The left eigenvectors are the POD modes.
3. Approximate the flow variables as a weighted sum of the POD modes.
4. Project the Navier-Stokes equations onto the POD modes.
5. Perform the necessary integrations over the domain in order to construct the coefficient matrices for the ROM.
6. Compute the ROM coefficients by integrating in time.

Chapter 2

Galerkin Projection for the Non-Linear Navier-Stokes Equations

The Navier-Stokes equations can be written in terms of density, velocity and temperature [2] to produce the continuity equation

$$\rho_{,t} + \rho_{,j} u_j + \rho u_{j,j} = 0, \tag{2.0.1}$$

three momentum equations

$$\rho u_{i,t} + \rho u_{i,j} u_j + R\left(\rho T\right)_{,i} - \tau_{ij,j} = 0,\ i = 1, 2, 3 \tag{2.0.2}$$

and the energy equation

$$\rho T_{,t} + \rho u_j T_{,j} + \left(\gamma - 1\right) u_{j,j} \rho T + \frac{\left(\gamma - 1\right)}{R}\left(-\kappa T_{,j}\right)_{,j} - \frac{\left(\gamma - 1\right)}{R} u_{i,j} \tau_{ij} = 0. \tag{2.0.3}$$

These equations can be rewritten in non-dimensional form as

$$\rho_{,t} + \rho_{,j} u_j + \rho u_{j,j} = 0, \tag{2.0.4}$$

$$\rho u_{i,t} + \rho u_{i,j} u_j + \frac{1}{\gamma M_{ref}^2}\left(\rho T\right)_{,i} - \frac{1}{Re}\tau_{ij,j} = 0,\ i = 1, 2, 3 \tag{2.0.5}$$

and

$$\rho T_{,t} + \rho u_j T_{,j} + \left(\gamma - 1\right) u_{j,j} \rho T + \left(\frac{\gamma}{PrRe}\right)\left(-\kappa T_{,j}\right)_{,j} - \left(\frac{\gamma\left(\gamma - 1\right) M_{ref}^2}{Re}\right) u_{i,j} \tau_{ij} = 0, \tag{2.0.6}$$

for $I = 1, ..., M$, where all terms are non-dimensional and $M_{ref} = u_{ref}/c_{ref}$, and M denotes the size of the reduced POD basis.

A POD basis can be constructed using the method of snapshots. The Navier-Stokes equations can then be projected onto the POD modes using the L2 inner product, resulting in

$$\left\langle \left(\rho u_{i,t} + \rho u_{i,j} u_j + \frac{1}{\gamma M_{ref}^2} \rho_{,i} T + \frac{1}{\gamma M_{ref}^2} \rho T_{,i} - \frac{1}{Re} \tau_{ij,j} \right), \phi_I^{u_i} \right\rangle$$
$$+ \left\langle \left(\rho T_{,t} + \rho u_j T_{,j} + (\gamma - 1) u_{j,j} \rho T + \left(\frac{\gamma}{Pr Re} \right) (-\kappa T_{,j})_{,j} - \left(\frac{\gamma (\gamma - 1) M_{ref}^2}{Re} \right) u_{i,j} \tau_{ij} \right), \phi_I^T \right\rangle$$
$$+ \left\langle (\rho_{,t} + \rho_{,j} u_j + \rho u_{j,j}), \phi_I^\rho \right\rangle = 0, \quad (2.0.7)$$

where $\langle u, v \rangle$ denotes the inner product.

The flow variables can then be approximated using the POD modes:

$$\rho \approx \sum_{k=1}^{M} a_k \phi_k^\rho, \quad (2.0.8)$$

$$u_i \approx \sum_{k=1}^{M} a_k \phi_k^{u_i}, \ i = 1, 2, 3 \quad (2.0.9)$$

and

$$T \approx \sum_{k=1}^{M} a_k \phi_k^T. \quad (2.0.10)$$

These approximations can be inserted into the Galerkin projection to produce the following terms:

$$\langle \rho_{,t}, \phi_I^\rho \rangle = \sum_{k=1}^{M} \dot{a}_k \langle \phi_k^\rho, \phi_I^\rho \rangle, \quad (2.0.11)$$

$$\langle \rho u_{i,t}, \phi_I^{u_i} \rangle = \sum_{k=1}^{M} \sum_{q=1}^{M} \dot{a}_k a_q \langle \phi_q^\rho \phi_k^{u_i}, \phi_I^{u_i} \rangle, \quad (2.0.12)$$

$$\langle \rho T_{,t}, \phi_I^T \rangle = \sum_{k=1}^{M} \sum_{q=1}^{M} \dot{a}_k a_q \langle \phi_q^\rho \phi_k^T, \phi_I^T \rangle, \quad (2.0.13)$$

$$\langle \rho_{,j} u_j, \phi_I^\rho \rangle = \sum_{k=1}^{M} \sum_{q=1}^{M} a_k a_q \langle \phi_{k,j}^\rho \phi_q^{u_j}, \phi_I^\rho \rangle, \quad (2.0.14)$$

$$\langle \rho u_{j,j}, \phi_I^\rho \rangle = \sum_{k=1}^{M} \sum_{q=1}^{M} a_k a_q \langle \phi_k^\rho \phi_{q,j}^{u_j}, \phi_I^\rho \rangle, \quad (2.0.15)$$

$$\langle \rho_{,i} T, \phi_I^{u_i} \rangle = \sum_{q=1}^{M} \sum_{r=1}^{M} a_q a_r \langle \phi_{q,j}^\rho \phi_r^T, \phi_I^{u_i} \rangle, \quad (2.0.16)$$

$$\langle \rho T_{,i}, \phi_I^{u_i} \rangle = \sum_{q=1}^{M} \sum_{r=1}^{M} a_q a_r \langle \phi_q^\rho \phi_{r,i}^T, \phi_I^{u_i} \rangle, \quad (2.0.17)$$

$$\left\langle u_{k,j}\tau_{kj}, \phi_I^T \right\rangle = \left\langle \sum_{q=1}^{M} a_q \phi_{q,j}^{u_k} \tau_{kj}, \phi_I^T \right\rangle, \tag{2.0.18}$$

$$\left\langle \rho u_{i,j} u_j, \phi_I^{u_i} \right\rangle = \sum_{k=1}^{M}\sum_{q=1}^{M}\sum_{r=1}^{M} a_k a_q a_r \left\langle \phi_q^{\rho} \phi_{k,j}^{u_i} \phi_r^{u_j}, \phi_I^{u_i} \right\rangle, \tag{2.0.19}$$

$$\left\langle \rho u_j T_{,j}, \phi_I^T \right\rangle = \sum_{k=1}^{M}\sum_{q=1}^{M}\sum_{r=1}^{M} a_k a_q a_r \left\langle \phi_q^{\rho} \phi_r^{u_j} \phi_{k,j}^{T}, \phi_I^{T} \right\rangle, \tag{2.0.20}$$

$$\left\langle \rho u_{j,j} T, \phi_I^T \right\rangle = \sum_{k=1}^{M}\sum_{q=1}^{M}\sum_{r=1}^{M} a_k a_q a_r \left\langle \phi_q^{\rho} \phi_{r,j}^{u_j} \phi_k^{T}, \phi_I^{T} \right\rangle, \tag{2.0.21}$$

for $I = 1, ..., M$. The following two terms are produced by performing integration by parts

$$\left\langle \tau_{ij,j}, \phi_I^{u_i} \right\rangle = \left\langle -\tau_{ij}, \phi_{I,j}^{u_i} \right\rangle, \tag{2.0.22}$$

and

$$\left\langle \left(-T_{,j}\right)_{,j}, \phi_I^T \right\rangle = \left\langle T_{,j}, \phi_{I,j}^T \right\rangle = \left\langle \sum_{k=1}^{M} a_k \phi_{k,j}^T, \phi_{I,j}^T \right\rangle = \sum_{k=1}^{M} a_k \left\langle \phi_{k,j}^T, \phi_{I,j}^T \right\rangle, \tag{2.0.23}$$

for $I = 1, ..., M$. Substituting these terms in 2.0.7 produces

$$\begin{aligned}
&\sum_{k=1}^{M} \dot{a}_k \left\langle \phi_k^{\rho}, \phi_I^{\rho} \right\rangle + \sum_{k=1}^{M}\sum_{q=1}^{M} \dot{a}_k a_q \left(\left\langle \phi_q^{\rho} \phi_k^{u_i}, \phi_I^{u_i} \right\rangle + \left\langle \phi_q^{\rho} \phi_k^{T}, \phi_I^{T} \right\rangle \right) \\
&= \frac{\gamma\kappa}{PrRe} \sum_{k=1}^{M} a_k \left\langle \phi_{k,j}^T, \phi_{I,j}^T \right\rangle + \frac{1}{Re} \left\langle -\tau_{ij}, \phi_{I,j}^{u_i} \right\rangle + \left(\frac{\gamma\left(\gamma-1\right) M_{ref}^2}{Re} \right) \left\langle \sum_{q=1}^{M} a_q \phi_{q,j}^{u_k} \tau_{kj}, \phi_I^T \right\rangle \\
&+ \sum_{k=1}^{M}\sum_{q=1}^{M} a_k a_q \left(\left\langle \phi_{k,j}^{\rho} \phi_q^{u_j}, \phi_I^{\rho} \right\rangle + \left\langle \phi_k^{\rho} \phi_{q,j}^{u_j}, \phi_I^{\rho} \right\rangle + \frac{1}{\gamma M_{ref}^2} \left\langle \phi_{k,j}^{\rho} \phi_q^{T}, \phi_I^{u_i} \right\rangle + \frac{1}{\gamma M_{ref}^2} \left\langle \phi_k^{\rho} \phi_{q,i}^{T}, \phi_I^{u_i} \right\rangle \right) \\
&+ \sum_{k=1}^{M}\sum_{q=1}^{M}\sum_{r=1}^{M} a_k a_q a_r \left(\left\langle \phi_q^{\rho} \phi_{k,j}^{u_i} \phi_r^{u_j}, \phi_I^{u_i} \right\rangle + \left\langle \phi_q^{\rho} \phi_r^{u_j} \phi_{k,j}^{T}, \phi_I^{T} \right\rangle + \left(\gamma-1\right) \left\langle \phi_q^{\rho} \phi_{r,j}^{u_j} \phi_k^{T}, \phi_I^{T} \right\rangle \right),
\end{aligned} \tag{2.0.24}$$

for $I = 1, ..., M$. Equation (2.0.24) can be rewritten in the form

$$\left[M_{ik} + \sum_{q=1}^{m} M_{ikq} a_q \right] \left\{ \dot{a}_k \right\} = \left\{ \sum_{k=1}^{m} A_{ik} a_k + \sum_{k=1}^{m}\sum_{q=1}^{m} A_{ikq} a_k a_q + \sum_{k=1}^{m}\sum_{q=1}^{m}\sum_{r=1}^{m} A_{ikqr} a_k a_q a_r \right\}. \tag{2.0.25}$$

The POD coefficients a_k can be found by integrating this equation in time.

Chapter 3

Implementation in SPIRIT and MATLAB

This section describes the changes made to the code SPIRIT in order to apply the POD/Galerkin method to the non-linear Navier-Stokes equations, and the MATLAB routines used to integrate the ROM system (2.0.25) forward in time. The SPIRIT code is a parallel C++ code that uses distributed vector and matrix data structures and parallel eigensolvers from the Trilinos project [3] and basis functions/quadrature routines from the libmesh finite element library [6].

Integration Using Quadrature

The following functions were added to the file *EpetraInterfaceTetMesh.C* to perform the integrations required by the projection onto the non-linear Navier-Stokes equations. These functions use quadrature to accurately compute the integrals of up to cubic functions defined in each element. The POD modes are defined at the nodes of the elements and are therefore linear inside the elements. The gradients of the POD modes are defined inside the elements and are therefore constant throughout the element volume.

IntegrateUsingQuadrature
There are three overloaded versions of this function. The different versions take in one, two or three nodal functions and one element function. This allows the product of up to three modes and one gradient to be integrated accurately using quadrature. Inside this function, the product of the modes is integrated over each tetrahedron and then summed to give the total integral over the domain. This function relies on the definition of the quadrature weights computed by the function ComputeTetQuadratureCoefficients.

AssembleIntegrand_Quadrature
This function is similar to the function IntegrateUsingQuadrature in that there are three versions to account for the varying number of nodal functions and that integration is performed using quadrature. The primary difference is that this function computes an integrand and stores it at the nodes. The summation over the domain is performed

outside this function. This function was created to be in the style of the previous linearized implementations in SPIRIT and is meant as a replacement for the function AssembleIntegrand. However, these functions are not used in the non-linear Navier-Stokes equations (IntegrateUsingQuadrature is used instead).

ComputeTetQuadratureCoefficients
This function calls ComputeTetQuadratureCoefficients_Linear, ComputeTetQuadratureCoefficients_Quadratic, ComputeTetQuadratureCoefficients_Cubic for each element in order to compute the tetrahedral quadrature coefficients.

ComputeTetQuadratureCoefficients_Linear
This function computes the quadrature coefficients required to accurately integrate a linear function in a tetrahedral volume. This function uses libmesh to compute the quadrature weights.

ComputeTetQuadratureCoefficients_Quadratic
This function computes the quadrature coefficients required to accurately integrate a quadratic function in a tetrahedral volume. This function uses libmesh to compute the quadrature weights.

ComputeTetQuadratureCoefficients_Cubic
This function computes the quadrature coefficients required to accurately integrate a cubic function in a tetrahedral volume. This function uses libmesh to compute the quadrature weights.

Specifying Zero Base Flow

The implementations of the linearized Euler and Navier-Stokes compute a POD basis for perturbations of the flow variables about some base flow. For the non-linear Navier-Stokes equations the POD basis is computed for the actual full (mean plus fluctuation) flow variables. This requires that the routines to read in the snapshots be modified so that there is no base flow. Three overloaded functions are added to *POD.C* to accomplish this: ComputePODBasis, LoadSnapshots, and LoadBaseFlow. There is also the function SetBaseFlow in *FluidEqns.C*.

Galerkin Projection onto the Non-Linear Navier-Stokes Equations

The actual Galerkin projection for the non-linear Navier-Stokes equations is carried out by the function GalerkinProjectNonLinNSL2_QuadInt in *FluidEqns.C*. This function computes the various terms in 2.0.24 and stores them in the form of the tensors in 2.0.25.

Time Integration of the ROM

The ROM coefficients are found by performing time integration using a fourth-order Runge-Kutta (RK4) scheme in MATLAB. An existing RK4 script was modified for non-linear equations.

Chapter 4

Discussion of Results

The above implementation was run on the 2DViscPulseBox example in the SPIRIT repository. The objective of this run was to verify the new code.

Description of 2DViscPulseBox Problem

The test case considered is that of a 2D viscous acoustic pressure pulse in the following 2D prismatic domain: $\Omega = (-1, 1) \times (-1, 1) \in \mathbb{R}^2$. The base flow for this problem was uniform, with the following values: $\bar{p} = 1.01325$ Pa, $\bar{T} = 300$ K, $\bar{\rho} = \frac{\bar{p}}{R\bar{T}} = 1.175784 \times 10^{-5}$ kg/m^3, $\bar{u}_1 = \bar{u}_2 = 0.0$ m/s. The kinematic viscosity was set to a constant value: $\mu = 1.458 \times 10^{-4}$ Pa $\cdot$ s. This value of the viscosity corresponds to a Reynolds number of $Re = 28$ (based on a reference speed of sound of $c_{ref} = 347.1887$ m/s^2, a length scale of $L = 1$ m and a reference velocity $\rho_{ref} = \bar{\rho}$).

The high-fidelity fluid simulation data from which the POD basis was constructed were generated using a Sandia in-house finite volume flow solver known as SIGMA CFD. As both the high-fidelity code as well as the ROM code are 3D codes, a 2D mesh of the domain Ω was converted to a 3D mesh by extruding the 2D mesh in the z-direction by one element. The computational grid for this test case was composed of 3362 nodes, cast into 9600 tetrahedral finite elements within the ROM code. A no slip boundary condition was imposed on the four sides of the domain in the x and y plane. To ensure the solution has no dynamics in the z-direction, the following values of the z-velocity component was specified: $u_3 = 0$ m/s^2. Symmetry boundary conditions were imposed for $z =$ constant in the high-fidelity code. The high-fidelity simulation was initialized with a pressure pulse in the middle of the domain, and run until time 0.01 seconds. During this simulation, the initial pressure pulse reflected from the walls of the domain a number of times. Snapshots from the high-fidelity simulation were saved every 5×10^{-5} seconds, to yield a total of 200 snapshots. These snapshots were employed to construct a 10 mode POD basis, orthonormal with respect to the L2 inner product. Both the high-fidelity simulation as well as the ROM simulation were run in non-dimensional variables, with the velocities non-dimensionalized by c_{ref}, the density non-dimensionalized by ρ_{ref} and the pressure non-dimensionalized by $p_{ref} = \rho_{ref} c_{ref}^2 = 1.41729$ Pa.

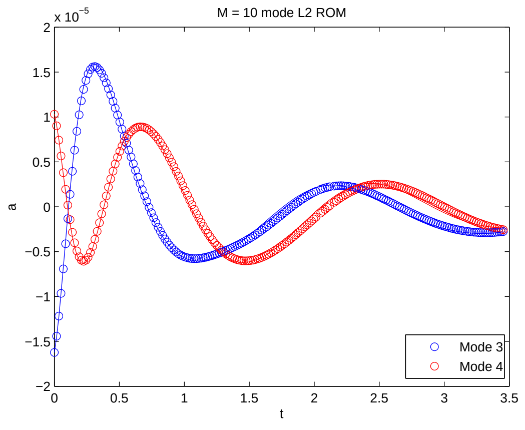
Results

The accuracy of the ROM can be visualized in a number of ways.

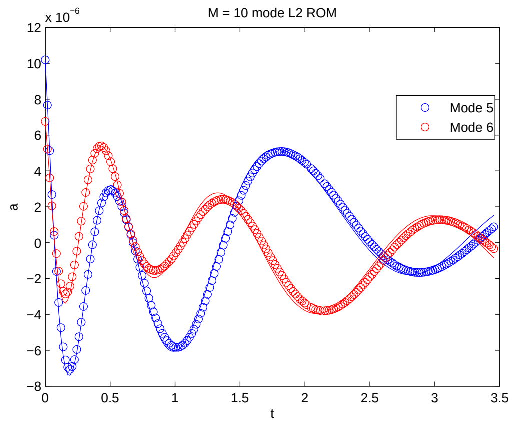
First, the ROM coefficients are compared to the projections of the snapshots onto the POD basis. Figure 4.1 shows a time history of the third, fourth, fifth and sixth ROM modal amplitudes (circles) compared to the projection of the full CFD simulation onto the third, forth, fifth and sixth POD modes (solid lines). Mathematically, this figure compares as a function of time t:

$$a_i(t) \quad \text{vs.} \quad \langle q_{\text{CFD}}, \phi_i \rangle, \quad i = 3, 4, 5, 6, \tag{4.0.1}$$

where $q_{\text{CFD}} \equiv \begin{pmatrix} u, & v, & w, & T, & \rho \end{pmatrix}^T$ is the high-fidelity CFD solution from which the ROM basis was constructed. There is good agreement between the ROM and the full simulation for the time interval considered. This provides one level of verification of the new code.



(a) Coefficients 3 and 4

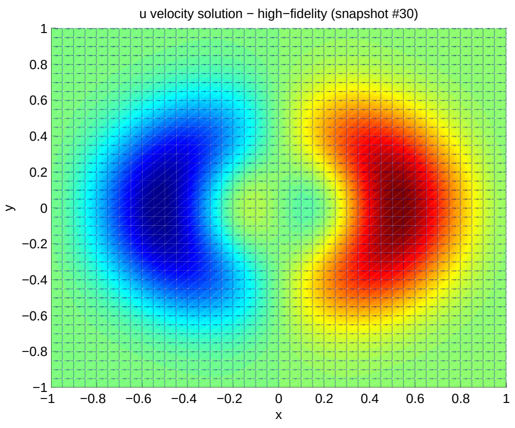


(b) Coefficients 5 and 6

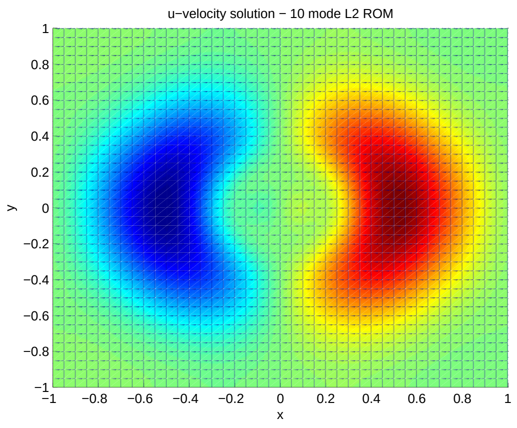
Figure 4.1. A comparison of the ROM coefficients with the projections of the snapshots onto the POD basis.

To further verify the implementation, as well as examine further the accuracy of the ROM, the ROM solution for the u-velocity is visualized and compared to the high-fidelity solution for the u-velocity at times $t = 1.5 \times 10^{-3}$ and 9.20×10^{-3} (Figures 4.2 and 4.3 respectively). The reader can observe that there is a good qualitative agreement between the high-fidelity solution and the ROM solution at both times. Some spurious features are observed in the solution at the later time (Figure 4.3), but this is expected for a ROM with as few modes as used here ($M = 10$). This provides further verification of the new code.

It is worthwhile to report that an instability in the ROM was observed when the ROM was constructed using dimensional variables, likely due to bad scaling. These results are omitted from this report.

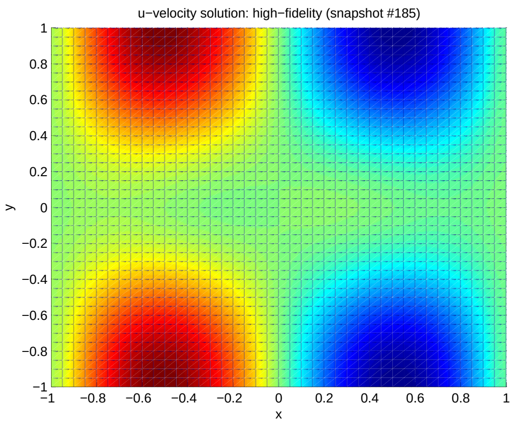


(a) High-fidelity CFD

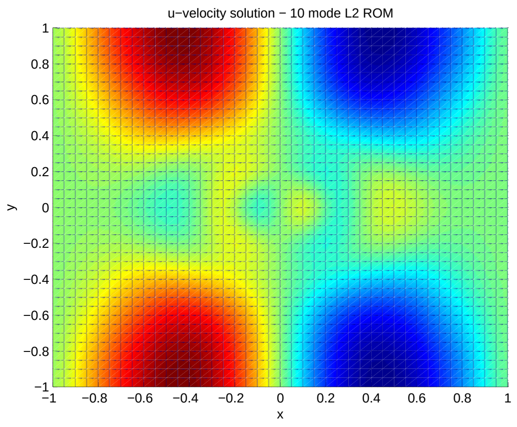


(b) 10 mode ROM

Figure 4.2. u-velocity at time $t = 1.5 \times 10^{-3}$ seconds



(a) High-fidelity CFD



(b) 10 mode ROM

Figure 4.3. u-velocity at time $t = 9.2 \times 10^{-3}$ seconds

Chapter 5

Conclusions and Future Work

The implementation of the L2 ROM for the Navier-Stokes equations produces good results. However, there is still more work to be done to fully test this implementation. After that, the next step would be to construct a ROM using the energy inner product.

Performing the Galerkin projection with an energy inner product will produce a stable ROM [5]. This requires writing the Navier-Stokes equations in a slightly different form:

$$a_{,t} + a_{,j}u_j + \frac{1}{2}au_{j,j} = 0, \tag{5.0.1}$$

$$a^2u_{i,t} + a^2u_{i,j}u_j + \frac{2}{\gamma}\left(a^2cc_{,i} - ac^2a_{,i},\right) - \tau_{ij,j} = 0,\ i = 1,2,3 \tag{5.0.2}$$

and

$$\frac{2a^2c}{\gamma(\gamma-1)}\left(c_{,t} + u_jc_{,j} + \frac{\gamma-1}{2}u_{j,j}c\right) - \frac{2\kappa}{\gamma R}\left(c_{,j}c_{,j} + c\left(c_{,j}\right)_{,j}\right) - u_{i,j}\tau_{ij} = 0, \tag{5.0.3}$$

where $a^2 = \rho$ and c is the speed of sound.

The energy inner product is defined as

$$\left(\mathbf{q}^{(1)}, \mathbf{q}^{(2)}\right)_E = \int_\Omega \left(\frac{1}{\gamma\left(\gamma-1\right)}a^{(1)}c^{(1)}a^{(2)}c^{(2)} + \frac{1}{2}u_i^{(1)}u_i^{(2)}\right) d\Omega. \tag{5.0.4}$$

References

[1] M. F. Barone, I. Kalashnikova, D. J. Segalman, and H. K. Thornquist. Stable Galerkin reduced order models for linearized compressible flow. *Journal of Computational Physics*, 228(6):1932–1946, 2009.

[2] S. S. Collis. *A Computational Investigation of Receptivity In High-Speed Flow Near a Swept Leading-Edge*. PhD thesis, Stanford University, Stanford, CA, 1997.

[3] M. A Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, Kolda T. G., Lehoucq R. B., Long K. R., Pawlowski R. P., Phipps E. T., Salinger A. G., Thornquist H. K., Tuminaro R. S., Willenbring J. M., Williams A., and Stanley K. S. An overview of the Trilinos project. *ACM Trans. Math. Sofw.*, 31(3), 2005.

[4] P. Holmes, J.L. Lumley, and G. Berkooz. *Turbulence, coherent structures, dynamical systems and symmetry*. Cambridge University Press, New York, NY, 1996.

[5] I. Kalashnikova. Stability-preserving energy inner product for full nonlinear Euler equations.

[6] B. Kirk, J. W. Peterson, R. H. Stogner, and G. F. Carey. libmesh: A C++ library for parallel adaptive mesh refinement/coarsening simulations. *Eng. Comput.*, 22(34):237–254, 2006.

[7] J.L. Lumley. *Stochastic tools in turbulence*. Academic Press, New York, NY, 1971.

[8] C. W Rowley, T. Colonius, and R. M. Murray. Model reduction for compressible flows using POD and Galerkin projection. *Physica D: Nonlinear Phenomena*, 189(1):115–129, 2004.

DISTRIBUTION:

1	MS 1320	Irina Kalashnikova, 1442
1	MS 1070	Matt Brake, 1526
1	MS 0825	Matt Barone, 1515
1	MS 0825	Srinivasan Arunajatesan, 1515
1	MS 1318	Bart van Bloemen Waanders, 1442
1	MS 0346	Dianne Peebles, 1526
		,
1	MS 0899	Technical Library, 9536 (electronic copy)

Sandia National Laboratories